VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

WEBOVÁ APLIKACE PRO HODNOCENÍ NÁSTROJŮ BEZPEČNOSTNÍHO TESTOVÁNÍ

WEB APPLICATION FOR EVALUATION OF SECURITY TESTING TOOLS

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

AUTOR PRÁCE — Vojtěch Moravec

AUTHOR

VEDOUCÍ PRÁCE — Ing. Petr Ilgner

SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Vojtěch Moravec ***ID:*** 219263

Ročník: 3 ***Akademický rok:*** 2023/24

NÁZEV TÉMATU:

Webová aplikace pro hodnocení nástrojů bezpečnostního testování

POKYNY PRO VYPRACOVÁNÍ:

Hlavním cílem bakalářské práce je vytvořit prokazatelně zranitelné prostředí ve formě webové aplikace, pomocí kterého bude možné efektivně měřit a porovnávat automatizované nástroje pro penetrační testování, skenování zranitelností a statickou bezpečnostní analýzu zdrojového kódu. Webová aplikace zahrne zranitelnosti ze všech kategorií žebříčku OWASP Top 10, přičemž každá zranitelnost bude označena a dostatečně dokumentována. Součástí bakalářské práce bude provedení benchmarkingu vybraných nástrojů nad vytvořenou zranitelnou aplikací.

DOPORUČENÁ LITERATURA:

[1] HOFFMAN, Andrew. Web Application security: exploitation and countermeasures for modern web applications. O'Reilly Media, 2020.

[2] WYSOPAL, Chris, et al. The art of software security testing: identifying software security flaws. Pearson Education, 2006.

Termín zadání: 5.2.2024 ***Termín odevzdání:*** 28.5.2024

Vedoucí práce: Ing. Petr Ilgner

doc. Ing. Jan Hajný, Ph.D.
předseda rady studijního programu

ABSTRAKT

Bakalářská práce se zabývá návrhem vývojem a implementací úmyslně zranitelného prostředí ve formě webové aplikace. Výsledná webová aplikace obsahuje zranitelnosti napříč všemi kategoriemi projektu OWASP Top 10, konkrétně poté jeho verze z roku 2021 a je skrze ni možné měřit a porovnávat automatizované nástroje pro penetrační testování stejně jako nástroje pro statickou bezpečnostní analýzu zdrojového kódu. Práce je rozčleněna do pěti kapitol ve kterých je na úvod popsána organizace OWASP Foundation pod jejíž záštitou projekt OWASP funguje, dále je popsána analýza vybraných úmyslně zranitelných webových aplikací. Další kapitoly se poté věnují návrhu vlastní úmyslně zranitelné webové aplikace, kde jsou mimo jiné popsány technologie použité při vývoji stejně jako všechny zranitelnosti v aplikaci obsažené. V závěru práce je poté provedeno testování výsledné zranitelné aplikace pomocí výše zmíněných nástrojů a shrnutí výsledků, kterých bylo dosaženo. Mimo již výše zmíněné testování a porovnávání automatizovaných nástrojů z oblasti penetračního testování a statické analýzy může být aplikace využita rovněž k výukovým účelům a to především díky přiloženým opravám a referencím, které jsou součástí každé zranitelnosti v aplikaci obsažené.

KLÍČOVÁ SLOVA

Webová aplikace, Bezpečnost webových aplikací, Informační bezpečnost, Python, Flask, PostgreSQL, OWASP, MVC, Docker, Zranitelnost.

ABSTRACT

The bachelor thesis focuses on the design, development, and implementation of an intentionally vulnerable environment in the form of a web application. The resulting web application encompasses vulnerabilities across categories outlined in the OWASP Top 10 project, specifically following its 2021 version. Through this application, it is possible to assess and compare automated tools for penetration testing, as well as tools for static code security analysis. The thesis is divided into five chapters.In the introduction, the OWASP Foundation, which oversees the OWASP project, is described. The analysis of selected intentionally vulnerable web applications is then presented. Subsequent chapters delve into the design of the custom intentionally vulnerable web application, detailing the technologies used in its development and outlining all vulnerabilities present in the application. In the conclusion of the thesis, testing of the resulting vulnerable application is conducted using the aforementioned tools, and a summary of the achieved results is provided. Apart from the aforementioned testing and comparison of automated tools in the fields of penetration testing and static analysis, the application can also be utilized for educational purposes. This is primarily facilitated by the attached fixes and explanations, which accompany each vulnerability within the application.

KEYWORDS

Web application, Web application security, Information security, Python, Flask, PostgreSQL, OWASP, MVC, Docker, Vulnerability.

MORAVEC, Vojtěch. *Webová aplikace pro hodnocení nástrojů bezpečnostního testování*. Bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2024. Vedoucí práce: Ing. Petr Ilgner

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Vojtěch Moravec
VUT ID autora:	219263
Typ práce:	Bakalářská práce
Akademický rok:	2023/24
Téma závěrečné práce:	Webová aplikace pro hodnocení nástrojů bezpečnostního testování

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu bakalářské práce panu Ing. Petru Ilgnerovi za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci. Stejnou měrou bych dále rád poděkoval odbornému konzultantovi panu Bc. Willimu Lazarovovi za poskytnuté rady a návrhy při řešení této práce.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů

Úvod

V dnešní digitální době jsou webové aplikace běžnou součástí života většiny populace [1]. Ať už se jedná o sociální sítě, internetové bankovnictví, nebo například online nákupy. Se zvyšujícím se zastoupením webových aplikací se však zvyšuje i četnost kybernetických útoků vůči takovým aplikacím a s tím související potřeba těmto útokům porozumět a umět jim předcházet [2].

V reakci na tuto problematiku se zvýšila [3, 4] také potřeba aplikací a platforem, které umožní demonstrovat zranitelnosti, různé typy útoků, stejně jako efektivní metody obrany. Tyto aplikace a platformy představují prostředí pro trénink a testování v oblasti informační bezpečnosti, umožňující odborníkům, studentům ale i nadšenců v této oblasti zdokonalovat své dovednosti a získávat praktické zkušenosti.

Cílem této práce bylo vytvořit úmyslně zranitelné prostředí ve formě webové aplikace zahrnující zranitelnosti z žebříčku OWASP (Open Worldwide Application Security Project) Top 10:2021 skrze které bude možné testování a porovnávání automatizovaných nástrojů pro penetrační testování stejně tak nástrojů pro statickou bezpečnostní analýzu zdrojového kódu.

Aplikace nese název Coffee Shop a reprezentuje online obchod s kávou, čaji a příslušenstvím pro jejich přípravu. Backendová část webové aplikace je vytvořena v programovacím jazyce Python s pomocí webového frameworku Flask, frontendová část poté pomocí HTML (HyperText Markup Language), CSS (Cascading Style Sheets), JavaScriptu a frontendového frameworku Bootstrap. Jako datové uložiště aplikace slouží databázový systém PostgreSQL. Aplikace je kompletně konteinerizovaná a lze ji tak spustit pomocí nástroje Docker.

Bakalářská práce je rozdělena do pěti kapitol. První kapitola poskytuje úvod do projektu OWASP, jeho historii poskytuje stručný popis stěžejních projektů. Druhá kapitola se poté zabývá analýzou úmyslně zranitelných webových aplikací přičemž nalezené zranitelnosti jsou kategorizovány dle OWASP Top 10, konkrétně poté dle verze z roku 2021. Návrh vlastní úmyslně zranitelné webové aplikace je obsahem třetí kapitoly. Čtvrtá kapitola hlouběji pojednává o zranitelnostech obsažených v aplikaci. Poslední pátá kapitole se poté věnuje testování zranitelností pomocí platforem GitHub, GitLab a Snyk a jejich následnému porovnání.

1 OWASP

OWASP je komunita fungující pod záštitou neziskové organizace OWASP Foundation zabývající se bezpečností software. Organizace OWASP Foundation byla založena v roce 2001 a je v současnosti největší neziskovou organizací zabývající se bezpečností software na světě.

OWASP pomáhá organizacím a jednotlivcům vyvíjet, provozovat a udržovat aplikace bezpečné, popřípadě zvýšit bezpečnostní úroveň aplikací již existujících a to skrze metodiky, dokumenty a nástroje, které jsou organizací OWASP publikovány jako volně dostupné [5, 6].

1.1 Projekty OWASP Foundation

Jak již bylo zmíněno, projekty jsou OWASP komunitou vyvíjeny a publikovány jako volně dostupné. Projekty jsou rozčleněny do třech kategorií a to *Vlajkové*, jež mají významný přínos pro OWASP a bezpečnost obecně (o vybraných projektech z této kategorie pojednává podkapitola 1.1). Dále *Produkční*, kam spadají projekty připravené k nasazení v produkčním prostředí a *Ostatní* kam spadají mimo jiné projekty jejichž vývoj stále probíhá [7, 8].

1.2 Vlajkové projekty OWASP

Jak již bylo zmíněno výše, mezi kategorie *Vlajkové* se řadí projekty jímž je ze strany OWASP komunity věnována zvýšená pozornost z důvodu jejich přínosu na poli bezpečnosti [7]. Vybrané projekty jsou popsány v následujících podkapitolách.

OWASP Application Security Verification Standard

OWASP ASVS (Application Security Verification Standard) je projekt jehož cílem je poskytnout otevřený bezpečnostní standard poskytující základy bezpečné návrhu, vývoje a testování webových služeb a aplikací [9].

OWASP Mobile Application Security

OWASP MAS (Mobile Application Security) je projektem zaměřeným na bezpečnost mobilních aplikací skládající se ze dvou částí a to OWASP MASVS (Mobile Application Security Verification Standard) a OWASP MASTG (Mobile Application Security Testing Guide) jež jsou popsány v následujících odstavcích [10].

OWASP MASVS, je standardem bezpečnosti mobilních aplikací a je využíván k zajištění kompletních a konzistentních výsledků testů mobilních aplikací z pohledu bezpečnosti [11].

OWASP MASTG je manuálem pro bezpečnostní testování a reverzní inženýrství mobilních aplikací [12].

OWASP Top Ten

OWASP Top Ten popřípadě také OWASP Top 10 je dokument shrnující nejzávažnější rizika z pohledu bezpečnosti webových aplikací a člení je do deseti kategorií. Ke každé kategorii je mimo jiné přiřazen popis konkrétní kategorie, jak předcházet rizikům s ní spojených, ale především seznam CWE (Common Weakness Enumeration) asociovaný s danou kategorií [13, 14].

CWE je komunitou vytvořený a spravovaný seznam hardwarových a softwarových bezpečnostních slabin, které mohou za určitých okolností vést ke zranitelnostem. Projekt je provozován společností MITRE Corporation za podpory DHS (U.S. Department of Homeland Security) a Cybersecurity and Infrastructure Security Agency (CISA) [15].

S jednotlivými záznamy CWE jsou dále asociovány CVE (Common Vulnerabilities and Exposures) záznamy, které již představují konkrétní známé zranitelnosti. Projekt CVE je řízen výhradně organizací MITRE Corporation [16, 17].

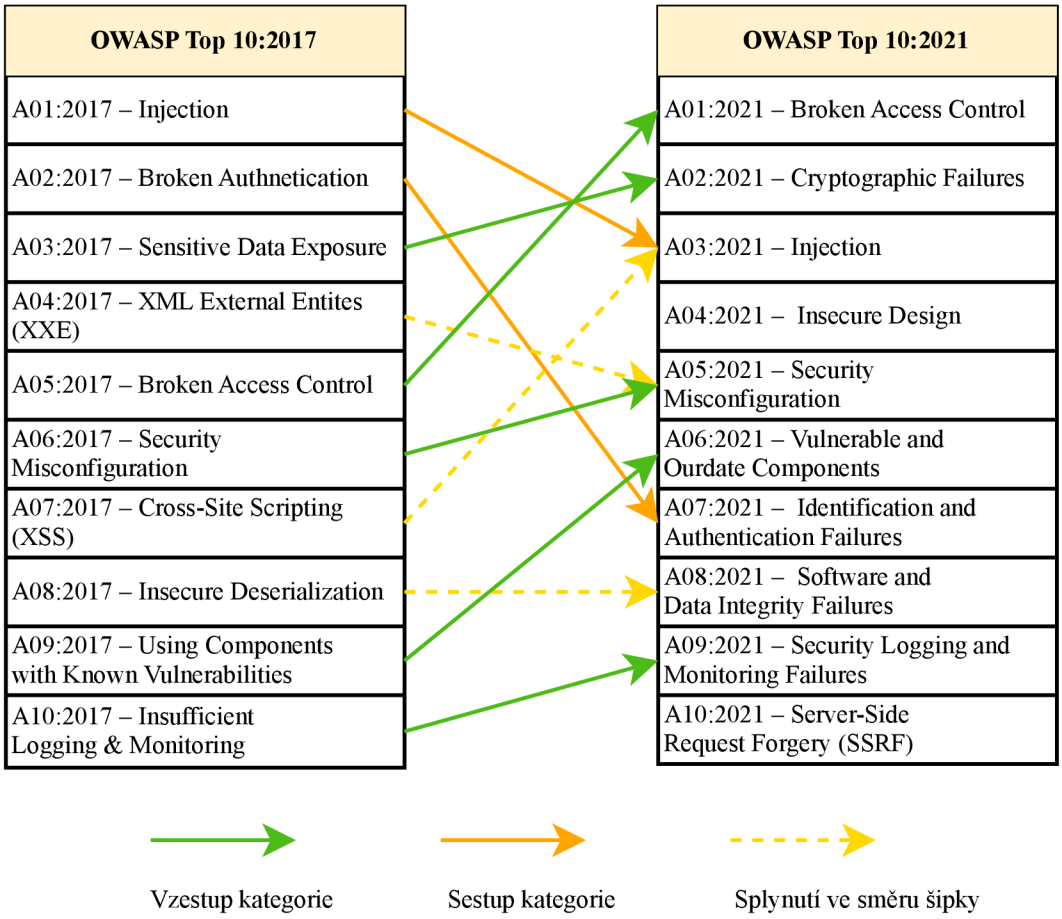
Dokument OWASP Top 10 je aktualizován každé tři až čtyři roky [18]. Současná verze OWASP Top 10:2021 je hlouběji rozebrána v kapitole 1.3.

OWASP Juice Shop

OWASP Juice Shop je úmyslně zranitelná webová aplikace, kterou lze použít pro účely demonstrace bezpečnostních slabin a zranitelností, výuky, testování bezpečnostních nástrojů, ale také jako CTF (Capture The Flag) scénář. OWASP Juice Shop obsahuje kromě všech bezpečnostních zranitelností z OWASP Top 10 také zranitelnosti a slabiny například z OWASP API Security Top 10, nebo Common Weakness Enumeration společnosti MITRE [19].

1.3 OWASP Top 10:2021

Jak již bylo v předcházející kapitole zmíněno, OWASP Top 10 shrnuje nejzávažnější rizika webových aplikací a poskytuje náměty jak jim předcházet. Obrázek 1.1 ilustruje změny v aktuální verzi dokumentu oproti předešlé verzi z roku 2017. Jednotlivé kategorie OWASP Top 10:2021 jsou hlouběji rozebrány níže.



Obr. 1.1: Srovnání OWASP Top 10:2017 a OWASP Top 10:2021 [20].

A01:2021 – Broken Access Control

Představuje kategorii zabývající se slabinami souvisejícími s řízení přístupu. Řízení přístupu při správné implementaci zajišťuje, že uživatel je v rámci aplikace schopen provádět pouze úkony v rámci přidělených oprávnění. Naopak nesprávně fungující řízení přístupu běžně vede k neoprávněnému přístupu, manipulaci popřípadě i zničení data a také k možnosti provedení operace v rámci aplikace, která je mimo jeho

oprávnění. Mezi nejvýznamnější bezpečnostní slabiny spojené s touto kategorií se řadí například [21]:

- CWE-200: Exposure of Sensitive Information to an Unauthorized Actor,
- CWE-201: Insertion of Sensitive Information Into Sent Data,
- CWE-352: Cross-Site Request Forgery.

A02:2021 – Cryptographic Failures

Představuje kategorii slabin souvisejících s kryptografickými prostředky v rámci aplikace. Kryptografie je v aplikacích využívána za účelem ochrany aktiv ať už se jedná o uživatelské údaje, lékařské záznamy, nebo například obchodní tajemství. Kryptografie může být v rámci aplikace realizována pomocí kryptografických protokolů, jako příklad lze uvést protokol TLS (Transport Layer Security), který je použit v přenosovém protokolu HTTPS (Hypertext Transfer Protocol Secure) kde se stará o šifrování komunikace mezi klientem a serverem. Dalším z mnoha příkladů užití kryptografie v rámci aplikací jsou hašovací funkce mezi, které patří například Argon2 běžně používaná pro hašování hesel. Mezi slabiny z této kategorie lze zařadit například [22]:

- CWE-259: Use of Hard-coded Password,
- CWE-327: Broken or Risky Crypto Algorithm,
- CWE-331 Insufficient Entropy.

A03:2021 – Injection

Představuje kategorii slabin, které nastávají v případě, kdy aplikace akceptuje uživatelský vstup popřípadě jiná data, která mohou být potenciálně nebezpečná bez předcházející validace, filtrace, popřípadě sanitizace. Slabiny z této kategorie se často nacházení v SQL (Structure Query Language) dotazech, nebo například příkazech operačního systému. Mezi významné bezpečnostní slabiny řadící se do této kategorie lze zmínit například [23, 24]:

- CWE-79: Cross-site Scripting,
- CWE-89: SQL Injection,
- CWE-73: External Control of File Name or Path.

A04:2021 – Insecure Design

Představuje kategorii vztahující se ke slabinám, vzniklým během procesu návrhu. V procesu návrhu by měl být kladen důraz na to, aby aplikace a její komponenty splňovaly minimální požadavky s ohledem na bezpečnost stanovené organizací. Z pohledu výše zmíněné bezpečnosti je nutné uvažovat *Kontext*, tedy co je účelem aplikace, jak zapadá do ekosystému společnosti, jaká data bude obsahovat a jaká data

popřípadě části aplikace budou ze své podstaty čelit většímu riziku ze strany útočníka, například heslo uživatele uložené v databázi je pro útočníka cennější než například přezdívka asociovaná s jeho účtem. Dále je nutné uvažovat *Komponenty*, které aplikace používá a jakou úrovní bezpečnosti tyto komponenty disponují. Mezi komponenty se řadí například knihovny, balíky a podobně. Dalšími prvky, které je nutné zvážit jsou *Souvislosti* tedy jak spolu jednotlivé komponenty spolupracují. Dále je nutné uvažovat rozdělení infrastruktury do vrstev jelikož každá část může vyžadovat jinou úroveň zabezpečení. *Kód*, který by měl v rámci bezpečnosti být mimo jiné schopen například ověřovat uživatelské vstupy, používat kryptografické funkce a protokoly a neměl by obsahovat, hesla, šifrovací klíče ani jiná tajemství. Posledním prvkem který je třeba v rámci návrhu bezpečné aplikace mít na paměti je *Konfigurace* jelikož nesprávně nakonfigurovaná i když jinak bezpečná aplikace může představovat bezpečnostní slabinu. Mezi příklady takových slabin lze uvést například [25, 26]:

- CWE-209: Generation of Error Message Containing Sensitive Information,
- CWE-256: Unprotected Storage of Credentials,
- CWE-522: Insufficiently Protected Credentials.

A05:2021 – Security Misconfiguration

Představuje kategorii zabývající se bezpečnostními slabinami vznikajícími ve fázi konfigurace. Příkladem bezpečnostních slabin řadících se do této kategorie jsou [28]:

- CWE-260 Password in Configuration File,
- CWE-526 Exposure of Sensitive Information Through Environmental Variables
- CWE-756 Missing Custom Error Page.

A06:2021 – Vulnerable and Outdated Components

Představuje kategorii zaměřující se na bezpečnostní slabiny plynoucí z užívání zranitelného, zastaralého popřípadě již zcela nepodporovaného softwaru, může se jednat o operační systém, webový server, aplikace a podobně. Slabiny spadající do této kategorie mohou vzniknout i vlivem používání produktů, organizace která dostatečně často neaktualizuje a nevydává bezpečnostní záplaty popřípadě pokud aktualizace či záplaty nejsou organizací používající výsledný software dostatečně často instalovány. Mezi bezpečnostní slabiny lze v rámci této kategorie zmínit především [29]:

- CWE-937 OWASP Top 10 2013: Using Components with Known Vulnerabilities,
- CWE-1104 Use of Unmaintained Third Party Components.

A07:2021 – Identification and Authentication Failures

Představuje kategorii, která se věnuje slabinám v procesech *Identifikace*, která slouží k identifikování konkrétního uživatele například skrze uživatelské jméno či ID a *Autentizace*, což je proces sloužící k prokázání identity uživatele, například pomocí hesla popřípadě kombinace uživatelského jména a hesla. Do této kategorie spadají také slabiny související s řízením relace. Řízením relace je proces zodpovědný za bezpečnou výměnu HTTP (Hypertext Transfer Protocol) požadavků a odpovědí mezi klientem a serverem v rámci jejich probíhající relace. Příkladem slabin spadajících do této kategorie mohou být [30, 31]:

- CWE-384: Session Fixation,
- CWE-521 Weak Password Requirements,
- CWE-613: Insufficient Session Expiration.

A08:2021 – Software and Data Integrity Failures

Představuje kategorii v rámci OWASP Top 10, která se věnuje bezpečnostním slabinám plynoucím z narušení integrity softwaru či dat z důvodu použití knihoven, modulů, balíků a kódu třetích stran obecně z neověřených zdrojů, popřípadě repositářů, které mohou obsahovat škodlivý kód například vlivem špatného řízení přístupu nebo nedostatečně zabezpečeného CI/CD (Continuous Integration and Continuous Delivery) procesu, skrze který může být útočník schopen přidat svůj škodlivý kód popřípadě provádět modifikace v kódu původním. Narušení integrity může nastat též v procesu kódování a serializace do struktury, kterou je útočník schopen přečíst a následně modifikovat. Mezi bezpečnostní slabiny patřící do této kategorie se řadí například [32]:

- CWE-502: Deserialization of Untrusted Data,
- CWE-353 Missing Support for Integrity Check.

A09:2021 – Security Logging and Monitoring

Představuje kategorii zaměřující se na slabiny, které nastávají vlivem neschopnosti efektivně detekovat a reagovat na bezpečnostní incidenty, zaznamenávat události související s bezpečností, jako například mnoho neúspěšných pokusů o přihlášení k účtu ze stejné IP adresy v krátkém čase. Bezpečnostními slabinami z této kategorie jsou mimo jiné [33]:

- CWE-778 Insufficient Logging,
- CWE-223 Omission of Security-relevant Information.

A10:2021 – Server-Side Request Forgery

Zkráceně také SSRF (Server-Side Request Forgery) je kategorie v rámci OWASP Top 10, zabývající se slabinami, které vznikají ve případě, že aplikace umožňuje načítání vzdálených zdrojů z webové adresy poskytnuté uživatelem avšak bez dostatečné validace takové adresy, výsledkem čehož může útočník poskytnout aplikaci webovou adresu, která je z jakéhokoli důvod v jeho zájmu a nechat aplikace provést HTTP požadavek na takovou adresu. Výše zmíněným postupem může být útočník schopen vyhnout se zabezpečení ve formě řízení přístupu či firewallu. Jedinou slabinou asociovanou s touto kategorií je CWE-918 SSRF (Server-Side Request Forgery) [34].

2 Srovnání úmyslně zranitelných webových aplikací

Následující kapitola se věnuje testování vybraných úmyslně zranitelných webových aplikací. V podkapitolách z nichž každá náleží jedné z aplikací jsou uvedeny základní informace o aplikaci, jak lze danou aplikaci spustit a kategorizace v aplikaci nalezených zranitelností dle OWASP Top 10:2021. Polední podkapitola obsahuje shrnutí a porovnání všech testovaných aplikací.

2.1 Damn Vulnerable Python Web Application

DVPWA (Damn Vulnerable Python Web Application) [35] je úmyslně zranitelná webová aplikace aplikace vytvořena v programovacím jazyce Python s pomocí templatovacího enginu Jinja, databázového systému PostgreSQL a datového úložiště Redis ve kterém aplikace ukládá relace.

Aplikace slouží k demonstraci bezpečnostních slabin a lze ji spustit pomocí zdrojového kódu jež je volně dostupný skrze repositář aplikace na platformě GitHub popřípadě pomocí nástroje Docker. Dle repositáře na platformě GitHub obsahuje následující bezpečnostní zranitelnosti, které jsou přiřazeny do příslušné kategorie OWASP Top 10:2021 v tabulce 2.1.

Tab. 2.1: Kategorizace zranitelností v DVPWA.

Zranitelnost	**Kategorie OWASP Top 10:2021**
Broken Cryptographic Algorithm	A02 – Cryptographic Failures
Insecure Transport	
SQL Injection	A03 – Injection
XSS (Cross-site Scripting)	
Session Fixation	A07 – Identification and Authentication Failure

2.2 Damn Simple Vulnerable Python Web Application

DSVPWA (Damn Simple Vulnerable Python Web Application) [36] je úmyslně zranitelná webová aplikace inspirována *Damn Small Vulnerable Web* a vytvořena v programovacím jazyce Python, značkovacím jazyce HTML a CSS stylech. Účelem aplikace je demonstrovat vybrané zranitelnosti z OWASP Top 10. Aplikaci je

jako v předchozím případě možné spustit pomocí nástroje Docker popřípadě pomocí zdrojového kódu, který je volně dostupný v repositáři aplikace na platformě GitHub. Zranitelnosti v aplikaci obsažené jsou shrnuty a kategorizovány dle OWASP Top 10:2021 v tabulce 2.2.

Tab. 2.2: Kategorizace zranitelností v DSVPWA.

Zranitelnost	**Kategorie OWASP Top 10:2021**
CSRF (Cross-Site Request Forgery)	A01 – Broken Access Control
Path Traversal	
EAR (Execution After Redirect)	
Open Redirect	
Insecure transport	A02 – Cryptograhic Failures
SQL Injection #1	A03 – Injection
SQL Injection #2	
Command Injection	
XSS - Reflected	
XSS - Stored	
Clickjacking	A04 – Insecure Design
Session Fixation	A07 – Identification and Authentication Failures
Session Hijacking	
Deserialization of Untrusted data	A08 – Software and Data Integrity Failures

2.3 Gruyere

Google Gruyere [37], popřípadě pouze Gruyere je úmyslně zranitelná webová aplikace vytvořená společností Google [38] za účelem demonstrace bezpečnostních slabin a zranitelností webových aplikací, jak je najít, zneužít a také jak jim předcházet. Gruyere představuje aplikaci která umožňuje uživateli mimo jiné publikovat textové příspěvky a nahrávat soubory. Aplikace je součástí laboratorní úlohy, která je rozdělena na části dle jednotlivých zranitelností kde každá z částí obsahuje stručný popis zranitelnosti, kroky, které vedou k jejímu nalezení, zneužití a jak konkrétní zranitelnosti či slabině předcházet. k testování aplikace lze přistupovat jako metodou black-box, tedy bez znalosti vnitřního fungování aplikace stejně tak white-box, tedy s kompletní znalostí vnitřního fungování aplikace. Gruyere je aplikace vytvořená v programovacím jazyce Python, značkovacím jazyce HTML a technologii AJAX

(Asynchronous JavaScript and XML), jež umožňuje asynchronní aktualizaci webových stránek skrze výměnu dat mezi klientem a serverem probíhající na pozadí. Aplikace je dostupná online a lze ji spustit přímo ve webovém prohlížeče, uživatel má taktéž možnost v prohlížeči projít a analyzovat zdrojový kód a v neposlední řadě je uživateli k dispozici možnost celý projekt stáhnout ve formátu ZIP a spustit ho lokálně [39]. Zranitelnosti obsažené v Gruyere jsou shrnuty a kategorizovány dle OWASP Top 10:2021 v tabulce 2.3.

Tab. 2.3: Kategorizace zranitelností v Gruyere.

Zranitelnost	**Kategorie OWASP Top 10:2021**
Privilege Elevation	A01 – Broken Access Control
CSRF	
Path Traversal	
Data Tampering via Path Travesal	
Forced Browsing	
Privilege Elevation	
XSS - File Upload	A03 – Injection
XSS - Reflected #1	
XSS - Reflected #2	
XSS - Stored #1	
XSS - Stored #2	
XSS- Stored #3	
Phishing via AJAX	
XSSI (Cross-Site Script Inclusion)	
Template Injection	
Remote Code Execution via Path Traversal	A05 – Security Misconfiguration
Information Disclosure #1	
Information Disclosure #2	
Information Disclosure	A08 – Software and Data Integrity Failures
DoS (Denial of Service) via AJAX	

2.4 HeadPage

HeadPage [40] je úmyslně zranitelná webová aplikace vytvořená za účelem testování a vzdělávání v oblasti bezpečnosti webových aplikací. Aplikace představuje sociální síť s možností vytváření profilu, přidávání veřejného popřípadě privátního obsahu asociovaného s konkrétním profile a též možnost prohlížení profilů ostatních uživatelů. Aplikace je vytvořena v programovacím jazyce Python za pomoci značkovacího jazyka HTML, kaskádových stylů CSS a lze ji spustit pomocí nástroje Docker popřípadě pomocí zdrojového kódu, který je volně dostupný na platformě GitHub. Zranitelnosti obsažené v aplikaci jsou jako v předchozích případech shrnuty a kategorizovány dle OWASP Top 10:2021 v tabulce 2.4.

Tab. 2.4: Kategorizace zranitelností v HeadPage.

Zranitelnost	**Kategorie OWASP Top 10:2021**
Open Redirect	A01 – Broken Access Control
Path Traversal	
Insecure Transport	A02 – Cryptograhic Failures
SQL Injection	A03 – Injection
XSS - Stored	
RCE (Remote Code Execution) Malicious File Upload	A04 – Insecure Design

2.5 Pygoat

Pygoat [41, 42] je úmyslně zranitelná webová aplikace kterou lze použít například pro výuku oblasti bezpečnosti webových aplikací, ale také demonstrovat vybrané webové zranitelnosti a jak jim předcházet. Aplikace je rozdělena do několika částí například „OWASP TOP 10 2021“, „SANS 25 Vulns“ a další, přičemž každá část reprezentuje vlastní kategorii zranitelností. Aplikace je vytvořena v programovacím jazyce Python za pomoci značkovacího jazyka HTML, kaskádových stylů CSS a JavaScriptu. Aplikaci lze spustit pomocí nástroje Docker popřípadě pomocí zdrojového kódu, který je volně dostupný na platformě GitHub. Zranitelnosti z pro tuto práci nestěžejnější kategorie „OWASP TOP 10 2021“ jsou shrnuty a kategorizovány v tabulce 2.5.

Tab. 2.5: Kategorizace vybraných zranitelností v Pygoat [43].

Zranitelnost	**Kategorie OWASP Top 10:2021**
Privilege Elevation #1	A01 – Broken Access Control
Privilege Elevation #2	
Privilege Elevation #3	
Privilege Elevation #4	
Privilege Elevation #5	
Forced Browsing	
Insecure Transport	A02 – Cryptograhic Failures
Broken Cryptographic Algorithm	
Use of a Key Past its Expiration Date	
SQL Injection	A03 – Injection
Command Injection #1	
Command Injection #2	
SSTI (Server-Side Template Injection)	
XSS - Reflected	
Weak Registration Process	A04 – Insecure Design
Sensitive Data Exposure via Enabled Debugg mode	A05 – Security Misconfiguration
Use of Vulnerable Component #1	A06 – Vulnerable and Outdated Components
Use of Vulnerable Component #2	
Brute Force	A07 – Identification and Authentication Failures
Overly Restrictive Account Lockout Mechanism	
Session Hijacking	
Insertion of Sensitive Information into Log File	A09 – Security Logging and Monitoring
Log Injection	
SSRF	A10 – Server-Side Request Forgery

2.6 Shrnutí

Následující podkapitola poskytuje shrnutí provedené analýzy úmyslně zranitelný webových aplikací a jejich srovnání z pohledu OWASP Top 10:2021, přičemž výsledky jsou shrnuty v tabulce 2.6. z analýzy, které bylo podrobeno celkem pět aplikací vyplynulo, že nejvíce zastoupenou kategorií OWASP Top 10:2021 v rámci testovaných aplikací je kategorie A03 – Injection, následovaná A01 – Broken Access Control a A02 – Cryptographic Failure, naopak nejméně zastoupenou kategorií je poté A10 – Server-Side Request Forgery.

Tab. 2.6: Zranitelnosti kategorizované dle OWASP Top 10:2021.

	OWASP Top 10:2021									
Název	**A01**	**A02**	**A03**	**A04**	**A05**	**A06**	**A07**	**A08**	**A09**	**A10**
DVPWA	✗	✓	✓	✗	✗	✗	✓	✗	✗	✗
DSVPW	✓	✓	✓	✓	✗	✗	✓	✓	✗	✗
Google Gruyere	✓	✗	✓	✗	✓	✗	✗	✓	✗	✗
HeadPage	✓	✓	✓	✓	✗	✗	✗	✗	✗	✗
PyGoat	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

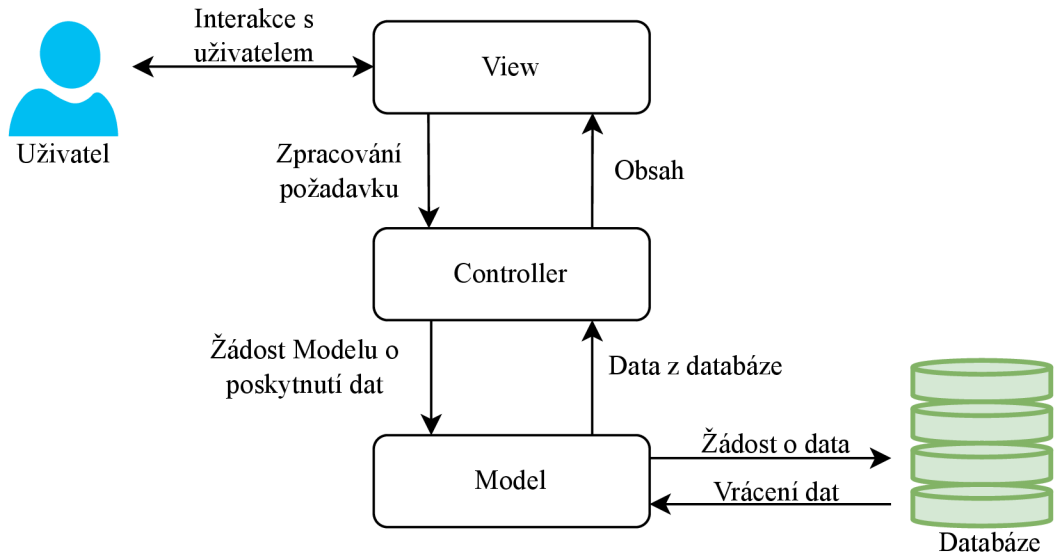
Pozn.: ✓ – implementováno, ✗ – neimplementováno.
A01 – Broken Access Control, **A02** – Cryptograhic Failures, **A03** – Injection, **A04** – Insecure Design, **A05** – Security Misconfiguration, **A06** – Vulnerable and Outdated Components, **A07** – Identification and Authentication Failures, **A08** – Software and Data Integrity Failures, **A09** – Security Logging and Monitoring, **A10** – Server-Side Request Forgery.

3 Návrh vlastní úmyslně zranitelné webové aplikace

Jak již bylo zmíněno v úvodní částí, cílem této práce bylo vytvořit úmyslně zranitelné prostředí ve formě webové aplikace. Vytvořená webová aplikace nese název Coffee Shop a představuje internetový obchod s kávou, čaji a souvisejícím příslušenstvím. Aplikace zahrnuje všechny kategorie zranitelností z OWASP Top 10:2021 a mimo výukových účelů je skrze aplikaci možné měřit a porovnávat automatizované nástroje pro penetrační testování stejně tak nástroje pro statickou bezpečnostní analýzu zdrojového kódu. Backendová část aplikace je vytvořena v programovacím jazyce Python pomocí webové frameworku Flask, frontendová část poté pomocí frameworku Bootstrap. Pro ukládání dat slouží databázový systém PostgreSQL a jako webový server je použit Apache. Aplikaci je možné spustit pomocí nástroje Docker.

3.1 Backendová část aplikace

Backendová část je vytvořena v programovacím jazyce Python s pomocí webové frameworku Flask s využitím architektury MVC (Model-View-Controller), která rozděluje aplikaci do tří logických celků jak ilustruje obrázek 3.1



Obr. 3.1: Architektura MVC [44].

Následující části stručně shrnují roli a úlohu každé části v rámci MVC.

View

Volně přeloženo jako *Pohled*, nebo také jako *Obraz* je typicky vytvořen pomocí HTML, CSS JavaScriptu a stará o prezentaci dat předaných z Modelu uživateli a představuje tak internace umožňující interakci [45, 46].

Controller

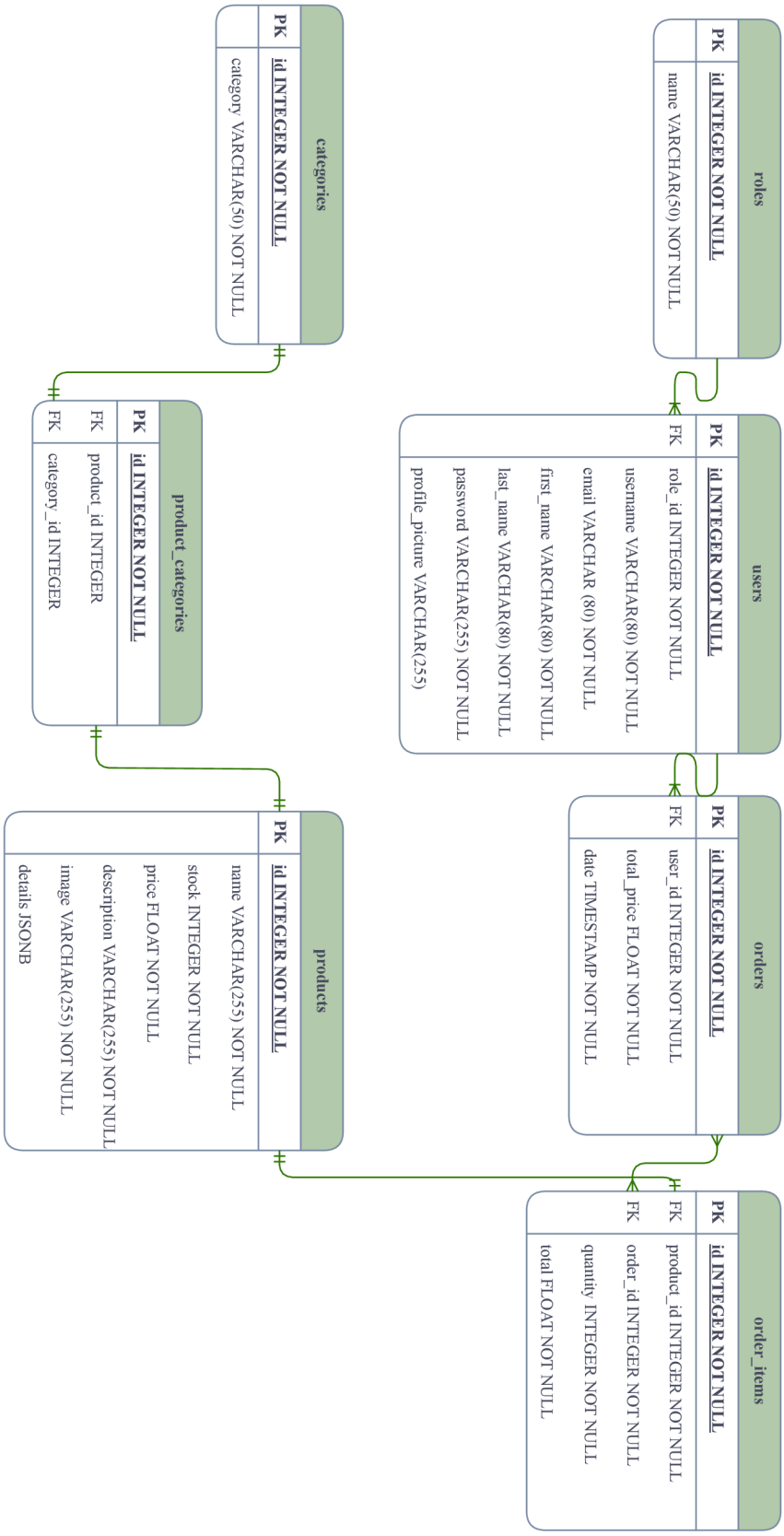
Controller, nebo také *Řadič* předává uživatelská data z View, spolupracuje s Modelem a funguje tak jako prostředník mezi nimi. Controller je zodpovědný zpracování uživatelského vstupu a také za zpracování dat před jejich zobrazením uživateli [45, 46].

Model

Model představuje data samotné aplikace, stará se o ukládání, vyhledávání a ověřování dat a celkovou interakci s databází čímž zajišťuje konzistenci a integritu [45, 46].

3.2 Struktura databáze

Strukturu databáze demonstruje ERD (Entity-Relationship Diagram) na obrázku 3.2. Uživatel může v aplikaci disponovat jednou ze dvou rolí a to *customer*, která je vyčleněná pro standardního uživatele a která neumožňuje přístup k administrativním částem aplikace stejně tak k částem které jsou stále ve vývoji, role *admin* umožňuje provádět administrativní operace mezi, které se řadí například přidání nového uživatele, modifikace dat uživatele a v neposlední řadě též odebrání uživatele. O uživateli jsou v databázi uchovávány standardní informace jako emailová adresa, uživatelské jméno, jméno, příjmení a hash hesla a profilový obrázek. Databáze rovněž uchovává záznamy objednávek jednotlivých uživatelů stejně jako informace ohledně jednotlivých produktů. Databáze aplikace je realizována pomocí databázového systému PostgreSQL.



Obr. 3.2: Entity-Relationship Diagram.

3.3 Zranitelnosti v rámci aplikace

Aplikace v současné verzi implementuje 31 zranitelností napříč všemi kategoriemi OWASP Top 10:2021 společně s jejich opravami. Jednotlivé zranitelnosti jsou zdokumentovány a kategorizovány dle OWASP Top 10:2021 v tabulce 3.1.

Tab. 3.1: Zranitelnosti v Coffee Shop kategorizované dle OWASP Top 10:2021.

Zranitelnost	**Kategorie OWASP Top 10:2021**
CSRF	A01 – Broken Access Control
Forced Browsing	
Complete Omission of Hash Function	A02 – Cryptographic Failures
Weak Hash Function	
Weak Hash Function with Salt	
Hard Coded Key	
Unprotected Transport of Credentials	
Hard Coded Credentials	
Stored XSS	A03 – Injection
Reflected XSS	
SQL Injection	
SQL Injection2	
Malicious File Upload	
OS Command Injection	
Clickjacking	A04 – Insecure Design
Sensitive Information Disclosure	
Weak Password Requirements	
Sensitive data within Cookie	
Custom Error Pages	A05 – Security Misconfiguration
Debug Mode ON	
Path Traversal	
Directory Listing	
Sensitive Cookie with Improper SameSite Attribute	
Vulnerable PostgreSQL	A06 – Vulnerable and Outdated Components
Brute Force	A07 – Identification and Authentication Failures
IDOR	
Insufficient Session Invalidation	

Functionality from Untrusted Source	A08 – Software and Data Integrity Failures
Insertion of Sensitive Information Into Log File	A09 – Security Logging and Monitoring
Insufficient Logging	
SSRF	Server-Side Request Forgery (SSRF)

3.4 Konfigurace a spuštění aplikace

Konfigurace aplikace, kterou je myšleno *vypnutí* popřípadě *zapnutí* konkrétních zranitelností či opravy v rámci konkrétní instance aplikace je možné skrze soubor *vulnerabilities.yaml* jehož obsah je vidět ve výpisu 3.1. Jak lze vidět z výpisu, struktura souboru se skládá z klíče *vulnerabilities* a korespondujících pod klíčů *SQLInjection*, *SQLInjection2* atd., reprezentujících jednotlivé zranitelnosti. Každá zranitelnost je dále asociovaná s klíčem *enable*, který může nabývat hodnot *True*, *False* nebo *null*, podle toho zda si uživatel přeje konkrétní zranitelnost či opravou, zapnout, vypnout popřípadě zcela vynechat. Poslední ze zmíněných možností slouží pro řešení kolizí kdy více zranitelností popřípadě jejich oprav upravuje stejné místo ve zdrojovém kódu a je tedy vhodné některé zranitelnosti zcela vynechat pokud jiné jsou aktivní [1]. Dalším klíčem, který je s každou zranitelností asociován je „description“, který obsahuje referenci na konkretní CWE.

Výpis 3.1: Konfigurace zranitelností.

```
1  vulnerabilities:
2    SQLInjection:
3      enabled: False
4      description: "CWE-89: SQL Injecttion -> 
           https://cwe.mitre.org/data/definitions/89.html"
5    SQLInjection2:
6      enabled: False
7      description: "CWE-89: SQL Injecttion -> 
           https://cwe.mitre.org/data/definitions/89.html"
8    SensitiveInformationDisclosure:
9      enabled: False
10     description: "CWE-209: Generation of Error Message Containing 
           Sensitive Information -> 
           https://cwe.mitre.org/data/definitions/209.html"
```

[1] Soubor *vulnerabilities.yaml* obsahuje poznámky, které pomáhají uživateli v orientaci, jak jednotlivé zranitelnosti nastavit tak aby se předešlo kolizím.

Soubor *vulnerabilities.yaml* s nakonfigurovanými zranitelnostmi je následně zpracován souborem *code_generation.py*, který na základě hodnoty konkrétního útoku vygeneruje zranitelný kód v případě, že hodnota útoku je nastavena na *True*, nebo opravu dané zranitelnosti v případě, že je hodnota *False*. Kódy zranitelností a k nim náležící opravy jsou uloženy ve složkách *vulns* a *fixes* v kořenovém adresáři projektu. Soubor *code_generation.py* slouží taktéž k mazání zranitelností či oprav v jednotlivých souborech aplikace a vrací tak projekt do původního stavu ve kterém části kódu související s konkrétní zranitelností či opravou nejsou vůbec přítomny a generují se až na základě hodnoty útoku ve *vulnerabilites.yaml*. Projekt je se všemi částmi konteinerizován a lze jej tak spustit pomocí nástroje Docker.

4 Zranitelnosti v aplikaci Coffee Shop

V této kapitole jsou podrobněji rozebrány jednotlivé zranitelnosti obsažené v aplikaci Coffee Shop. V úvodu kapitoly jsou v tabulce 4.1 zranitelnosti obsažené v aplikaci shrnuty a asociovány s názvem dle organizace MITRE a následně s příslušným kódem CWE. Následující podsekce se poté věnují popisu jednotlivých zranitelnosti a způsobu jejich implementace v aplikaci Coffee Shop.

Tab. 4.1: Zranitelnosti v Coffee Shop.

CWE	**Název zranitelnosti dle MITRE**	**Název zranitelnosti**
CWE-352	Cross-Site Request Forgery (CSRF)	CSRF
CWE-425	Direct Request (Forced Browsing)	Forced Browsing
CWE-327	Use of a Broken or Risky Cryptographic Algorithm	Complete Omission of Hash Function
		Weak Hash Function
		Weak Hash Function with Salt
CWE-321	Use of Hard-coded Cryptographic Key	Hard Coded Key
CWE-523	Unprotected Transport of Credentials	Unprotected Transport of Credentials
CWE-798	Use of Hard-coded Credentials	Hard Coded Credentials
CWE-79	Improper Neutralization of Input During Web Page Generation (Cross-site Scripting)	Stored XSS
		Reflected XSS
CWE-89	Improper Neutralization of Special Elements used in an SQL Command (SQL Injection)	SQL Injection
		SQL Injection2
CWE-434	Unrestricted Upload of File with Dangerous Type	Malicious File Upload
CWE-78	Improper Neutralization of Special Elements used in an OS Command (OS Command Injection)	OS Command Injection
CWE-1021	Improper Restriction of Rendered UI Layers or Frames	Clickjacking
CWE-200	Exposure of Sensitive Information to an Unauthorized Actor	Sensitive Information Disclosure
CWE-521	Weak Password Requirements	Weak Password Requirements
CWE-315	Cleartext Storage of Sensitive Information in a Cookie	Sensitive Data within Cookie
CWE-756	Missing Custom Error Page	Custom Error Pages
CWE-489	Active Debug Code	Debug Mode ON
CWE-35	Path Traversal: '.../...//'	Path Traversal
CWE-548	Exposure of Information Through Directory Listing	Directory Listing

CWE-1275	Sensitive Cookie with Improper SameSite Attribute	Sensitive Cookie with Improper SameSite Attribute
CWE-1035	Using Components with Known Vulnerabilities	Vulnerable PostgreSQL
CWE-307	Improper Restriction of Excessive Authentication Attempts	Brute Force
CWE-639	Authorization Bypass Through User-Controlled Key	IDOR
CWE-613	Insufficient Session Expiration	Insufficient Session Invalidation
CWE-830	Inclusion of Web Functionality from an Untrusted Source	Functionality from Untrusted Source
CWE-532	Insertion of Sensitive Information Into Log File	Insertion of Sensitive Information Into Log File
CWE-778	Insufficient Logging	Insufficient Logging
CWE-918	Server-Side Request Forgery (SSRF)	SSRF

CWE-352: Cross-Site Request Forgery (CSRF)

CSRF je zranitelnost, skrze kterou může útočník provést akci na straně uživatele, kterou sám uživatel neměl v úmyslu provést. Pro to aby mohl být CSRF útok úspěšně proveden musí být splněny následující podmínky

- **Relevantní akce** - v rámci aplikace existuje funkce, kterou má útočník zájem provést ze strany uživatele.
- **Relace založená na identifikátorech** - druhou podmínkou je, že aplikace spoléhá výhradně identifikátory relace v případě správy relace a ověřování HTTP požadavků.
- **Žádné neočekávané parametry v HTTP požadavku** - HTTP požadavek, kterým útočník provede požadovanou akci neobsahuje žádné parametry, které by útočník nemohl určit, popřípadě snadno odhadnout.

V aplikaci Coffee Shop je tato zranitelnost realizována skrze funkcionalitu pro smazání uživatelského účtu. Zneužití zranitelnosti může vypadat tak, že útočník vytvoří falešnou webovou stránku, přičemž odkaz na ni doručí uživateli například pomocí emailu, popřípadě skrze libovolnou sociální síť. Obsahem falešné webové stránky bude URL (Uniform Resource Locator) adresa, skrze, kterou dojde k provedení smazání uživatelského účtu v případě, že má uživatel s aplikací aktivní relaci, tj. je přihlášen. Adresa může být umístěna například v HTML elementu *img*, který má výhodu v tom, že jakmile uživatel výše zmíněnou falešnou stránku otevře, prohlížeč provede HTTP požadavek na adresu v elementu *img* automaticky bez potřeby další akce od uživatele. Oprava zranitelnosti je poté realizována implementací CSRF tokenu na všech formulářích. CSRF token představuje právě oný neočekávaný parametr zmíněný výše. [47, 48].

CWE-425: Forced Browsing

Forced Browsing je zranitelností, při které nedochází k patřičné kontrole oprávnění ze strany aplikace při přístupu k chráněným zdrojům aplikace. Mezi takové zdroje mohou patřit například administrační panely, které umožňují přístup k citlivým datům, lze skrze ně provádět senzitivní operace jako například, přidání, odebrání popřípadě změna údajů uživatele [49, 50]. V aplikaci Coffee Shop je tato zranitelnost realizována formou administračního panelu u kterého však neprobíhá žádná kontrola z pohledu autorizace, tedy jestli uživatel je či není oprávněn k přístupu a administrační panel je tím pádem dostupný komukoli. Této skutečnosti může využít útočník a skrze panel provádět změny. Následná oprava této zranitelnosti je v aplikaci realizována, ověřením uživatelovi role před přístupem k panelu, přičemž role je získána pomocí objektu *current_user*, který, umožňuje přístup ke všem parametrům uživatele [51].

CWE-327: Use of a Broken or Risky Cryptographic Algorithm

Tato zranitelnost nastává v případě, že aplikace používá prolomený, nebo jiným způsobem nevhodný kryptografický algoritmus. V aplikace Coffee Shop je tato zranitelnost realizována v rámci *Weak Hash Function* kde dochází k implementaci již prolomeného hašovacího algoritmu MD5, dále *Weak Hash Function with Salt*, kde je stejně jako v předchozím případě použit již prolomený hašovací algoritmus MD5, tentokrát je však na vstupu hašovací funkce použita kryptografická sůl a *Complete Omission of Hash Function* kde je hašovací algoritmus zcela vynechám. Hašovací algoritmus je matematická funkce jež produkuje výstup pevné délky na základě vstupu libovolné délky. Pro zvýšení bezpečnosti lze na vstup hašovacího algoritmu přidat kryptografickou sůl, což je náhodně vygenerovaná hodnota, která je přidaná ke vstupu hašovacího algoritmu. Sůl přidává do procesu hašování prvek náhodnosti a zvyšuje ochranu proti slovníkovým útokům stejně jako útokům realizovaných pomocí duhových tabulek. Hašovací funkce se používají například ke kontrole integrity, ověřování digitálních podpisů a také k ověřování hesel [52, 53, 54, 55].

Oprava této zranitelnost je realizována použitím hašovacího algoritmu Argon2, konkrétně poté jeho nejbezpečnější varianty Argon2id [56].

CWE-321: Use of Hard-coded Cryptographic Key

Nastává v případě, že je šifrovací klíč zakomponován přímo v kódu aplikace. V aplikaci Coffee Shop je tato zranitelnost realizovaná tak, že se šifrovací klíč s názvem *SECRET_KEY* nachází přímo ve zdrojovém kódu, konkrétně poté v souboru *config.py* místo toho, aby byl načítán z *.env* souboru. Soubory *.env* jsou používány za účelem bezpečného uložení údajů a jiných citlivých informací. *SECRET_KEY*

je v aplikacích založených na frameworku Flask používán k podepsání identifikátoru relace [57, 58, 59, 60]. Tato zranitelnost je v aplikaci využívána společně s „CWE-315: Cleartext Storage of Sensitive Information in a Cookie“ kde je v identifikátoru relace přenášená i role, kterou uživatel v aplikaci disponuje a která taktéž slouží ke kontrole přístupu do administračního panelu aplikace. V případě, že útočník disponuje klíčem *SECRET_KEY* může s jeho pomocí změnit svoji roli na *admin*[2], modifikovaný identifikátor relace pomocí klíče podepsat a úspěšně tak získat přístup k administračnímu panelu. Oprava je realizována tak, že *SECRET_KEY*, je načítán z *.env* souboru místo toho, aby byl přímo součástí zdrojového kódu.

CWE-523: Unprotected Transport of Credentials

K této zranitelnosti dochází v případě, že aplikace nepoužívá dostatečné zabezpečení při přenosu citlivých informací (přihlašovací jméno, heslo, . . .) což může vést situaci kdy útočník přenášená na trase odposlechne. V aplikaci Coffee Shop je tato zranitelnost realizována vynecháním implementace protokolu TLS, který zabezpečuje přenos dat mezi klientem, což může být například webový prohlížeč a serverem. Protokol TLS zajišťuje důvěrnost, autentičnost a integritu přenášených dat pomocí TLS certifikátů. TLS certifikáty jsou vydávány certifikačními autoritami vlastníkovi domény. Certifikát obsahuje informace vztahující se k doméně ke které byl vydán, především však obsahuje veřejný klíč serveru, který je stěžejní pro ověření jeho identity [61, 62]. Oprava této zranitelnosti je v aplikaci realizována pomocí implementace TLS.

CWE-798: Use of Hard-coded Credentials

Tato zranitelnost nastává v případě, kdy jsou údaje jako heslo, uživatelské jména a podobně, které jsou používány k autentizaci, přímo součástí zdrojového kódu místo *.env* souboru. V případě, že je útočník schopen získat přístup ke zdrojovému kódu, například je-li zdrojový kód veřejně dostupný může údaje v něm obsažené použít a získat tak přístup k citlivým datům [63, 64]. V aplikaci Coffee Shop je tato zranitelnost realizována v souboru *config.py* kde proměnná *SQLALCHEMY_DATABASE_URI* obsahuje údaje k databázovému systému PostgreSQL. Podobně jako v případě zranitelnosti *CWE-321: Use of Hard-coded Cryptographic Key* je i tato oprava realizována načítáním hodnoty proměnné *SQLALCHEMY_DATABASE_URI* z *.env* souboru.

[2]Zde je nutné mít aktivní zranitelnost *CWE-315: Cleartext Storage of Sensitive Information in a Cookie*, aby došlo k vložení uživatelovi role do identifikátoru relace a probíhala na jeho základě kontrola přístupu k administračnímu panelu.

CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')

Zranitelnost, ke které dochází vlivem nesprávné popřípadě nedostatečné neutralizace externě dodané škodlivé hodnoty ve formě JavaScriptového kódu, která se v dalších krocích stává součástí výstupu aplikace. Injekce škodlivého skriptu do aplikace může proběhnout například během HTTP dotazu, kdy následně aplikace tuto hodnotu v nějaké formě zakomponuje do příslušně odpovědi přičemž při následné generaci obsahu dojde k provedení toho kódu. Mezi následky exploitace lze zmínit například krádež webové cookie příslušné relace mezi prohlížečem na straně klienta a serverem. Existují 3 hlavní typy [65, 66, 67, 68]:

- **Reflected XSS** – o tento typ se jedná v případě, že škodlivý skript dodaný útočníkem je okamžitě vrácen tak zvaně „reflektován" v následné odpovědi například ve formě chybové zprávy.
- **Stored XSS** – v tomto případě dochází k injekci škodlivého skriptu a následném uložení tohoto skriptu na serveru na, kterém aplikace běží.
- **DOM-based XSS** – tento typ vyplývá na rozdíl od dvou předešle zmíněných ze zranitelnosti na straně klient nežli serveru. Může se jednat například o skript běžící na straně klienta zpracovávající data od nedůvěryhodného zdroje,

V aplikace Coffee Shop jsou realizovány první dva typy XSS, přičemž *Reflected XSS* je realizován v rámci chybové zprávy na přihlašovací stránce kdy výsledkem zadání nesprávného přihlašovacího jména je chybová zpráva, která zadané jméno obsahuje. Tato hodnota však není ze strany aplikace nijak ošetřena. *Stored XSS* je poté realizován v rámci funkcionalit pro vytvoření a aktualizaci uživatelského účtu, kde však nedochází k žádné validaci zadávaných hodnot, mezi které patří emailová adresa, uživatelské jméno a podobně. Oprava *Reflected XSS* je realizována pomocí šablonovacího enginu Jinja, který je součástí framewroku Flask a který poskytuje ochranu proti *Reflected XSS* pomocí takzvaného „escapování", což je proces při, kterém jsou potenciálně škodlivé znaky nahrazeny jejich bezpečnými ekvivalenty [69, 70]. Oprava *Stored XSS* je poté realizována pomocí funkcí `input_validation` a `email_validation`, které pracují s množinou povolených hodnot vůči, které jsou data ve formulářích porovnávána a následně příjmuta popřípadě zamítnuta.

CWE-89: SQL Injection

SQL Injection je zranitelnost, vznikající v případě, kdy aplikace zkonstruuje část popřípadě celý SQL dotaz za pomocí externě dodané vstupní hodnoty, přičemž však nedochází k neutralizaci speciálních elementů, které tento vstup může obsahovat a které mohou potenciálně ovlivnit původní SQL dotaz. Mezi běžné následky zneužití této zranitelnosti patří odcizení citlivých dat z databáze, elevace privilegií,

vložení nových dat popřípadě smazání či modifikace stávajících dat [71, 72]. V aplikaci Coffee Shop je tato zranitelnost realizována dvakrát a to v rámci autentizačního mechanizmu *SQLInjection* a dále na stránce libovolného produktu úpravou URL *SQLInjection2*. Oprava obou výše zmíněných zranitelností je poté realizována použitím parametrizovaných dotazů, které útočníkovi neumožní manipulovat s původním SQL dotazem [73].

CWE-434: Unrestricted Upload of File with Dangerous Type

Zranitelnost ke které dochází v případě, že aplikace umožňuje uživateli nahrávat soubory avšak bez toho aniž by docházelo ke kontrole typu, velikosti, jména a dalších prvků souboru, nebo tato kontrola není dostatečná. Zneužití této zranitelnosti může být realizováno například získáním citlivých dat ze serveru skrze nahrání škodlivého skriptu, infikací cílového serveru pomocí malwaru [74, 75]. V aplikaci Coffee Shop je tato zranitelnost realizovaná v rámci správy účtu kde je uživateli umožněno nahrát profilový obrázek, přičemž však nejsou implementovaná, žádná omezení co se typu a velikosti souboru týče. Oprava této zranitelnosti je v aplikaci realizována formou omezení a kontroly přípony nahrávaných souborů, následně kontroly MIME (Multipurpose Internet Mail Extensions) [76] což je hodnota, která vypovídá o formátu a povaze souboru a dále je ze strany aplikace omezena maximální velikost souboru, kterou může uživatel nahrát.

CWE-78: OS Command Injection

Zranitelnost OS Command Injection nastává v případě, že aplikace zahrne externě dodaný vstup do systémového příkazu, který je následně proveden na serveru na kterém aplikace běží. Skrze zneužití této zranitelnosti může útočník elevovat svá oprávnění v rámci cílového systému a potenciálně číst, vytvářet, mazat popřípadě jinak modifikovat data v cílovém systému [77, 78]. V aplikaci Coffee Shop se zranitelnost OS Command Injection nachází v rámci funkcionality administračního panelu, která umožňuje zkontrolovat stav služeb PostgreSQL a Apache2. Oprava této zranitelnosti je realizována tak, že nedochází k volání konkrétního příkazu, ale numerické hodnoty, která je následně ne straně serveru na příkaz mapována čímž je zabráněno provádění libovolného příkazu.

CWE-1021: Improper Restriction of Rendered UI Layers or Frames

Jedná se o tak zvaný *interface-based* útok založený na nedostatečném, nebo zcela chybějícím omezení načítání webových stránek aplikace z HTML rámu `<iframe>`. Tento útok je realizován pomocí vytvoření falešné webové stránky (*návnady*, *decoy*

website) do které útočník pomocí výše zmíněného HTML rámu zakomponuje skrytou webovou stránku (*hidden website*), kterou pomocí nastavení HTML parametrů skryje. Útočník může odkaz na takto vytvořenou falešnou webovou stránku šířit pomocí emailu, popřípadě skrze sociální sítě. Cílem útoku je oklamat uživatele ke kliknutí na určitý prvek falešné webové stránky, přičemž zároveň dojde ke kliknutí na jiný prvek na skryté webové stránce. Falešná webová stránka může uživatele informovat o výhře, odměně, limitované nabídce a podobně a vyzývat uživatele aby kliknutím na odkaz, tlačítko a podobně danou odměnu aktivoval. Skrytou webovou stránkou může být například webová stránka uživatelova oblíbeného internetového obchodu napozicovaná tak, aby kliknutí na falešné webové stránce vyvolalo kliknutí na tlačítko „Delete“ na skryté webové stránce čímž dojde ke smazání uživatelova účtu. Výše zmíněný scénář popisuje způsob jakým je tato zranitelnost realizována v aplikaci Coffee Shop [79, 80]. Oprava této zranitelnosti je realizována pomocí nastavení HTTP hlavičky *X-Frame-Options* na *DENY*, což zabraňuje načítání stránek webové aplikace skrze `<frame>` či `<iframe>` [81].

CWE-200: Exposure of Sensitive Information to an Unauthorized Actor

Jedná se o zranitelnost při, které dochází ze strany aplikace ke generování chybových zpráv obsahujících citlivé informace, popřípadě informace, které mohou útočníkovi posloužit v dalších fázích jeho útoku. V rámci aplikace Coffee Shop je tato zranitelnost realizována na přihlašovací stránce, kdy je v případě zadání existujícího uživatelského jména, ale špatného hesla vrácena zpráv „Incorrect password.“, tedy, že heslo je nesprávné, což útočníkovi dává informaci, že uživatel v systému existuje a může se tak například v případě útoku hrubou silou soustředit pouze na prolomení hesla [82]. Oprava této zranitelnosti je realizována generováním obecnější zprávy „Incorrect credentials, try again.“.

CWE-521: Weak Password Requirements

Ke zranitelnosti Weak Password Requirements dochází v případě, že aplikace nevynucuje po uživatelých výběr silného hesla což útočníkovi usnadňuje kompromitaci účtů. Útočník může tuto zranitelnost využít například provedením slovníkového útoku, při kterém dochází ke zkoušení běžných slov a frází z předem definovaného slovníku, popřípadě může použít útok hrubou silou při kterém dochází ke zkoušení všech možných kombinací písmen, slov a číslic [83, 84, 85]. V aplikaci Coffee Shop je tato zranitelnost realizovaná tak, že během vytvoření účtu ani během změny hesla není uživatel nikterak limitován a na heslo nejsou kladeny žádné požadavky z hlediska délky ani komplexnosti. Oprava této zranitelnosti je realizována

pomocí funkce `check_for_password_complexity`, která kontroluje délku a komplexnost hesla a zajišťuje, že uživatelem zvolené heslo je alespoň osm znaků dlouhé a obsahuje jak velká, tak malá písmena, stejně jako čísla a speciální znaky.

CWE-315: Cleartext Storage of Sensitive Information in a Cookie

Tato zranitelnost nastává v případě, že ze strany aplikace dochází k ukládání citlivých, nebo z pohledu útočníka zajímavých informací do identifikátoru relace v otevřené podobě. Této situace může následně útočník zneužít a za pomocí nástrojů jako například Flask Unsign obsah identifikátoru zobrazit, popřípadě i upravit. V případě aplikace Coffee Shop je tato zranitelnost realizována tak, že role, kterou uživatel v systému disponuje je uložena v rámci identifikátoru relace přičemž aplikace provádí kontrolu přístupu výhradně na základě této hodnoty. Útočník může použít výše zmíněný nástroj Flask Unsign a zobrazit si obsah identifikátoru relace. V případě, že útočník disponuje znalostí *SECRET_KEY*, který je frameworkem Flask používán pro podpis identifikátoru relace, může přepsat hodnotu role z *customer* na *admin*, takto modifikovaný identifikátor následně pomocí výše zmíněného nástroje podepsat a provést tak elevaci privilegií [86, 87, 88, 89, 57]. Oprava této zranitelnosti je realizována tak, že kontrola přístupu je aplikací prováděna nikoli na základě hodnoty role uložené v identifikátoru relace, ale na základě hodnoty, která je uložená v databázovém systému.

CWE-756: Missing Custom Error Page

Zranitelnost Missing Custom Error Page nastává v případě, že aplikace používá výchozí chybové stránky skrze, které může docházet k odhalení citlivých informací [90]. V aplikaci Coffee Shop je tato zranitelnost realizována skrze ponechání výchozích chybový stránek webového serveru Apache2, které obsahují mimo jiné informace o verzi webového serveru Apache2. Oprava této zranitelnosti je poté realizována pomocí vytvoření a implementace vlastních chybových stránek, které citlivé informace na rozdíl od výchozích nezahrnují.

CWE-489: Active Debug Code

K této zranitelnosti dochází v případě, že aplikace je nasazená v produkčním režimu s aktivní debugovací funkcionalitu, přičemž tato funkcionalita odhaluje citlivé informace popřípadě umožňuje provádět operace, které může útočník zneužít [91]. V aplikaci Coffee Shop je tato zranitelnost realizovaná skrze povolený *Werkzeug Debugger* a vypnutý *Werkzeug Console PIN*, který umožňuje neomezený přístup ke konzoli na cestě */console* popřípadě vyvoláním výjimky a následným provedení RCE.

Werkzeug je soubor knihoven využívaných k tvorbě WSGI (Web Server Gateway Interface) v programovacím jazyce Python [92, 93, 94]. Oprava této zranitelnosti je realizována vypnutím *Werkzeug Debugger* a současné implementaci vlastních chybových stránek.

CWE-35: Path Traversal: '.../...//'

Path Traversal je zranitelností, která nastává v případě, kdy aplikace k vytvoření cesty k požadovaným datům využívá externě dodanou hodnotu u které však nedochází k neutralizaci sekvence „../“, která slouží k posunu v souborovém systému o úroveň výše a umožňuje tak útočníkovi číst soubory i mimo původně zamýšlený adresář. Zneužití této zranitelnosti nemusí útočník nutně získat právo ke čtení všech souborů v rámci cílového systému, záleží na oprávnění uživatele za kterého aplikace na cílovém systému běží [95, 96]. V aplikaci Coffee Shop se zranitelnost Path Traversal nachází na */tips_and_tricks/guide?file_name=guide1.txt* kde lze modifikací parametru *file_name* číst libovolný soubor. Oprava této zranitelnosti je realizována v první fázi tak, že je provedena kontrola, zda název uživatelem žádaného souboru je ve správném formátu. V případě, že ano, následuje kontrola systémových cest, tedy zda je soubor skutečně načítán ze zamýšleného adresáře a nedochází k pokusu o manipulaci a čtení z jiného adresáře a v případě, že i výsledek této kontroly je v pořádku a požadovaný soubor je nalezen, je uživateli zobrazen. V případě selhání libovolné kontroly, nebo v případě, že soubor není nalezen je generována chyba.

CWE-548: Exposure of Information Through Directory Listing

Jedná se o zranitelnost, kdy jsou skrze povolený výpis adresáře odhaleny všechny soubory uvnitř adresáře ve kterém se aplikace na serveru nachází což vede k odhalení citlivých informací [97]. V aplikaci Coffee Shop je tato zranitelnost realizována v rámci konfiguračního souboru webového serveru Apache2 *coffee-shop.conf*. Samotný výpis adresáře se poté nachází na cestě */listing*. Oprava této zranitelnosti je realizována zakázáním výpisu adresáře v již výše uvedeném konfiguračním souboru webového serveru Apache2.

CWE-1275: Sensitive Cookie with Improper SameSite Attribute

Atribut *SameSite* specifikuje zda dojde k přiložení identifikátoru relace k požadavkům přicházejícím z jiné webové stránky. S atributem *SameSite* se dále pojí dva typy požadavků

- **same-site** – o same-site požadavek se jedná v případě, že se shoduje *schéma (http, https, . . .) TLD (Top-Level-Domain) (.com, .net, .edu, . . .)* a doménové jméno zdroje a cíle.

- **cross-site** – požadavkem u kterého se liší buď schéma, TLD či doménové jméno je považován za cross-site.

SameSite atribut může nabývat tří hodnot, které se liší v tom zda prohlížeč identifikátor relace v případě cross-site požadavku přiloží či nikoli.

- **Strict** – prohlížeč zašle identifikátor relace pouze v případě, že se jedná o same-site požadavek.
- **Lax** – prohlížeč zašle identifikátor relace i v případě cross-site požadavku, avšak pouze v případě, že HTTP metoda požadavku je GET a požadavek vznikl z navigace na nejvyšší úrovni (například kliknutí na odkaz). Cookie však nebude zaslána v případě, že je požadavek vyvolán skriptem, skrze `<iframe>` a podobně.
- **None** – Tato hodnota vypíná veškerá omezení atributu *SameSite* a prohlížeč tak zasílá identifikátor relace kdykoliv kdy je vyžádána.

V aplikaci Coffee Shop je tato zranitelnost realizována nastavením hodnoty atributu *SameSite* na „None“, čímž jak již bylo zmíněno výše, dojde k vypnutí veškerých omezení. Skrze tuto zranitelnost je dále možné zneužít další zranitelnosti, které jsou v aplikaci přítomné, jmenovitě *CSRF* a *Clickjacking*. Oprava této zranitelnosti je realizována nastavením hodnoty *SameSite* atributu na „Strict“ což má za následek jak již byla zmíněno výše, že prohlížeč bude zasílat identifikátor relace pouze v případě same-site požadavku [98, 99, 100].

CWE-1035 2017 Top 10 A9: Using Components with Known Vulnerabilities

Zranitelnost, která nastává v případě, že aplikace používá komponenty třetích stran jako například knihovny, frameworky a podobně, která obsahují známé zranitelnosti a/nebo už nejsou vydavatelem podporovány. Útočník může této skutečnosti využít, provést zneužití zranitelnosti, popřípadě zranitelností a získat tak přístup k citlivým datům, popřípadě provést jiné škodlivé operace [101, 102]. V aplikaci Coffee Shop je tato zranitelnost realizována použitím zranitelné verze databázového systému PostgreSQL ve verzi 9.6.1, která je zranitelná vůči CVE-2019–9193 [103, 104]. Oprava této zranitelnosti je realizována použitím poslední verze databázového systému PostgreSQL.

CWE-307: Improper Restriction of Excessive Authentication Attempts

K této zranitelnosti dochází v případě, že aplikace neimplementuje opatření zamezující mnohonásobným neúspěšným pokusům o přihlášení, což umožňuje útočníkovy provést slovníkový útok popřípadě útok hrubou silou. Jednou z možných obran a zároveň tou, která použita v aplikaci Coffee Shop je implementace reCAPTCHA, což

je služba, jejímž cílem je mimo jiné chránit webové aplikace před automatizovanými útoky [105, 106, 85].

CWE-639: Authorization Bypass Through User-Controlled Key

K této zranitelnosti dochází v případě, že aplikace používá externě dodaný vstup k přímému přístupu k objektům, například uživatelským účtům v rámci aplikace. Útočník může tuto zranitelnosti zneužít a modifikací hodnoty, která slouží k identifikaci objektu, může získat přístup k datům libovolného uživatele [107, 108]. V aplikaci Coffee Shop je tato zranitelnost realizována v rámci karty *Account* kde útočník může modifikací URL konkrétně poté parametru *id* přistupovat k účtům ostatních uživatelů. Oprava této zranitelnosti je v aplikaci realizována tak, že parametr *id* je zcela vypuštěn a uživatel je identifikován na základě objektu *current_user*, který je součástí knihovny Flask-Login a který slouží k reprezentaci uživatele [51].

CWE-613: Insufficient Session Expiration

Tato zranitelnost nastává v případě, že nedochází k invalidaci relace ze strany aplikace což dává útočníkovi možnost tyto relace skrze odcizené identifikátory relace opětovně používat a tím úspěšně obcházet autentizační proces. V případě aplikace Coffee Shop je tato zranitelnost realizována tak, že odhlášení uživatele z aplikace nevede k invalidaci dané relace, ale pouze k přesměrování na přihlašovací stránku, výsledkem čehož však relace zůstává aktivní a v případě, že útočník získá identifikátor relace, může převzít kontrolu nad uživatelským účtem s nímž je identifikátor asociován [109]. Oprava této zranitelnosti je v aplikaci realizována pomocí funkce knihovny Flask-Login s názvem *logout_user*, která je zodpovědná za odhlášení uživatele [51].

CWE-830: Inclusion of Web Functionality from an Untrusted Source

K této zranitelnosti dochází v případě, že je v rámci aplikace použita funkcionalita třetí strany z nedůvěryhodného zdroje, popřípadě z důvěryhodného avšak kompromitovaného zdroje [110]. V aplikace Coffee Shop je tato zranitelnost realizována pomocí kontejneru nástroje Docker s názvem „compromised_service“, který představuje službu ze které aplikace Coffee Shop načítá mapu, která po úspěšném vytvoření objednávky uživateli ukáže kde si ji může vyzvednout. Služba však byla útočníkem kompromitována a nyní kromě samotné mapy načítá i škodlivý skript[3]. Oprava této zranitelnosti je realizována použitím Map od společnosti Google v rámci HTML rámu.

[3]V reálném případě kdy by byla služba kompromitovaná, by bylo v zájmu útočníka použít skutečný škodlivý skript, v tomto případě byla pro demonstraci použita funkce alert().

CWE-532: Insertion of Sensitive Information into Log File

K této zranitelnosti dochází v případě, že aplikace do záznamových zpráv vkládá citlivé údaje, popřípadě údaje, které mohou útočníkovi poskytnout informace, které jsou potenciálně zneužitelné. V aplikaci Coffee Shop je tato zranitelnost realizovaná na přihlašovací stránce, kdy úspěšné přihlášení vede k vytvoření záznamu obsahujícího jméno a heslo se kterým se uživatel do aplikace přihlásil, neúspěšné přihlášení poté vygeneruje zprávu o tom, že uživatelské jméno neexistuje popřípadě, že zadané heslo je nesprávné. Tyto údaje však útočníkovi poskytují dodatečné informace kterých může využít, získá-li k souboru se záznamy přístup, především heslo, respektive hash hesla je informace, která by do záznamové zprávy nikdy neměla být zahrnuta [111]. Oprava této zranitelnosti je v aplikaci realizována úpravou záznamových zpráv na „User failed to login! Wrong credentials.“ v případě, nesprávného jména, či hesla a „User admin successfully logged in.“ v případě úspěšného přihlášení v tomto konkrétním případě uživatele *admin*.

CWE-778: Insufficient Logging

Insufficient Logging nastává v případě, že aplikace neprovádí záznam událostí které jsou považovány za významné z pohledu bezpečnosti popřípadě dochází k vynechání důležitých informací [112]. V aplikaci Coffee Shop je tato zranitelnost realizována vynecháním IP adresy a časové známky u záznamu. Oprava je poté realizována přidáním těchto údajů.

CWE-918: Server-Side Request Forgery (SSRF)

Zranitelnost Server-Side Request Forgery zkráceně SSRF nastává v případě kdy aplikace umožňuje uživateli provést HTTP požadavky jménem serveru na kterém běží, přičemž však nedochází ke kontrole těchto požadavků. Vlivem výše zmíněného může poté útočník provést například HTTP požadavky směrem k interním systémům, které jsou chráněny firewallem a nejsou tak z venčí dostupné, popřípadě získat přístup k datům uloženým na cílovém serveru a to pomocí schématu *file:///* a v neposlední řadě též provést skenování portů [113, 114, 115, 116]. V aplikaci Coffee Shop je tato zranitelnost realizována v rámci sekce */development*, která reprezentuje část webové aplikace, která se stále nachází ve vývoji a umožňuje uživateli provádět HTTP požadavky jménem aplikace, přičemž však nedochází k žádné kontrole cíle požadavku, ani použitého schématu. Skrze manipulaci hodnoty URL parametru *url* může útočník číst soubory ze serveru na kterém aplikace běží, a to pomocí URI (Uniform Resource Identifier) schématu *file:///*. Oprava této zranitelnosti je poté realizována pomocí definování povolených URI schémat a domén, porovnáním každé uživatelem zadané adresy vůči těmto hodnotám a následným provedením daného

HTTP požadavku popřípadě jeho zamítnutím v případě, že adresa nesplňuje dané požadavky. K sekci */development* je dále umožněn přístup pouze uživateli *admin* a cesta k této sekci je tedy */admin/development*.

5 Bezpečnostní testování výsledné aplikace

Následující kapitola se zabývá bezpečnostním testováním výsledné aplikace. Pro testování byly vybrány platformy GitHub, GitLab a Snyk. GitHub, stejně jako GitLab je platforma, která umožňuje hostování a zprávu kódu a mimo jiné nabízí také zabezpečení kódu ve formě skenů, auditů a podobně. Snyk je poté komplexní platformou zabývající se bezpečností nejen kódu, ale také například kontejnerů a cloudových konfigurací [117, 118, 119].

Nastavení se, kterým byla aplikace otestována je zobrazeno v tabulce 5.1. Porovnání výsledků jednotlivých platforem je poté v tabulce 5.2.

Tab. 5.1: Testovaný stav aplikace.

Zranitelnost	**Stav**
CSRF	✓
Forced Browsing	✓
Complete Omission of Hash Function	-
Weak Hash Function	✓
Weak Hash Function with Salt	-
Hard Coded Key	✓
Unprotected Transport of Credentials	✓
Hard Coded Credentials	✓
Stored XSS	✓
Reflected XSS	✓
SQL Injection	✓
SQL Injection2	✓
Malicious File Upload	✓
OS Command Injection	✓
Clickjacking	✓
Sensitive Information Disclosure	✓
Weak Password Requirements	-
Sensitive Data within Cookie	✓
Custom Error Pages	-
Debug Mode ON	✓
Path Traversal	✓
Directory Listing	✓

Sensitive Cookie with Improper SameSite Attribute	✓
Vulnerable PostgreSQL	✓
Brute Force	✓
IDOR	✓
Insufficient Session Invalidation	✓
Functionality from Untrusted Source	✓
Insertion of Sensitive Information Into Log File	✓
Insufficient Logging	✓
SSRF	✓

Pozn.: ✓ – aktivní zranitelnost, **-** – přeskočeno.

GitHub

Platforma GitHub nabízí možnost analýzy a hledání zranitelností ve zdrojovém kódu pomocí vlastního enginu CodeQL. Pro identifikaci zranitelností CodeQL používá dotazy, které jsou vytvářeny a udržovány profesionály a výzkumníky z oboru počítačové bezpečnosti, stejně tak členy komunity [120].

GitLab

Platforma GitLab umožňuje analýzu zdrojového kódu kódu pomocí SAST (Static Application Security Testing) což je metodologie používaná pro analýzu a hledání zranitelností ve zdrojovém kódu. GitLab umožňuje pomocí SAST analyzovat širokou škálu programovacích jazyků mezi, které se řadí například (C/C++, .NET, Python a další). GitLab dále umožňuje nakonfigurovat konkrétní SAST analyzátor, který má být pro analýzu použit. Mezi podporované analyzátory se řadí mimo jiné Semgrep, který byl použit pro otestování výsledné aplikace Coffee Shop [121].

Snyk

Poslední z platforem, které byly použity pro otestování výsledné aplikace byla platforma Snyk. Snyk je komplexní platformou která dokáže provést analýzu nejen samotného zdrojového kódu, ale také kontejnerů, závislostí (dependencies) a také například cloudových konfigurací. Snyk může být mimo jiné propojen s výše uvedenými platformami GitHub a GitLab a provést analýzu v nich obsažených repositářů čehož bylo využito v této práci [119].

Shrnutí

Následující tabulka poskytuje shrnutí výsledků analýzy aplikace Coffee Shop při konfiguraci zobrazené v tabulce 5.1. Z výsledků je zřejmé, že nejvíce zranitelností bylo nalezeno platformou GitHub a to sedm, následuje platforma Snyk se šesti nalezenými zranitelnostmi a na třetím místě se poté umístila platforma GitLab.

Tab. 5.2: Výsledky testování aplikace.

Platforma	**True Positive**	**True Negative**	**False Positive**	**False Negative**
GitHub	7	0	0	20
GitLab	3	0	0	24
Snyk	6	0	0	21

Pozn.: **True Positive** – Zranitelnsot se v aplikaci vyskytuje a byla platformou detekována, **True Negative** – Zranitelnost se v aplikaci nevyskytuje a nebyla platformou detekována, **False Positive** – Zranitelnsot se v aplikaci nevyskytuje a platforma zranitelnsot detekovala, **False Negative** – Zranitelnost se v aplikaci vyskytuje a nebyla platformou detekována.

Závěr

Tato bakalářská práce se zabývala návrhem a vývojem úmyslně zranitelného prostředí ve formě webové aplikace. Výsledkem této práce je webová aplikace s názvem Coffee Shop, která reprezentuje online obchod s kávou, čaji a příslušenstvím. Aplikace obsahuje celkem 31 zranitelností napříč všemi kategoriemi OWASP Top 10:2021 společně s jejich opravami a lze ji využít jednak pro porovnávání automatizovaných nástrojů pro penetrační testování, nástrojů pro statickou bezpečnostní analýzu zdrojového kódu, ale také pro výukové účely.

V první kapitole této práce byla představena komunita OWASP, která funguje pod záštitou neziskové organizace OWASP Foundation a také její pro tuto práci nejstěžejnější projekt OWASP Top 10, konkrétně poté jeho verze z roku 2021. Druhá kapitola se věnuje analýze vybraných úmyslné zranitelných aplikací, která vývoji aplikace popsané v této práci předcházela. Kapitola tři se poté věnuje návrhu samotné aplikace Coffee Shop a zahrnuje popis architektury, použitých technologií a kategorizaci zranitelností obsažených v aplikaci dle již zmíněného projektu OWASP Top 10:2021. Následující, tedy čtvrtá kapitola se hlouběji věnuje jednotlivým zranitelnostem, které jsou v rámci aplikace implementovány, každé jednotlivé zranitelnosti je přiřazen kód CWE a odpovídající název v dle organizace MITRE. Součástí čtvrté kapitoly je taktéž popis jednotlivých zranitelností a také způsob jakým jsou v aplikaci Coffee Shop implementovány. Poslední pátá kapitola se poté věnuje bezpečnostnímu testování výsledné aplikace pomocí platforem GitHub, GitLab a Snyk.

Praktickým výstupem této bakalářské práce je úmyslně zranitelná webová aplikace, která může sloužit jak již bylo v úvodu zmíněno k testování bezpečnostních nástrojů, ale také k výukovým účelům. Vzhledem k rozšířenosti webových aplikací v dnešní době a s tím spojenými kybernetickými útoky na ně je neustálé vzdělání v oblasti bezpečnosti webových aplikací stěžejní, jelikož stejně jako se vyvíjí samotné webové aplikace, tak se vyvíjejí a stávají se komplexnějšími i útoky na ně a lze předpokládat, že tento trend bude i nadále pokračovat..

Literatura

[1] BrowserStack. *Web Application Development in 2023*. Online. Dostupné z: https://www.browserstack.com/guide/web-application-development. [citováno 2023-09-16].

[2] Akamai. *Slipping Through the Security Gaps: The Rise of Application and API Attacks*. Online. Dostupné z: https://www.akamai.com/blog/security/the-rise-of-application-and-api-attacks. [citováno 2023-09-16].

[3] TryHackMe. *TryHackMe Experiences 144% User Growth in 2021*. Online. Dostupné z: https://tryhackme.com/r/resources/blog/tryhackme-rapid-user-growth. [citováno 2023-09-16].

[4] Hack The Box. *Hack The Box reaches 2 million platform members worldwide*. Online. Dostupné z: https://www.hackthebox.com/blog/htb-two-million-platform-members. [citováno 2023-09-16].

[5] OWASP. *About OWASP Foundtaion*. Online. Dostupné z: https://owasp.org/about/. [citováno 2023-09-15].

[6] LinkedIn. *OWASP Foundation*. Online. Dostupné z: https://www.linkedin.com/company/owasp/about/. [citováno 2023-09-15].

[7] OWASP. *Projects*. Online. Dostupné z: https://owasp.org/projects/#. [citováno 2023-09-15].

[8] OWASP. *Other Projects*. Online. Dostupné z: https://owasp.org/other_projects/. [citováno 2023-09-15].

[9] GitHub. *OWASP Application Security Verification Standard*. Online. Dostupné z: https://github.com/OWASP/ASVS. [citováno 2023-09-15].

[10] OWASP. *OWASP Mobile Application Security*. Online. Dostupné z: https://mas.owasp.org/. [citováno 2023-09-16].

[11] OWASP. *OWASP MASVS*. Online. Dostupné z: https://mas.owasp.org/MASVS/. [citováno 2023-09-16].

[12] OWASP. *OWASP MASTG*. Online. Dostupné z: https://mas.owasp.org/MASTG/. [citováno 2023-09-16].

[13] OWASP. *OWASP Top Ten*. Online. Dostupné z: https://owasp.org/www-project-top-ten/. [citováno 2023-09-16].

[14] OWASP. *Welcome to the OWASP Top 10 - 2021*. Online. Dostupné z: https://owasp.org/Top10/. [citováno 2023-09-16].

[15] MITRE. *About CWE*. Online. Dostupné z: https://cwe.mitre.org/about/index.html. [citováno 2023-09-17].

[16] CVE. *About the CVE Program*. Online. Dostupné z: https://www.cve.org/About/Overview. [citováno 2023-09-16].

[17] Codiga. *CVE vs. CWE Vulnerability: What's The Difference*. Online. Dostupné z: https://www.codiga.io/blog/cve-vs-cwe/. [citováno 2023-09-16].

[18] Sucuri. *OWASP Top Security Risks & Vulnerabilities 2021* . Online. Dostupné z: https://sucuri.net/guides/owasp_top_10_2021_edition/. [citováno 2023-09-16].

[19] OWASP. *OWASP Juice Shop*. Online. Dostupné z: https://owasp.org/www-project-juice-shop/. [citováno 2023-09-16].

[20] [Mapping]. In: OWASP. Online. Dostupné z: https://owasp.org/Top10/assets/mapping.png. [citováno 2023-09-16].

[21] OWASP. *A01:2021 - Broken Access Control*. Online. Dostupné z: https://owasp.org/Top10/A01_2021-Broken_Access_Control/. [citováno 2023-09-17].

[22] OWASP. *A02:2021 - Cryptographic Failures*. Online. Dostupné z: https://owasp.org/Top10/A02_2021-Cryptographic_Failures/. [citováno 2023-09-18].

[23] OWASP. *A03:2021 - Injection*. Online. Dostupné z: https://owasp.org/Top10/A03_2021-Injection/. [citováno 2023-09-18].

[24] OWASP. *Injection Prevention Cheat Sheet*. Online. Dostupné z: https://cheatsheetseries.owasp.org/cheatsheets/Injection_Prevention_Cheat_Sheet.html. [citováno 2023-09-20].

[25] OWASP. *A04:2021 - Insecure Design*. Online. Dostupné z: https://owasp.org/Top10/A04_2021-Insecure_Design/. [citováno 2023-09-20].

[26] OWASP. *Secure Product Design Cheat Sheet*. Online. Dostupné z: https://cheatsheetseries.owasp.org/cheatsheets/Secure_Product_Design_Cheat_Sheet.html. [citováno 2023-09-20].

[27] Check Point. *OWASP Top 10 Web Application Security Vulnerabilities*. Online. Dostupné z: https://www.checkpoint.com/cyber-hub/cloud-security/

what-is-application-security-appsec/owasp-top-10-vulnerabilities/. [citováno 2023-09-19].

[28] OWASP. *A05:2021 - Security Misconfiguration*. Online. Dostupné z: https://owasp.org/Top10/A05_2021-Security_Misconfiguration/. [citováno 2023-09-20].

[29] OWASP. *A06:2021 - Vulnerable and Outdated Components*. Online. Dostupné z: https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/. [citováno 2023-09-20].

[30] OWASP. *A07:2021 - Identification and Authentication Failures*. Online. Dostupné z: https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/. [citováno 2023-09-20].

[31] Packetlabs. *Session Management in HTTP: How does it work?*. Online. Dostupné z: https://www.packetlabs.net/posts/session-management/. [citováno 2023-09-20].

[32] OWASP. *A08:2021 - Software and Data Integrity Failures*. Online. Dostupné z: https://owasp.org/Top10/A08_2021-Software_and_Data_Integrity_Failures/. [citováno 2023-09-19].

[33] OWASP. *A09:2021 – Security Logging and Monitoring Failures*. Online. Dostupné z: https://owasp.org/Top10/A09_2021-Security_Logging_and_Monitoring_Failures/. [citováno 2023-09-20].

[34] OWASP. *A10:2021 – Server-Side Request Forgery (SSRF)*. Online. Dostupné z: https://owasp.org/Top10/A10_2021-Server-Side_Request_Forgery_%28SSRF%29/.[citováno 2023-09-20].

[35] GitHub. *DVPWA – Damn Vulnerable Python Web Application*. Online. Dostupné z: https://github.com/anxolerd/dvpwa.[citováno 2023-10-16].

[36] GitHub. *Damn Simple Vulnerable Python Web Application*. Online. Dostupné z: https://github.com/sgabe/DSVPWA.[citováno 2023-10-21].

[37] Google. *Web Application Exploits and Defenses*. Online. Dostupné z: https://google-gruyere.appspot.com/.[citováno 2023-10-21].

[38] OWASP. *OWASP Vulnerable Web Applications Directory*. Online. Dostupné z: https://owasp.org/www-project-vulnerable-web-applications-directory/.[citováno 2023-10-21].

[39] Google. *Setup*. Online. Dostupné z: https://google-gruyere.appspot.com/part1#1__setup.[citováno 2023-10-21].

[40] GitHub. *HeadPage*. Online. Dostupné z: https://github.com/jvlsg/HeadPage.[citováno 2023-10-21].

[41] OWASP. *Pygoat*. Online. Dostupné z: https://owasp.org/www-project-pygoat/.[citováno 2023-10-22].

[42] GitHub. *Pygoat*. Online. Dostupné z: https://github.com/adeyosemanputra/pygoat/tree/master.[citováno 2023-10-22].

[43] GitHub. *Solutions to all the Lab Exercise*. Online. Dostupné z: https://github.com/adeyosemanputra/pygoat/blob/master/Solutions/solution.md.[citováno 2023-10-22].

[44] [Working of MVC]. In: InterviewBit. Online. Dostupné z: https://www.interviewbit.com/blog/wp-content/uploads/2022/05/Working-of-MVC-768x514.png.[citováno 2023-11-29].

[45] LinkedIn. *Understanding MVC Architecture: Simplifying Software Development*. Online. Dostupné z: https://www.linkedin.com/pulse/understanding-mvc-architecture-simplifying-software-development-v-g/.[citováno 2023-11-29].

[46] Ramotion. *MVC Architecture: Simplifying Web Application Development*. Online. Dostupné z: https://www.ramotion.com/blog/mvc-architecture-in-web-application/.[citováno 2023-11-29].

[47] MITRE. *CWE-352: Cross-Site Request Forgery (CSRF)*. Online. Dostupné z: https://cwe.mitre.org/data/definitions/352.html.[citováno 2024-05-02].

[48] PortSwigger. *Cross-site request forgery (CSRF)*. Online. Dostupné z: https://portswigger.net/web-security/csrf.[citováno 2024-05-02].

[49] MITRE. *CWE-425: Direct Request ('Forced Browsing')*. Online. Dostupné z: https://cwe.mitre.org/data/definitions/425.html.[citováno 2024-05-02].

[50] Acunetix. *What Is Forced Browsing*. Online. Dostupné z: https://www.acunetix.com/blog/web-security-zone/what-is-forced-browsing/.[citováno 2024-05-02].

[51] DigitalOcean. *How To Add Authentication to Your App with Flask-Login*. Online. Dostupné z: https://www.digitalocean.com/community/tutorials/

how-to-add-authentication-to-your-app-with-flask-login.[citováno 2024-05-02].

[52] MITRE. *CWE-327: Use of a Broken or Risky Cryptographic Algorithm*. Online. Dostupné z: https://cwe.mitre.org/data/definitions/327.html.[citováno 2024-05-02].

[53] GeeksforGeeks. *Introduction to Hashing – Data Structure and Algorithm Tutorials*. Online. Dostupné z: https://www.geeksforgeeks.org/introduction-to-hashing-data-structure-and-algorithm-tutorials/.[citováno 2024-05-02].

[54] Investopedia. *Cryptographic Hash Functions: Definition and Examples*. Online. Dostupné z: https://www.investopedia.com/news/cryptographic-hash-functions/.[citováno 2024-05-02].

[55] Secret Double Octopus. *Salted Secure Hash Algorithm*. Online. Dostupné z: https://doubleoctopus.com/security-wiki/encryption-and-cryptography/salted-secure-hash-algorithm/.[citováno 2024-05-02].

[56] Educative. *What is the Argon2 function?*. Online. Dostupné z: https://www.educative.io/answers/what-is-the-argon2-function.[citováno 2024-05-02].

[57] Pallets. *Configuration Handling*. Online. Dostupné z: https://flask.palletsprojects.com/en/2.3.x/config/.[citováno 2024-05-02].

[58] Medium. *What is the use of .env file in projects?How to store sensitive information like API keys in .env file?*. Online. Dostupné z: https://medium.com/@sujathamudadla1213/what-is-the-use-of-env-8d6b3eb94843.[citováno 2024-05-02].

[59] MITRE. *CWE-321: Use of Hard-coded Cryptographic Key*. Online. Dostupné z: https://cwe.mitre.org/data/definitions/321.html.[citováno 2024-05-02].

[60] CQR Company. *HARD-CODED CRYPTOGRAPHIC KEYS*. Online. Dostupné z: https://cqr.company/web-vulnerabilities/hard-coded-cryptographic-keys/.[citováno 2024-05-02].

[61] MITRE. *CWE-523: Unprotected Transport of Credentials*. Online. Dostupné z: https://cwe.mitre.org/data/definitions/523.html.[citováno 2024-05-02].

[62] Cloudflare. *What is TLS (Transport Layer Security)?*. Online. Dostupné z: https://www.cloudflare.com/learning/ssl/transport-layer-security-tls/.[citováno 2024-05-02].

[63] MITRE. *CWE-798: Use of Hard-coded Credentials*. Online. Dostupné z: https://cwe.mitre.org/data/definitions/798.html.[citováno 2024-05-02].

[64] Medium. *What is the use of .env file in projects?How to store sensitive information like API keys in .env file?*. Online. Dostupné z: https://medium.com/@sujathamudadla1213/what-is-the-use-of-env-8d6b3eb94843.[citováno 2024-05-02].

[65] MITRE. *CWE-79: Improper Neutralization of Input During Web Page Generation ('Cross-site Scripting')*. Online. Dostupné z: https://cwe.mitre.org/data/definitions/79.html.[citováno 2024-05-01].

[66] PortSwigger. *Cross-site scripting*. Online. Dostupné z: https://portswigger.net/web-security/cross-site-scripting.[citováno 2024-05-01].

[67] OWASP. *Cross Site Scripting (XSS)*. Online. Dostupné z: https://owasp.org/www-community/attacks/xss/.[citováno 2024-05-01].

[68] OWASP. *DOM Based XSS*. Online. Dostupné z: https://owasp.org/www-community/attacks/DOM_Based_XSS.[citováno 2024-05-01].

[69] Pallets. *Security Considerations*. Online. Dostupné z: https://flask.palletsprojects.com/en/3.0.x/web-security/.[citováno 2024-05-01].

[70] He3. *Understanding HTML Escape: Keeping Your Content Safe*. Online. Dostupné z: https://he3app.com/en/blogs/understanding-html-escape-keeping-your-content-safe/.[citováno 2024-05-01].

[71] MITRE. *CWE-89: Improper Neutralization of Special Elements used in an SQL Command ('SQL Injection')*. Online. Dostupné z: https://cwe.mitre.org/data/definitions/89.html.[citováno 2024-05-01].

[72] PortSwigger. *SQL injection*. Online. Dostupné z: https://portswigger.net/web-security/sql-injection.[citováno 2024-05-01].

[73] OWASP. *SQL Injection Prevention Cheat Sheet*. Online. Dostupné z: https://cheatsheetseries.owasp.org/cheatsheets/SQL_Injection_Prevention_Cheat_Sheet.html.[citováno 2024-05-01].

[74] MITRE. *CWE-434: Unrestricted Upload of File with Dangerous Type*. Online. Dostupné z: https://cwe.mitre.org/data/definitions/434.html.[citováno 2024-05-02].

[75] PortSwigger. *File upload vulnerabilities*. Online. Dostupné z: https://portswigger.net/web-security/file-upload.[citováno 2024-05-02].

[76] Mozilla. *MIME types (IANA media types)*. Online. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Basics_of_HTTP/MIME_types.[citováno 2024-05-02].

[77] MITRE. *CWE-78: Improper Neutralization of Special Elements used in an OS Command ('OS Command Injection')*. Online. Dostupné z: https://cwe.mitre.org/data/definitions/78.html.[citováno 2024-05-02].

[78] PortSwigger. *OS command injection*. Online. Dostupné z: https://portswigger.net/kb/issues/00100100_os-command-injection.[citováno 2024-05-02].

[79] MITRE. *CWE-1021: Improper Restriction of Rendered UI Layers or Frames*. Online. Dostupné z: https://cwe.mitre.org/data/definitions/1021.html.[citováno 2024-05-02].

[80] PortSwigger. *Clickjacking (UI redressing)*. Online. Dostupné z: https://portswigger.net/web-security/clickjacking.[citováno 2024-05-02].

[81] OWASP. *Clickjacking Defense Cheat Sheet*. Online. Dostupné z: https://cheatsheetseries.owasp.org/cheatsheets/Clickjacking_Defense_Cheat_Sheet.html.[citováno 2024-05-02].

[82] MITRE. *CWE-200: Exposure of Sensitive Information to an Unauthorized Actor* . Online. Dostupné z: https://cwe.mitre.org/data/definitions/200.html.[citováno 2024-05-01].

[83] MITRE. *CWE-521: Weak Password Requirements*. Online. Dostupné z: https://cwe.mitre.org/data/definitions/521.html.[citováno 2024-05-02].

[84] Jetpack. *How Weak Passwords Expose You to Serious Security Risks*. Online. Dostupné z: https://jetpack.com/blog/weak-passwords/.[citováno 2024-05-02].

[85] Rapid7. *Brute-Force and Dictionary Attacks*. Online. Dostupné z: https://www.rapid7.com/fundamentals/brute-force-and-dictionary-attacks/.[citováno 2024-05-02].

[86] MITRE. *CWE-315: Cleartext Storage of Sensitive Information in a Cookie*. Online. Dostupné z: https://cwe.mitre.org/data/definitions/315.html.[citováno 2024-05-02].

[87] OWASP. *Test Role Definitions*. Online. Dostupné z: https://owasp.org/www-project-web-security-testing-guide/latest/4-Web_Application_Security_Testing/03-Identity_Management_Testing/01-Test_Role_Definitions.[citováno 2024-05-02].

[88] Imperva. *Parameter Tampering*. Online. Dostupné z: https://www.imperva.com/learn/application-security/parameter-tampering/.[citováno 2024-05-02].

[89] PyPI. *flask-unsign 1.2.0*. Online. Dostupné z: https://pypi.org/project/flask-unsign/.[citováno 2024-05-02].

[90] MITRE. *CWE-756: Missing Custom Error Page*. Online. Dostupné z: https://cwe.mitre.org/data/definitions/756.html.[citováno 2024-05-02].

[91] MITRE. *CWE-489: Active Debug Code*. Online. Dostupné z: https://cwe.mitre.org/data/definitions/489.html.[citováno 2024-05-02].

[92] HackTrikcs. *Werkzeug / Flask Debug*. Online. Dostupné z: https://book.hacktricks.xyz/network-services-pentesting/pentesting-web/werkzeug.[citováno 2024-05-02].

[93] HackTrikcs. *What is Werkzeug?*. Online. Dostupné z: https://testdriven.io/blog/what-is-werkzeug/.[citováno 2024-05-02].

[94] Pallets. *Debugging Applications*. Online. Dostupné z: https://werkzeug.palletsprojects.com/en/3.0.x/debug/.[citováno 2024-05-02].

[95] MITRE. *CWE-35: Path Traversal: '.../...//'*. Online. Dostupné z: https://cwe.mitre.org/data/definitions/35.html.[citováno 2024-05-02].

[96] PortSwigger. *Path traversal*. Online. Dostupné z: https://portswigger.net/web-security/file-path-traversal.[citováno 2024-05-02].

[97] MITRE. *CWE-548: Exposure of Information Through Directory Listing*. Online. Dostupné z: https://cwe.mitre.org/data/definitions/548.html.[citováno 2024-05-02].

[98] MITRE. *CWE-1275: Sensitive Cookie with Improper SameSite Attribute*. Online. Dostupné z: https://cwe.mitre.org/data/definitions/1275.html.[citováno 2024-05-02].

[99] PortSwigger. *Bypassing SameSite cookie restrictions*. Online. Dostupné z: https://portswigger.net/web-security/csrf/bypassing-samesite-restrictions.[citováno 2024-05-02].

[100] Mozilla. *Using HTTP cookies*. Online. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Cookies.[citováno 2024-05-02].

[101] MITRE. *CWE CATEGORY: OWASP Top Ten 2017 Category A9 - Using Components with Known Vulnerabilities*. Online. Dostupné z: https://cwe.mitre.org/data/definitions/1035.html.[citováno 2024-05-02].

[102] Vumetric. *OWASP Top 10 – A06 Vulnerable And Outdated Components Explained*. Online. Dostupné z: https://www.vumetric.com/blog/owasp-top-10-a06-vulnerable-and-outdated-components-explained/. [citováno 2024-05-02].

[103] NIST. *CVE-2019-9193 Detail*. Online. Dostupné z: https://nvd.nist.gov/vuln/detail/CVE-2019-9193. [citováno 2024-05-02].

[104] Exploit Database. *PostgreSQL 9.6.1 - Remote Code Execution (RCE) (Authenticated)*. Online. Dostupné z: https://www.exploit-db.com/exploits/51247. [citováno 2024-05-02].

[105] MITRE. *CWE-307: Improper Restriction of Excessive Authentication Attempts*. Online. Dostupné z: https://cwe.mitre.org/data/definitions/307.html.[citováno 2024-05-02].

[106] OWASP. *Blocking Brute Force Attacks*. Online. Dostupné z: https://owasp.org/www-community/controls/Blocking_Brute_Force_Attacks.[citováno 2024-05-02].

[107] MITRE. *CWE-639: Authorization Bypass Through User-Controlled Key*. Online. Dostupné z: https://cwe.mitre.org/data/definitions/639.html.[citováno 2024-05-02].

[108] PortSwigger. *Insecure direct object references (IDOR)*. Online. Dostupné z: https://portswigger.net/web-security/access-control/idor.[citováno 2024-05-02].

[109] MITRE. *CWE-613: Insufficient Session Expiration*. Online. Dostupné z: https://cwe.mitre.org/data/definitions/613.html.[citováno 2024-05-01].

[110] MITRE. *CWE-830: Inclusion of Web Functionality from an Untrusted Source*. Online. Dostupné z: https://cwe.mitre.org/data/definitions/830.html.[citováno 2024-05-02].

[111] MITRE. *CWE-532: Insertion of Sensitive Information into Log File*. Online. Dostupné z: https://cwe.mitre.org/data/definitions/532.html.[citováno 2024-05-02].

[112] MITRE. *CWE-778: Insufficient Logging*. Online. Dostupné z: https://cwe.mitre.org/data/definitions/778.html.[citováno 2024-05-02].

[113] MITRE. *CWE-918: Server-Side Request Forgery (SSRF)*. Online. Dostupné z: https://cwe.mitre.org/data/definitions/918.html. [citováno 2024-05-02].

[114] Medium. *Server-side request forgery (SSRF)*. Online. Dostupné z: https://medium.com/@errorfiathck/server-side-request-forgery-ssrf-e00b585b6f67. [citováno 2024-05-02].

[115] PortSwigger. *Server-side request forgery (SSRF)*. Online. Dostupné z: https://portswigger.net/web-security/ssrf.[citováno 2024-05-02].

[116] Acunetix. *Port scanning with Server Side Request Forgery (SSRF)*. Online. Dostupné z: https://www.acunetix.com/blog/articles/ssrf-vulnerability-used-to-scan-the-web-servers-network/.[citováno 2024-05-02].

[117] GitHub. *The tools you need to build what you want.*. Online. Dostupné z: https://github.com/features.[citováno 2024-05-02].

[118] GitLab. *GitLab Features*. Online. Dostupné z: https://about.gitlab.com/features/.[citováno 2024-05-02].

[119] Snyk. *What is Snyk?*. Online. Dostupné z: https://snyk.io/product/.[citováno 2024-05-02].

[120] GitHub. *About code scanning with CodeQL*. Online. Dostupné z: https://docs.github.com/en/code-security/code-scanning/introduction-to-code-scanning/about-code-scanning-with-codeql#about-codeql-queries.[citováno 2024-05-02].

[121] GitLab. *Static Application Security Testing (SAST)*. Online. Dostupné z: https://docs.gitlab.com/ee/user/application_security/sast/.[citováno 2024-05-02].

Seznam symbolů a zkratek

AJAX	Aynchronous JavaScript and XML
ASVS	Application Security Verification Standard
CI/CD	Continuous Integration and Continuous Delivery)
CSS	Cascading Style Sheets
CSRF	Cross-Site Request Forgery
CTF	Capture The Flag
CVE	Common Vulnerabilities and Exposures
CWE	Common Weakness Enumeration
DHS	U.S. Department of Homeland Security
DoS	Denial of Service
DSVPWA	Damn Simple Vulnerable Python Web Application
DVPWA	Damn Vulnerable Python Web Application
EAR	Execution After Redirec
ERD	Entity-Relationship Diagram
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
MAS	Mobile Application Security
MASTG	Mobile Application Security Testing Guide
MASVS	Mobile Application Security Verification Standard
MIME	Multipurpose Internet Mail Extensions
MVC	Model-View-Controller
OWASP	The Open Worldwide Application Security Project
RCE	Remote Code Execution

SAST	Static Application Security Testing
SQL	Structured Query Language
SSRF	Server-Side Request Forgery
SSTI	Server-Side Template Injection
TLD	Top-Level Domain
TLS	Transport Layer Security
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
WSGI	Web Server Gateway Interface
XML	Extensible Markup Language
XSS	Cross-site Scripting
XSSI	Cross-Site Script Inclusion
XXE	XML External Entities