FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Bezpečnostní analýza aplikace Anketa ČVUT
Student:	Vojtěch Zabořil
Vedoucí:	Ing. Josef Kokeš, Ph.D.
Studijní program:	Informatika
Obor / specializace:	Informační bezpečnost 2021
Katedra:	Katedra informační bezpečnosti
Platnost zadání:	do konce letního semestru 2024/2025

Pokyny pro vypracování

1) Seznamte se s aplikací Anketa ČVUT jak z pohledu uživatele, tak z pohledu vývojáře.
2) Proveďte rešerši aktuálně používaných metodologií pro provádění bezpečnostních analýz webových aplikací.
3) Vyberte vhodnou metodologii a aplikujte ji na aplikaci Anketa ČVUT. Zaměřte se zejména na zranitelnosti umožňující spuštění útočníkova kódu nebo prolomení anonymity respondentů.
4) Vyhodnoťte závažnost nalezených zranitelností a navrhněte opravu (počínaje těmi nejzávažnějšími).
5) Zformulujte doporučení pro budoucí vývojáře Ankety, kterým oblastem věnovat zvýšenou pozornost.

Elektronicky schválil/a prof. Ing. Róbert Lórencz, CSc. dne 10. ledna 2024 v Praze.

Bakalářská práce

BEZPEČNOSTNÍ ANALÝZA APLIKACE ANKETA ČVUT

Vojtěch Zabořil

Fakulta informačních technologií
Katedra informační bezpečnosti
Vedoucí: Ing. Josef Kokeš, Ph.D.
15. května 2024

Obsah

Poděkování viii

Prohlášení ix

Abstrakt x

Seznam zkratek xii

Úvod 1

1 Aplikace Anketa ČVUT 3
- 1.1 Význam aplikace . . . 3
- 1.2 Role uživatelů . . . 3
- 1.3 Cyklus ankety . . . 4
- 1.4 Historie . . . 4
- 1.5 Technologie a architektura aplikace . . . 5
- 1.6 Jednotlivá prostředí . . . 5

2 Metodologie pro provádění bezpečnostních analýz 6
- 2.1 Význam a zaměření používaných metodologií . . . 6
  - 2.1.1 OWASP Web Security Testing Guide . . . 7
  - 2.1.2 Penetration Testing Execution Standard . . . 7
  - 2.1.3 Open Source Security Testing Methodology Manual . . . 8
  - 2.1.4 NIST SP 800-115 . . . 8
  - 2.1.5 Výběr metodologie . . . 8
- 2.2 Hodnocení závažnosti zranitelností . . . 8

3 Bezpečnostní analýza aplikace 10
- 3.1 Information Gathering . . . 10
  - 3.1.1 Conduct Search Engine Discovery Reconnaissance for Information Leakage . . . 11
  - 3.1.2 Fingerprint Web Server . . . 11
  - 3.1.3 Review Webserver Metafiles for Information Leakage . . . 12
  - 3.1.4 Enumerate Applications on Webserver . . . 12
  - 3.1.5 Review Webpage Content for Information Leakage . . . 12
  - 3.1.6 Identify Application Entry Points . . . 12

3.1.7 Map Execution Paths Through Application 14
3.1.8 Fingerprint Web Application Framework 14
3.1.9 Fingerprint Web Application 14
3.1.10 Map Application Architecture 15
3.2 Configuration and Deployment Management Testing 15
3.2.1 Test Network Infrastructure Configuration 15
3.2.2 Test Application Platform Configuration 15
3.2.3 Test File Extensions Handling for Sensitive Information 16
3.2.4 Review Old Backup and Unreferenced Files for Sensitive Information 16
3.2.5 Enumerate Infrastructure and Application Admin Interfaces 16
3.2.6 Test HTTP Methods 17
3.2.7 Test HTTP Strict Transport Security 17
3.2.8 Test RIA Cross Domain Policy 17
3.2.9 Test File Permission 18
3.2.10 Test for Subdomain Takeover 18
3.2.11 Test Cloud Storage 18
3.3 Identity Management Testing 18
3.3.1 Test Role Definitions 18
3.3.2 Test User Registration Process 19
3.3.3 Test Account Provisioning Process 19
3.3.4 Testing for Account Enumeration and Guessable User Account 19
3.3.5 Testing for Weak or Unenforced Username Policy 19
3.4 Authentication Testing 20
3.4.1 Testing for Credentials Transported over an Encrypted Channel 21
3.4.2 Testing for Default Credentials 21
3.4.3 Testing for Weak Lock Out Mechanism 21
3.4.4 Testing for Bypassing Authentication Schema 21
3.4.5 Testing for Vulnerable Remember Password 22
3.4.6 Testing for Browser Cache Weaknesses 22
3.4.7 Testing for Weak Password Policy 22
3.4.8 Testing for Weak Security Question Answer 22
3.4.9 Testing for Weak Password Change or Reset Functionalities 22
3.4.10 Testing for Weaker Authentication in Alternative Channel 22
3.5 Authorization Testing 23
3.5.1 Testing Directory Traversal File Include 23
3.5.2 Testing for Bypassing Authorization Schema 23
3.5.3 Testing for Privilege Escalation 24
3.5.4 Testing for Insecure Direct Object References 24

3.6 Session Management Testing 24
3.6.1 Testing for Session Management Schema 24
3.6.2 Testing for Cookies Attributes 25
3.6.3 Testing for Session Fixation 27
3.6.4 Testing for Exposed Session Variables 27
3.6.5 Testing for Cross Site Request Forgery 27
3.6.6 Testing for Logout Functionality 28
3.6.7 Testing Session Timeout 28
3.6.8 Testing for Session Puzzling 28
3.6.9 Testing for Session Hijacking 28
3.7 Input Validation Testing 29
3.7.1 Testing for Reflected Cross Site Scripting 29
3.7.2 Testing for Stored Cross Site Scripting 30
3.7.3 Testing for HTTP Verb Tampering 30
3.7.4 Testing for HTTP Parameter Pollution 30
3.7.5 Testing for SQL Injection 30
3.7.6 Testing for LDAP Injection 31
3.7.7 Testing for XML Injection 31
3.7.8 Testing for SSI Injection 32
3.7.9 Testing for XPath Injection 32
3.7.10 Testing for IMAP SMTP Injection 32
3.7.11 Testing for Code Injection 33
3.7.12 Testing for Command Injection 33
3.7.13 Testing for Format String Injection 33
3.7.14 Testing for Incubated Vulnerability 33
3.7.15 Testing for HTTP Splitting Smuggling 34
3.7.16 Testing for HTTP Incoming Requests 34
3.7.17 Testing for Host Header Injection 34
3.7.18 Testing for Server-side Template Injection 34
3.7.19 Testing for Server-Side Request Forgery 34
3.8 Testing for Error Handling 35
3.8.1 Testing for Improper Error Handling 35
3.8.2 Testing for Stack Traces 35
3.9 Testing for Weak Cryptography 35
3.9.1 Testing for Weak Transport Layer Security 36
3.9.2 Testing for Padding Oracle 37
3.9.3 Testing for Sensitive Information Sent via Unencrypted Channels 37
3.9.4 Testing for Weak Encryption 37
3.10 Business Logic Testing 38
3.10.1 Test Business Logic Data Validation 38
3.10.2 Test Ability to Forge Requests 38
3.10.3 Test Integrity Checks 38
3.10.4 Test for Process Timing 38

3.10.5 Test Number of Times a Function Can Be Used Limits . 39
3.10.6 Testing for the Circumvention of Work Flows 39
3.10.7 Test Defenses Against Application Misuse 40
3.10.8 Test Upload of Unexpected File Types 40
3.10.9 Test Upload of Malicious Files 40
3.11 Client-side Testing 41
3.11.1 Testing for DOM-Based Cross Site Scripting 41
3.11.2 Testing for JavaScript Execution 41
3.11.3 Testing for HTML Injection 41
3.11.4 Testing for Client-side URL Redirect 41
3.11.5 Testing for CSS Injection 42
3.11.6 Testing for Client-side Resource Manipulation 42
3.11.7 Testing Cross Origin Resource Sharing 42
3.11.8 Testing for Cross Site Flashing 43
3.11.9 Testing for Clickjacking 43
3.11.10 Testing WebSockets 43
3.11.11 Testing Web Messaging 43
3.11.12 Testing Browser Storage 43
3.11.13 Testing for Cross Site Script Inclusion 44
3.12 API Testing 44
3.12.1 Testing GraphQL 44

4 Vyhodnocení a oprava nalezených zranitelností 45
4.1 Session Management 46
4.2 Cookie parametry 47
4.3 Nedostatečné logování a obrana proti zneužití 48
4.4 Process Timing při autentizaci 48
4.5 Chybějící Lock Out mechanismus 49
4.6 Password Policy 49
4.7 Platnost tokenů pro tvorbu hesla 50
4.8 Přístupnost vývojového prostředí a použitá data 50
4.9 Verze webserveru 51
4.10 Hashování hesel 51
4.11 Načtení webu v iframe 52
4.12 Chybné napojení FE a BE 52
4.13 Nevalidace vstupů ve vývojovém prostředí 53
4.14 Proces registrace uživatelů 53

5 Doporučení pro vývojáře Ankety ČVUT 54

6 Závěr 56

Obsah příloh 61

Seznam obrázků

3.1 Skenování pomocí Nikto 11
3.2 Skenování serveru pomocí nmap 12
3.3 WhatWeb sken 15
3.4 Nástroj gospider 16
3.5 HTTP metody 17
3.6 HTTP Strict Transport Security 17
3.7 DNS záznamy pro doménu aplikace 18
3.8 JWT token 25
3.9 Parametry Session Cookie 27
3.10 Testování Stored XSS 30
3.11 Ukázka SQL Injection 31
3.12 Odpověď serveru při chybném požadavku 35
3.13 Skenování pomocí sslscan 36
3.14 Testování CORS 42
3.15 Požadavek na načtení v iframe 43

Seznam tabulek

2.1 Hodnocení závažnosti zranitelností CVSS 9

3.1 GET požadavky na FE a jejich parametry 13
3.2 GET požadavky na BE a jejich parametry 13
3.3 POST požadavky na BE a jejich parametry 14

4.1 Nalezené zranitelnosti 45

Seznam výpisů kódu

3.1 Zpracování autentizace 20
3.2 Ověření autorizace 23
3.3 Tvorba JWT tokenu 25
3.4 Kontrola JWT tokenu 26
3.5 Odhlášení na FE 28
3.6 Autentizace jako libovolný uživatel ve vývojovém prostředí . . . 29
3.7 Funkce s SQL dotazem 31
3.8 Použití LDAP při autentizaci 32
3.9 Hashování hesel 37
3.10 Process Timing v autentizaci 39
3.11 Ověření předchozího nevyplnění ankety 40
3.12 Přesměrování při přihlášení 42

Chtěl bych poděkovat především svému vedoucímu Ing. Josefu Kokešovi, Ph.D. za zodpovědné vedení mé práce a také za mnoho užitečných rad a postřehů. Stejně tak musím poděkovat Ing. Michalu Valentovi, Ph.D., který mi vždy poskytl nezbytné informace a rady ohledně testované aplikace. Nakonec bych chtěl poděkovat své rodině a přítelkyni za podporu nejen při tvorbě této práce, ale i celém studiu.

Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací.

Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 15. května 2024

Abstrakt

Tato bakalářská práce se soustředí na provedení bezpečnostní analýzy Ankety ČVUT, což je webová aplikace sloužící k anonymnímu hodnocení zapsaných předmětů a jejich vyučujících. Pro realizaci bezpečnostní analýzy je zvolena metodologie OWASP Web Security Testing Guide. Tato metodologie je posléze použita k samotnému testování aplikace Anketa ČVUT, v rámci něhož bylo odhaleno 14 zranitelností nebo chyb. Nalezena byla například kriticky závažná zranitelnost týkající se Session Managementu nebo chybné nastavení parametrů Cookies, které je ohodnoceno vysokou závažností. V práci jsou nalezené zranitelnosti vyhodnoceny z hlediska bezpečnosti a je navržena jejich oprava. Výstupem práce je sada doporučení k provedení změn v aplikaci Anketa ČVUT, především z hlediska zvýšení bezpečnosti této aplikace.

Klíčová slova bezpečnostní analýza, webová aplikace, online anketní systém, Anketa ČVUT, metodologie bezpečnostní analýzy, OWASP Web Security Testing Guide

Abstract

This bachelor's thesis focuses on performing a security analysis of the CTU Survey – a web application used for an anonymous evaluation of enrolled subjects and their teachers. The OWASP Web Security Testing Guide methodology is chosen for this security analysis. This methodology is then used to test the CTU Survey application itself. 14 vulnerabilities or errors were discovered during this test, including a critical vulnerability related to Session Management or an incorrect setting of Cookies parameters which is rated as high severity. The found vulnerabilities are evaluated from the point of view of security and fixes are proposed. The output of the work is a set of recommendations for making changes in the CTU Survey application, primarily from the point of view of increasing the security of this application.

Keywords security analysis, web application, online survey system, CTU Survey, security analysis methodology, OWASP Web Security Testing Guide

Seznam zkratek

API	Application Programming Interface
BE	backend
CORS	Cross Origin Resource Sharing
CSRF	Cross Site Request Forgery
CSS	Cascading Style Sheets
CVSS	Common Vulnerability Scoring System
ČVUT	České vysoké učení technické v Praze
DNS	Domain Name System
DOM	Document Object Model
FIT	Fakulta informačních technologií
FE	frontend
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Hypertext Transfer Protocol Secure
IDOR	Insecure Direct Object References
IP	Internet Protocol
JS	JavaScript
JSON	JavaScript Object Notation
JWT	JSON Web Token
KOS	Komponenta studium, studijní informační systém ČVUT
LDAP	Lightweight Directory Access Protocol
NIST	National Institute of Standards and Technology
OSSTMM	Open Source Security Testing Methodology Manual
OWASP	Open Web Application Security Project
PTES	Penetration Testing Execution Standard
RIA	Rich Internet Applications
SQL	Structured query language
SSI	Server Side Includes
SSO	Single sign-on
URL	Uniform Resource Locator
VPN	Virtual Private Network
WSTG	Web Security Testing Guide
XML	Extensible Markup Language
XSS	Cross Site scripting
XSSI	Cross Site Script Inclusion

Úvod

Každým rokem je odhaleno a popsáno stále více zranitelností v rámci bezpečnosti informačních technologií – jejich počet se v posledních 10 letech zvýšil téměř 6krát. [1] Jejich zneužití je zároveň stále jednodušší, a to také pomocí automatizovaných nástrojů, které dokáží zjišťovat, zda mají dané systémy konkrétní zranitelnost. Zneužití takovýchto zranitelností může útočník využít pro získání cenných informací (osobní údaje, …), získání finančních prostředků (bankovnictví, těžba kryptoměn, …) a také narušení funkcí systému (DoS – odmítnutí služby, …). V posledních letech se navíc ukazuje, že útočníci cílí i na vzdělávací instituce – především na vysoké školy. [2] Tento trend se nevyhnul ani ČVUT v Praze – např. Fakulta jaderná a fyzikálně inženýrská čelila útoku v roce 2023. [3]

Bezpečnost aplikace je nutná pro zabránění jejího zneužití útočníkem. Druhotným důvodem pro bezpečnost je důvěra zákazníků/uživatelů v aplikaci (resp. i organizaci, která ji provozuje). Závažné problémy s bezpečností aplikace odrazují její uživatele (i potencionální) od jejího používání. Uživatelé se pak oprávněně bojí zneužití jejich údajů, nedostupnosti dané služby, …

Práce je určena především pro budoucí vývojáře této aplikace. Práce bude pro ně přínosem díky upozornění na chyby, kterých se dopustili vývojáři aktuální verze aplikace.

Motivací pro danou problematiku je zájem o odhalování bezpečnostních problémů ve webových aplikacích, a to za účelem jejich budoucího vylepšení a zvýšení jejich odolnosti proti kybernetickým útokům.

Zaměřením práce je otestování webové aplikace Anketa ČVUT, a to především z hlediska zranitelností umožňujících spuštění útočníkova kódu nebo prolomení anonymity respondentů. Práce se nezaměřuje na testování samotného způsobu zajištění anonymity.

Práce navazuje na samotný vývoj této aplikace, který provedli Oleksandr Chmel [4], Sengül Furkan [5] a také vývojáři předchozích verzí aplikace. Testování anonymity respondentů v aplikaci Anketa ČVUT se zabývala Eliška Helikarová [6]. Tato práce je tedy i součástí projektu Anketa ČVUT.

Cílem bakalářské práce je provést bezpečnostní analýzu aplikace Anketa ČVUT podle zvolené metodologie. Nejprve je popsána existující aplikace se zaměřením na informace, které jsou relevantní vzhledem k testování. Dalším cílem je shrnout aktuálně používané metodologie pro testování a vybrat vhodnou metodologii pro analýzu aplikace Anketa ČVUT. Následujícím krokem je samotné provedení analýzy pomocí zvolené metodologie. Dále jsou vyhodnoceny zranitelnosti a chyby nalezené během analýzy; větší důraz je kladen na zranitelnosti s vyšší závažností. Jsou také navrženy způsoby, jakými zmírnit nebo eliminovat dopady nalezených zranitelností. Cílem práce je také vytvořit sadu doporučení pro budoucí vývojáře. Tato doporučení se zaměřují na části aplikace s větším množstvím nalezených zranitelností.

Kapitola 1

Aplikace Anketa ČVUT

Tato kapitola je zahájena popisem použití aplikace k hodnocení předmětů a učitelů. Dále jsou popsány jednotlivé fáze, kterými konkrétní anketa prochází, včetně toho, kdo hodnocení může v těchto fázích vytvářet a komentovat. Poté následuje stručná historie aplikace Anketa ČVUT spolu s popisem způsobu jejího vývoje. V krátkosti jsou popsány technologie, které aplikace aktuálně využívá. Závěr kapitoly je soustředěn na infrastrukturu aplikace a vztahy mezi vývojovým, stage a produkčním prostředím.

1.1 Význam aplikace

Aplikace Anketa ČVUT je určena pro hodnocení předmětů a učitelů a tvoří jak zpětnou vazbu pro učitele, tak umožňuje studentům získat informace o obtížnosti jednotlivých částí předmětu nebo přístupu vyučujících od těch, kteří předmět již absolvovali.

Principem Ankety ČVUT je možnost hodnotit a komentovat dokončené i nedokončené předměty, jejich vyučující nebo celou fakultu. To vše se zaměřením na anonymitu uživatelů.

Informace o procesech a událostech souvisejících s Anketou ČVUT jsou detailně popsány v [7].

1.2 Role uživatelů

V systému jsou definovány uživatelské role. Zde je výčet těch nejdůležitějších a jejich určení (pro správu Ankety se však využívá samostatná aplikace, jejíž testování není předmětem této práce):

- Správce systému – příprava ankety, životní cyklus.
- Fakultní správce – úprava předmětů a vyučujících před spuštěním ankety.

- Student – komentování zapsaných předmětů na konci semestru.
- Vyučující – reakce na komentáře studentů. [7]

1.3 Cyklus ankety

Instance ankety je spuštěna dvakrát ročně – po letním a po zimním semestru. Každá instance postupně prochází těmito stavy:

1. Příprava – import z KOS a další ruční úpravy předmětů.
2. Vyplnění – vyplnění anketních lístků studenty.
   a. Dokončené předměty – lze vyplnit pouze dokončené předměty.
   b. Zapsané předměty – lze vyplnit i nedokončené předměty.
3. Uzavření – studenti nemohou vyplňovat, vyučující mají možnost se vyjádřit k hodnocení studentů dle stavů níže.
   a. Owner – vyučující mohou pouze reagovat na komentáře ke svojí osobě nebo vyučovaným předmětům.
   b. All – učitelé mohou komentovat hodnocení ostatních vyučujících a hodnocení předmětů.
   c. Read-only – již nelze komentovat. [7]

1.4 Historie

První verze Ankety ČVUT vznikla v roce 2002 na Fakultě elektrotechnické ČVUT. Posléze začala být používána i ostatními fakultami. V roce 2007 byla Robertem Jiříkem vytvořena nová verze aplikace. Aplikace byla dále vylepšována pod vedením Ing. Michala Valenty, Ph.D. především jako součást závěrečných prací studentů.

V roce 2018 byl v rámci diplomové práce Davida Knapa představen nový design celé aplikace – verze 3.0. [8]

Posledním vylepšením aplikace je nová verze backendu i frontendu; tyto verze jsou v době dokončení této práce nasazeny pouze ve vývojovém prostředí. Autorem nové verze backendu je Oleksandr Chmel (v rámci diplomové práce) [4], frontend vytvořil Sengül Furkan (v rámci Internship). [5] Aktuálně probíhají další úpravy pod vedení Oleksandra Chmela.

Většina aplikace Anketa ČVUT (včetně jejích vylepšení) vznikala jako součást závěrečných prací. Lze tedy očekávat méně zkušeností s vývojem takovýchto aplikací a větší množství chyb. Naopak lze předpokládat, že tvůrci mají aktuálnější znalosti teorie ze svého studia, což může eliminovat určité druhy chyb, které jsou dnes již velice známé (např. SQL Injection, nehashování hesel, Cross Site Scripting, …).

1.5 Technologie a architektura aplikace

Frontend poslední verze aplikace (v dubnu 2024 je tato verze pouze ve vývojovém prostředí) je postaven na technologii React. Na backendu je použit programovací jazyk Kotlin (tato verze je v dubnu 2024 pouze ve vývojovém prostředí). Pro uložení aplikačních dat je použita databáze Oracle.

1.6 Jednotlivá prostředí

K provozu, vývoji a testování aplikace slouží 3 prostředí. Jedná se o vývojové prostředí, které slouží k vývoji, testování a ladění nových verzí aplikace nebo jejich úprav. Stagging prostředí pak slouží k většímu uživatelskému testování v případě výraznějších úprav aplikace. Posledním je pak produkční prostředí, ve kterém běží ostrá verze aplikace přístupná všem studentům i vyučujícím ČVUT.

Mezi prostředími probíhá synchronizace dat tak, aby bylo možné provádět testování na reálných datech; vždy se tedy jedná o data pocházející z produkčního prostředí.

Kapitola 2

Metodologie pro provádění bezpečnostních analýz

V této kapitole je popsán význam metodologií pro provádění bezpečnostních analýz. Metodologie, které jsou v současnosti nejpoužívanější, jsou detailněji prozkoumány. Metodologiím, které jsou buď neaktuální nebo se zaměřují na jinou oblast než bezpečnostní analýzu nebo penetrační testování, není věnována žádná pozornost. V závěru kapitoly je vybrána metodologie pro testování Ankety ČVUT, spolu s důvody tohoto výběru. Dále je také uvedeno, jakým způsobem je hodnocena závažnost nalezených zranitelností a proč je toto hodnocení nezbytné.

2.1 Význam a zaměření používaných metodologií

Cílem metodologie pro bezpečnostní analýzu nebo penetrační testování je poskytnutí postupů a metod nutných pro provedení těchto testů. Ty slouží k nalezení existujících zranitelností a chyb, a to především proto, aby se zamezilo jejich zneužití potencionálními útočníky; případně k ověření, že je daný systém nebo aplikace v rámci určité oblasti bezpečná. Metodologie slouží také pro popis a určení rizik spojených s těmito zranitelnostmi včetně způsobů, jak jejich dopad eliminovat nebo zmírnit. Metodologie mají dále zajistit konzistenci těchto analýz a zajistit neopomenutí některého typu zranitelností. [9]

V současnosti jsou nejpoužívanějšími metodologiemi *The OWASP Web Security Testing Guide*, *The Open Source Security Testing Methodology Manual* a *NIST SP 800-115*. [9] Dle [10] je používán ještě *Penetration Testing Execution Standard*.

2.1.1 OWASP Web Security Testing Guide

Jedná se o projekt vytvořený organizací *Open Web Application Security Project* – tato organizace se zabývá bezpečností webových aplikací a také tvorbou návodů a doporučení týkajících se tohoto tématu. [11]

OWASP WSTG se zaměřuje především na testování webových aplikací. Metodologie je členěna do kapitol dle jednotlivých kategorií hrozeb (testování metod HTTP, uživatelské role, eskalace privilegií, …). V každém bodu je také popsáno, jakými technikami je možné tuto hrozbu realizovat, což pomáhá testerovi ve zkoušení různých postupů *útoku*. Metodologie je vytvořena systematicky, jednotlivé body proto pokrývají naprostou většinu hypotetických možností, jak lze na aplikaci *zaútočit*. [11]

Postup testování je rozdělen na 12 částí, z nichž každá je ještě dále členěna. První částí je *Information Gathering* – tedy získávání informací o subjektu testování. Každá z dalších částí se pak věnuje testování jedné kategorie zranitelností (např. autentizace, ověřování uživatelských vstupů, …). Metodologie se také v krátkosti zaobírá tvorbou závěrečné zprávy o testování – reportu – především se zaměřením na její členění. [11]

2.1.2 Penetration Testing Execution Standard

PTES je metodologií, která je spravována skupinou odborníků z různých oborů informační bezpečnosti. Metodologie byla vytvořena z důvodu standardizace nejen procesu testování, ale i tvorby reportu. Na rozdíl od ostatních metodologií se zaměřuje i na fyzickou bezpečnost – přístup k serverům a další infrastruktuře. [12]

Metodologie přistupuje k problému velice systematicky a zahrnuje i aktivity, které je nutné provést před i po samotném testování. Jedná se jak o přípravu na testování – dohodu s klientem, harmonogram, tak i prezentaci výsledků – již zmíněný report. Samotný proces penetračního testování je rozdělen do 7 fází, které detailně popisují jednotlivé kroky:

1. *Pre-engagement* – příprava na testování, definice rozsahu, metod a harmonogramu testování.
2. *Intelligence Gathering* – získávání informací o *oběti*, a to jak z otevřených zdrojů, tak z aktivního skenování i fyzického průzkumu.
3. *Threat Modeling* – vytvoření modelu hrozeb, které jsou reálné pro danou organizaci.
4. *Vulnerability Analysis* – samotné hledání zranitelností a chyb.
5. *Exploitation* – *útočník* se snaží využít nalezené zranitelnosti k útoku na organizaci.

6. *Post Exploitation* – využití přístupu, získaného v předchozí fázi, ke sběru dalších informací, eskalaci oprávnění a perzistenci.

7. *Reporting* – tvorba zprávy o průběhu celého testování. [12]

2.1.3 Open Source Security Testing Methodology Manual

Autorem této metodologie je Pete Herzog, spravována je organizací Institute for Security and Open Methodologies (ISECOM). [13]

OSSTMM se zabývá nejen bezpečnostním testováním, ale obsahuje i kapitoly věnující se bezpečnostním metrikám a důvěře v jejich výsledky. Dokument se zaměřuje i na testování informačních systémů, bezdrátových technologií, telekomunikace, lidský faktor nebo fyzický přístup – není tedy primárně zaměřen na testování informačních systémů nebo aplikací. [13]

2.1.4 NIST SP 800-115

Jedná o publikaci *Národního institutu pro standardy a technologii (NIST) Spojených států amerických*. Dokument se soustředí na bezpečnostní testování, na techniky, které se při něm používají a také na posuzování bezpečnosti. Kromě toho obsahuje také kapitolu o aktivitách po samotném testování: report o testování a návrh oprav chyb a zranitelností. Cílem je poskytnout komplexní informace k bezpečnostním testům a jejich provádění. Metodologie je pravidelně aktualizována, aby reflektovala nové trendy v informační bezpečnosti. Má také vyšší oficiální úroveň než ostatní metodologie, protože *NIST* je federální agenturou Spojených států amerických. [14]

2.1.5 Výběr metodologie

Pro testování aplikace Anketa ČVUT je z posuzovaných metodologií vybrána *OWASP Web Security Testing Guide*. Tato metodologie se totiž zaměřuje přímo na problematiku bezpečnosti webových aplikací a umožňuje se soustředit na podstatné záležitosti v provádění testu. Ostatní metodologie se oproti tomu zaměřují na obecnější téma testování daného informačního systému, což zahrnuje více případů než pouze testování webové aplikace.

2.2 Hodnocení závažnosti zranitelností

Hodnotit závažnost zranitelností je nutné především pro účely následujícího procesu jejich odstranění nebo zmírnění jejich potencionálních následků. Díky známé závažnosti je možné prioritizovat opravy těch kritických před těmi méně vážnými. Ohodnocení zranitelností je důležitou součástí procesu *Vulnerability*

Management, tedy správy zranitelností. Navíc, organizace také téměř nikdy nemají dostatek zdrojů pro opravu všech problémů, je proto nutné se soustředit především na ty, které mohou způsobit organizaci nějakou škodu. [15, 16]

Pro hodnocení závažnosti zranitelností je zvolen *Common Vulnerability Scoring System* – CVSS. Tento systém slouží k jednoduchému ohodnocení závažnosti na základě několika položek.

V nejzákladnější verzi – nomenklatuře *Base metrics* (nomenklatura popisuje, jaké metriky jsou v hodnocení použity) – spočívá v ohodnocení 11 položek. Jedná se o kontext prostředí (fyzický nebo vzdálený přístup), komplexitu útoku, předpoklady útoku (takové, které nedokáže útočník ovlivnit), uživatelská oprávnění útočníka, nutnost interakce uživatele. Poslední dvě kategorie pak tvoří dopad na důvěrnost, integritu a dostupnost, a to samostatně v kontextu zranitelného systému a systému, který na něj navazuje. V každém z těchto bodů je třeba zvolit z 2 až 4 možností, které jsou detailně popsány. [17]

Dalšími nomenklaturami jsou *Base and Environmental metrics*, *Base and Threat metrics* a *Base, Threat, Environmental metrics*. Tyto zahrnují navíc oproti *Base Metrics* hodnocení v oblastech prostředí aplikace, hodnocení hrozby (pravděpodobnosti, že bude útok proveden) nebo obou předchozích. [17]

Bodové skóre se pohybuje od 0 do 10, kde 10 značí nejvyšší závažnost. Způsob výpočtu tohoto skóre z jednotlivých položek hodnocení je detailně popsán v [17]. Bodové skóre lze převést na určitou kategorii závažnosti, počínaje *None* – žádná závažnost, konče *Critical* – tedy kritická závažnost. Převod ukazuje tabulka 2.1. [17]

Tabulka 2.1 Hodnocení závažnosti zranitelností [17]

CVSS Base Score	CVSS Severity Level
0	None
0,1–3,9	Low
4,0–6,9	Medium
7,0–8,9	High
9,0–10,0	Critical

Kapitola 3

Bezpečnostní analýza aplikace

V této kapitole je provedena bezpečnostní analýza aplikace Anketa ČVUT podle metodologie OWASP Web Security Testing Guide. Názvy částí odpovídají názvům kapitol dle WSTG. V rámci testu některých zranitelností je popsán i princip této zranitelnosti. Části, ve kterých je nalezena nějaká zranitelnost nebo chyba, jsou odkazovány na příslušnou část další kapitoly, kde je zranitelnost nebo chyba vyhodnocena podrobněji.

Je nutné zmínit, že testování je prováděno s přístupem k databázi i ke zdrojovému kódu aplikace. Jedná se tedy o *White box Testing*. Testování je prováděno ve vývojovém, nikoliv v produkčním prostředí; kromě případů, kdy provádění testu na vývojovém prostředí není relevantní. Tato informace je explicitně zmíněna u příslušné části testu spolu s důvodem tohoto rozhodnutí.

Podrobný popis zranitelností a také detaily ohledně toho, jak provádět daný test, lze nalézt v metodologie OWASP WSTG. [11].

Kompletní zdrojové kódy frontendu [5] a backendu [4], ze kterých je vycházeno při testování, lze nalézt v `frontend.zip` a `backend.zip` v `/attach/code/`.

3.1 Information Gathering

Tato sekce se zaměřuje na získávání informací o testovaném systému. Cílem je získat informace o aplikaci s pomocí vyhledávacích nástrojů, zjistit jaký webový server je využit a jaké technologie jsou používány v rámci testované aplikace. Dalším cílem je poznat princip fungování aplikace a její strukturu. Takto zjištěné informace mohou být využity v dalších částech testování, mohou však být také zneužity útočníkem.

3.1.1 Conduct Search Engine Discovery Reconnaissance for Information Leakage

Vyhledávací nástroje (např. *Google*, *Bing*, ...) neustále indexují dostupné webové stránky (pomocí odkazů z jiných stránek nebo pomocí *Sitemap* – souboru popisujícího strukturu webu). Je možné, že takto byly indexovány i stránky obsahující citlivé informace, které mohou být zneužity útočníkem. Cílem tohoto testu je takové informace vyhledat a posoudit citlivost jejich obsahu. [11]

Pomocí vyhledávání na *Google* byla nalezena veřejně přístupná stránka dokumentace – `http://anketa-vyvoj.cvut.cz/wiki/doc-info-admini`. Tato stránka obsahuje citlivé informace jako hesla a uživatelská jména do vývojového prostředí – tyto údaje se však jeví jako neaktuální, nepodařilo se je využít k přihlášení. V dokumentaci se dále zmiňuje, že data ve vývojovém a stage prostředí pochází z prostředí produkčního, což lze považovat za citlivou informaci, kterou mohou útočníci využít. Dále je zde návod, jak se přihlásit ve vývojovém prostředí jako kdokoliv jiný, což v kombinaci s používáním produkčních dat vytváří závažný problém – útočník má přístup k informacím, které jsou dostupné pouze pro studenty nebo vyučující ČVUT a mohou pomoci s prolomením anonymity.

Vývojové a stage prostředí je navíc veřejně přístupné. Kdokoliv tedy může prohlížet komentáře učitelů a výsledky ankety bez jakéhokoliv oprávnění. Dále je možné sledovat vyplněnost ankety jednotlivými studenty. Musím však zmínit, že aktualizace dat ve vývojovém prostředí je nepravidelná a útočník ji nemůže ovlivnit.

Pomocí dalších vyhledávacích nástrojů (*Bing*, *DuckDuckGo*) nebyly nalezeny další citlivé informace.

Přístupnost vývojového a stage prostředí je spolu s použitím produkčních dat vyhodnocena v sekci 4.8.

3.1.2 Fingerprint Web Server

Obrázek 3.1 Použití nástroje Nikto pro zjištění webserveru (celý výpis k dispozici v `/attach/tool-output/nikto.txt`)

```
$ nikto -h anketa-vyvoj.cvut.cz
...
+ Server: Apache/2.4.57 (Debian) SVN/1.14.2 mod_jk/1.2.48
mod_python/3.5.0+git20211031.e6458ec Python/3.11.2
```

Pro skenování serveru byl použit nástroj *Nikto*. Z výstupu lze určit, že se jedná o server *Apache/2.4.57*. Tato verze obsahuje středně závažnou zranitel-

nost *CVE-2023-45802*, která je opravena v další verzi. [18] Verze 2.4.57 byla vydána již v dubnu 2023. [19]

3.1.3 Review Webserver Metafiles for Information Leakage

Na webu se nenachází soubory `robots.txt`, `sitemap.xml` ani další podobné. Pomocí nástroje *DirBuster* také nebylo nic citlivého nenalezeno.

3.1.4 Enumerate Applications on Webserver

Tato část testu je prováděna v produkčním prostředí. Ve vývojovém prostředí je otevřeno více portů.

Použitím *nmap* nebyly zjištěny žádné neočekávané informace. Na serveru jsou otevřeny porty pro HTTP a HTTPS. Při pokusu o zjištění stavu všech portů byla komunikace se serverem zablokována (nevracely se žádné odpovědi). Toto je nejspíše způsobeno firewallem.

Obrázek 3.2 Výstup nástroje nmap (celý výstup lze nalézt v `/attach/tool-output/nmap.txt`)

```
$ nmap anketa.is.cvut.cz
...
PORT    STATE SERVICE
80/tcp  open  http
443/tcp open  https
```

Pomocí *Reverse IP Lookup* bylo dále zjištěno, že k IP adrese tohoto serveru nejsou asociovány žádné jiné DNS záznamy (toto je patrné z obrázku 3.7).

3.1.5 Review Webpage Content for Information Leakage

HTML zdrojový kód neobsahuje žádné citlivé informace v HTML komentářích ani v *META* tazích. Ani ve zdrojovém kódu JavaScriptu (JS) se mi nepodařilo najít jakékoliv citlivé informace jako např. uživatelská jména, hesla nebo API klíče.

3.1.6 Identify Application Entry Points

GET požadavky, které byly v aplikaci nalezeny pomocí sledování komunikace se serverem v nástroji *BurpSuite*, jsou uvedeny v tabulce 3.1 pro FE a v ta-

bulce 3.2 pro BE. Téměř všechny požadavky na FE pouze získávají komponenty webové stránky (kromě `/` s parametrem `token`, který nastavuje token z URL jako Session Cookie a `/set-password` s parametrem `token`, který je určen pro nastavení hesla uživatele). *GET* požadavky na BE jsou určeny pro získaní samotných dat (ankety, výsledky anket, …). Webová stránka `/set-password` je na FE chybně implementována (odesílá data na neexistující endpoint `/login/create` místo `/user`). Tato chyba je vyhodnocena v sekci 4.12.

Tabulka 3.1 URL adresy GET požadavků na FE a jejich parametry (celé požadavky jsou v `/attach/tool-output/http-get-fe.txt`)

relativní URL	parametry
`/`	-
`/`	`token`
`/assets/{asset-name}`	-
`/set-password`	`token`
`/results/semesters/{…}/surveys/1/courses/{…}`	-
`/results/semesters/{…}/surveys/{…}`	-
`/results/semesters/{…}/surveys/1/teachers/{…}`	-

Tabulka 3.2 URL adresy nalezených GET požadavků na BE a jejich parametry (celé znění nalezených požadavků je v `/attach/tool-output/http-get-be.txt`)

relativní URL	parametry
`/courses`	-
`/teachers`	-
`/surveys`	-
`/reactions`	-
`/publicationOptions`	-
`/questions`	-
`/auth/test/{username}`	-
`/noticeBoard/landingPage/{lang}`	-
`/noticeBoard/homePage/{lang}`	-
`/results/semesters`	-
`/results/semesters/{…}/surveys/{…}`	-
`/results/semesters/{…}/surveys/1/courses/{…}`	-
`/results/semesters/{…}/surveys/1/teachers/{…}`	-

Místa, kde je využito metody *POST*, jsou uvedena v tabulce 3.3. Ta, která slouží k autentizaci, mají jako parametry pouze uživatelské jméno a heslo (kromě SSO). Požadavek sloužící k odeslání odpovědí v anketě obsahuje jako parametry identifikátory konkrétního předmětu a ankety a také odpovědi na anketní otázky. Za parametry, které uživatel neovlivňuje pomocí svých odpovědí, lze považovat identifikátory předmětu, semestru, ankety, …

Další URL byly nalezeny v souboru `anketa-api-1.0.0.yml` (lze nalézt v `/attach/code/anketa-api-1.0.0.yml`). Musím však zmínit, že ne všechny jsou v aplikaci používány.

Tabulka 3.3 URL adresy POST požadavků na BE a jejich parametry (celé požadavky jsou v `/attach/tool-output/http-post-be.txt`)

relativní URL	parametry
`/surveys`	`courseId`, `surveyId`, `semesterCode`, `lang`, …
`/slack`	`fallback`, `color`, `title`, `title_link`, …
`/login/externalLdap`	`userId`, `password`
`/login/external`	`userId`, `password`
`/user`	`password`, `token`

Dále byly prozkoumány i odpovědi od serveru a jejich obsah. Při neúspěšném přihlášení je vráceno `404 Not Found`, pro neexistující uživatele je při dotazu na BE vráceno `401 Unauthorized`. Při testovací autentizaci (dostupné pouze ve vývojovém prostředí) je vráceno `301 Moved Permanently`.

3.1.7 Map Execution Paths Through Application

Tento bod zahrnuje zjišťování informací o tom, jak aplikace funguje a jaká je její struktura. Jelikož je test prováděn s přístupem ke zdrojovému kódu, nebude tato část realizována, neboť by v rámci ní nebyly získány žádné informace.

3.1.8 Fingerprint Web Application Framework

V *HTTP* odpovědích se nenachází žádné informace, které by vedly k identifikaci frameworku webové aplikace (`X-Powered-By` se v odpovědi nenachází). Pomocí názvu Cookies, které je `Authorization`, také nelze určit o jaký framework se jedná.

Z *HTML* zdrojového kódu lze určit, že aplikace používá JavaScript a CSS. Aplikace také nemá žádné specifické adresáře (např. `wp-admin` pro *WordPress*), které by jednoznačně určily framework aplikace. Framework se nepodaří určit ani pomocí přípon souborů nebo chybových hlášek.

Dále byl pro zjištění těchto informací použit nástroj *WhatWeb*, ale ani pomocí něj nebyly nezískány informace o použitém frameworku (výpis nástroje je v obrázku 3.3). Využité technologie lze však nalézt na stránce `http://anketa-vyvoj.cvut.cz/dev/about`.

3.1.9 Fingerprint Web Application

Tato část je dle metodologie přesunuta do sekce 3.1.8.

Obrázek 3.3 Výstup skenu nástrojem WhatWeb (celý výstup lze nalézt v `attach/tool-output/whatweb.txt`)

```
$ whatweb anketa-vyvoj.cvut.cz/dev
...
http://anketa-vyvoj.cvut.cz/dev/ [200 OK] Apache[2.4.57]
[mod_jk/1.2.48,mod_python/3.5.0+git20211031.e6458ec],
Country[CZECH REPUBLIC][CZ], HTML5, HTTPServer[Debian Linux]
[Apache/2.4.57 (Debian) SVN/1.14.2 mod_jk/1.2.48 mod_python/
3.5.0+git20211031.e6458ec Python/3.11.2], IP[147.32.3.141],
Python[3.11.2], SVN[1.14.2], Script[module],
Title[Anketa ČVUT]
```

3.1.10 Map Application Architecture

Architektura aplikace je popsána v sekci 1.5. Tato kapitola se jejímu zkoumání věnovat nebude.

3.2 Configuration and Deployment Management Testing

Tato sekce se zabývá testování konfigurace aplikace a webového serveru. Součástí je pátrání po citlivých souborech, např. zálohách nebo nereferencovaných souborech, testování oprávnění k těmto souborům. Dále je také testováno, které HTTP metody jsou akceptovány webserverem, nebo zda je vynucena komunikace přes HTTPS.

3.2.1 Test Network Infrastructure Configuration

Jednou z nejvíce exponovaných komponent je samotný web server – zjištění jeho verze a obsažených zranitelností je popsáno v sekci 3.1.2.

3.2.2 Test Application Platform Configuration

Nebyly nalezeny žádné soubory ani složky, které by neměly být přístupné a mohly prozrazovat detaily o konfiguraci serveru (o hledání souborů pojednává sekce 3.2.4). Ani v HTML zdrojovém kódu se nenachází žádné takové informace.

3.2.3 Test File Extensions Handling for Sensitive Information

Přípony souborů, které se nachází v odpovědích od serveru, jsou `js`, `png`, `css`, `ttf` a `ico`. Žádná z těchto přípon mi nepřinesla nové informace. Další přípona souborů – `jsx` – byla nalezena nástrojem *gospider* (výstup nástroje je v obrázku 3.4). Tato přípona je spojena s technologií *React*, ale v současnosti je používána i jinými.

3.2.4 Review Old Backup and Unreferenced Files for Sensitive Information

Pro tento test byl použit nástroj *gospider* (výstup nástroje je v obrázku 3.4), pomocí něj nebyly nalezeny žádné přístupné citlivé soubory. Dále byl využit nástroj *dirb*; tento nástroj nalezl již známé adresy `/admin` (administrátorská aplikace, kterou není testována), `/assets` (adresář s obrázky, skripty, …) – celý výstup tohoto nástroje se nachází v texttt/attach/tool-output/dirb.txt.

Obrázek 3.4 Výstup nástroje gospider (kompletní výpis lze nalézt v `/attach/tool-output/gospider.txt`)

```
$ gospider -s http://anketa-vyvoj.cvut.cz/dev -v
...
[linkfinder] - [from: http://anketa-vyvoj.cvut.cz/dev/
    assets/index-D2OTjSPd.js] - ./vendor-Cruk6umk.js
[linkfinder] - [from: http://anketa-vyvoj.cvut.cz/dev/
    assets/index-D2OTjSPd.js] - /builds/anketa-cvut-v3/
    anketa/src/Components/LocaleSwitcher.jsx
...
[linkfinder] - [from: http://anketa-vyvoj.cvut.cz/dev/
    assets/index-D2OTjSPd.js] - /login/create
...
[linkfinder] - [from: http://anketa-vyvoj.cvut.cz/dev/
    assets/index-D2OTjSPd.js] - /set-password
...
```

3.2.5 Enumerate Infrastructure and Application Admin Interfaces

Rozhraní pro administraci se nachází na `/admin`. Rozhraní je řešeno jako samostatná aplikace, jejímuž testování se tato práce nevěnuje.

3.2.6 Test HTTP Methods

Podporované HTTP metody byly zjištěny pomocí *nmap*, výstup nástroje je v obrázku 3.5.

Obrázek 3.5 HTTP metody zjištěné pomocí nmap (kompletní výpis lze nalézt v `attach/tool-output/nmap-http-methods.txt`)

```
$ nmap -p 80 --script http-methods --script-args http-methods\
    .url-path='/dev' anketa-vyvoj.cvut.cz
...
80/tcp open  http
| http-methods:
|   Supported Methods: GET HEAD POST OPTIONS
|_  Path tested: /dev
```

Metoda *PUT* není povolena, server vrací `405 Method Not Allowed`. Metoda *HEAD*, *TRACE* ani *DELETE* se mi nepodařila zneužít – server u všech vrací `400 Bad Request`.

3.2.7 Test HTTP Strict Transport Security

Tato část testu je prováděna v produkčním prostředí. Vývojové prostředí nepodporuje HTTPS. Musí být zmíněno, že v produkčním prostředí zatím není nová verze aplikace.

Odpověď od serveru obsahuje HSTS (odpověď je v obrázku 3.6), použití HTTPS je tímto vynuceno.

Obrázek 3.6 HTTP Strict Transport Security v odpovědi od serveru (kompletní výpis lze nalézt v `attach/tool-output/hsts.txt`)

```
Strict-Transport-Security: max-age=63072000; includeSubDomains
```

3.2.8 Test RIA Cross Domain Policy

V aplikaci se nepoužívají soubory Cross Domain Policy (`Crossdomain.xml` nebo `Clientaccesspolicy.xml`) – žádné takové soubory nebyly při hledání souborů nalezeny (o hledání souborů pojednávají sekce 3.2.3 a 3.2.4).

3.2.9 Test File Permission

Není možné přímo otestovat oprávnění k souborům na webovém serveru – není k dispozici přístup, který by umožňoval tyto informace získat. Nicméně při testování v sekci 3.2.4 nebyl nalezen žádný soubor, který by neměl být přístupný.

3.2.10 Test for Subdomain Takeover

Tento test je prováděn v produkčním prostředí. Chybná konfigurace DNS ve vývojovém prostředí by neměla tak závažný dopad jako potencionálně chybná konfigurace produkčního prostředí.

Aplikace Anketa ČVUT je umístěna na doméně `anketa.is.cvut.cz`; domény `anketa.cvut.cz` a `www.anketa.cvut.cz` jsou na ní odkázány v rámci DNS (pomocí *CNAME* záznamu, jak je patrné z obrázku 3.7). DNS je nastaveno správně – domény jsou pod kontrolou provozovatele aplikace. Neexistují ani další subdomény patřící pod tyto domény. Není tedy možné tuto zranitelnost zneužít.

Obrázek 3.7 DNS záznamy pro doménu aplikace získané pomocí DNS Lookup Tool (`https://mxtoolbox.com/DNSLookup.aspx`)

```
CNAME   www.anketa.cvut.cz      anketa.cvut.cz
CNAME   anketa.cvut.cz          anketa.is.cvut.cz
A       anketa.is.cvut.cz       147.32.3.33
```

3.2.11 Test Cloud Storage

Aplikace nevyužívá cloudové úložiště; veškerá data jsou uložena v databázi na webovém serveru.

3.3 Identity Management Testing

Tato sekce se zaobírá testováním procesů souvisejících s identitou uživatele. Mezi ně patří role uživatelů, proces registrace uživatelů, tvorba účtů a jejich správa, enumerace uživatelských účtů nebo pravidla a omezení pro uživatelská jména.

3.3.1 Test Role Definitions

Role jsou popsány v sekci 1.2, informace o rolích jsou veřejně dostupné v [7]. V každé funkci na BE, která nějaká data čte nebo zapisuje, je správně pro-

vedeno ověření, jestli má uživatel k dané akci oprávnění (toto je testováno v sekci 3.5.2).

3.3.2 Test User Registration Process

Registrace uživatele není možná prostřednictvím této aplikace. Registrace interních uživatelů (jejich přihlášení neprobíhá pomocí LDAP ani SSO) probíhá následovně (v nové verzi, která je zatím pouze ve vývojovém prostředí, je dostupná tato funkce pouze na BE; na FE není implementována):

1. žádost o tvorbu účtu (mimo aplikaci),
2. tvorba náhodného řetězce (tokenu) a jeho vložení do databáze (mimo aplikaci),
3. vytvoření hesla uživatelem (pomocí URL s tokenem).

Jednorázový token je generován pomocí `pwgen -A 25 1` – jeho délka je dostatečná. Jeho platnost však není časově omezená – tento fakt usnadňuje útočníkovi jeho prolomení.

Zranitelnost vyplývající z neomezené platnosti těchto tokenů je vyhodnocena v sekci 4.7.

Samotná existence tohoto postupu (tedy registrace uživatelů bez jejich ČVUT identity) je z hlediska bezpečnosti problematická – vytváří prostor pro chyby, dále je nutná implementace tohoto postupu do samotné aplikace. Tento způsob také není v souladu s politikou ČVUT (přihlášení do veškerých IS pomocí hesla ČVUT).

Existence tohoto postupu je vyhodnocena v sekci 4.14.

3.3.3 Test Account Provisioning Process

V aplikaci není možné vytvářet a spravovat uživatelské účty.

3.3.4 Testing for Account Enumeration and Guessable User Account

Pomocí přihlašovacího formuláře není možné zjistit platná uživatelská jména – v případě špatného uživatelského jména nebo hesla je vrácena hláška *Špatné heslo nebo username* (BE vrací `unauthorized`). Lze toho ale dosáhnout pomocí sledování času odpovědi od serveru, o čemž pojednává sekce 3.10.4.

3.3.5 Testing for Weak or Unenforced Username Policy

Naprostá většina uživatelů se přihlašuje pomocí SSO (jejich uživatelské jméno se shoduje s Identitou ČVUT). Z databáze se mi podařilo zjistit, že existují

Výpis kódu 3.1 Zpracování autentizace na BE (plná verze kódu se nachází v `/attach/code/authorizeExternalUser.kt`) [4]

```
override suspend fun authorizeExternalUser(
    authPostRequest: AuthPostRequest,
): ResponseEntity<AuthPostResponse> {
    val user =
        authRepository
        .getExternalRegistration(authPostRequest.userId)
        ?: return unauthorized<AuthPostResponse>()

    val validPassword = HashUtils.validatePassword(...)
    return if (validPassword) {
        ok {
            val token = tokenService
                .provideToken(user.id.toString())
            AuthPostResponse(token)
        }
    } else {
        unauthorized<AuthPostResponse>()
    }
}
```

uživatelé (jedná se především o vyučující) s kratším uživatelským jménem než 8 znaků (standardní délka používaná v účtech ČVUT) – tato uživatelská jména jsou odhadnutelná podobně jako jména používaná v účtech ČVUT (obsahují fragmenty jména a příjmení uživatele).

3.4 Authentication Testing

Tato sekce se zabývá testování autentizace – ověřením identity uživatele. Testy jsou zaměřeny na způsob, jakým se uživatelské údaje přenáší, na použití výchozích uživatelských údajů nebo Password Policy. Dále jsou testovány podpůrné funkcionality, mezi které patří: zapamatování přihlášení prohlížečem, Lock Out mechanismus, reset hesla, bezpečnostní otázky a další.

3.4.1 Testing for Credentials Transported over an Encrypted Channel

Tato část testu je prováděna v produkčním prostředí. Vývojové prostředí nepodporuje HTTPS. Přihlašovací údaje jsou přenášeny pouze šifrovaně; vynucení HTTPS je pak zajištěno pomocí HSTS, což je zmíněno v sekci 3.2.7. Testování atributů Session Cookies je prováděno v sekci 3.6.2.

3.4.2 Testing for Default Credentials

V databázi se mi nepodařilo najít žádné přihlašovací údaje, které by vypadaly jako výchozí přihlašovací údaje; nenachází se zde ani účty jako `admin`, `root` nebo podobné. V samotné autentizaci také není žádný problém (`if (name == 'admin')`, uživatelská jména ve zdrojovém kódu, …), toto je patrné z výpisu kódu 3.1.

3.4.3 Testing for Weak Lock Out Mechanism

Mechanismus Lock Out slouží k dočasnému zablokování uživatelského účtu např. z důvodu nadměrného počtu neúspěšných pokusů o přihlášení. [11]

Částí aplikace, která by měla být chráněna pomocí tohoto mechanismu, je především autentizace uživatele. Zaměřím se pouze na LDAP a interní přihlášení. SSO autentizace probíhá mimo samotnou aplikaci a její testování není součástí této práce.

U LDAP ani interního přihlášení se nenachází žádný mechanismus, který by zabraňoval opakovanému zkoušení přihlašovacích údajů (při autentizaci se ověřuje pouze uživatelské jméno a heslo, jak je patrné z ukázky kódu 3.1). Toto umožňuje útočníkovi zkoušet libovolné množství přihlašovacích údajů (vše je samozřejmě omezeno výkonem serveru a propustností sítě).

Tato zranitelnost je vyhodnocena v sekci 4.5.

3.4.4 Testing for Bypassing Authentication Schema

Tento test zkoumá, zda je možné obejít autentizaci a přistoupit ke zdrojům, které jsou přístupné pouze po autentizaci. [11]

Nepodařilo se mi najít stránku, u které by měla být vyžadována autentizace, a přesto bylo možné k této stránce přistoupit. Vždy dojde k přesměrování na přihlášení. Ani modifikace parametrů v URL nepomůže k přístupu na stránku, která by přístupná být neměla. Samotné zpracování autentizace také není zranitelné takovým způsobem, aby jej bylo možné obejít (zpracování autentizace je v ukázce kódu 3.1).

3.4.5 Testing for Vulnerable Remember Password

Zranitelnost spočívá v nevhodném mechanismu zapamatování si uživatele, např. uložením jeho přihlašovacích údajů v úložišti prohlížeče. [11]

V aplikaci není možnost zvolit si zapamatování uživatele. Není tedy možné zde nalézt tuto zranitelnost.

3.4.6 Testing for Browser Cache Weaknesses

Použitím tlačítka zpět (v prohlížeči po odhlášení) se nelze dostat k informacím dostupným pouze po přihlášení; uživatel je vždy přesměrován na přihlašovací stránku. V samotné cache taktéž nebyly nalezeny žádné citlivé údaje (např. Session ID).

3.4.7 Testing for Weak Password Policy

V aplikaci není možná změna hesel. Je zde dostupná tvorba hesel pro interní uživatele (více o tvorbě účtů a hesel v sekci 3.3.2). Jediným požadavkem na hesla uživatelů je délka 6 znaků. Doporučovaná délka je alespoň 12 nebo 14 znaků. [20, 21]

Tato problematika je vyhodnocena v sekci 4.6.

3.4.8 Testing for Weak Security Question Answer

V aplikaci se žádné bezpečností otázky nepoužívají, tato zranitelnost zde tedy být nemůže.

3.4.9 Testing for Weak Password Change or Reset Functionalities

Aplikace neumožňuje změnu ani reset hesla. Naprostá většina uživatelů používá pro přihlášení SSO. Ani ti, kteří se přihlašují interně (tedy pouze pomocí této aplikace, bez SSO nebo LDAP), nemají možnost si změnit heslo pomocí této aplikace.

3.4.10 Testing for Weaker Authentication in Alternative Channel

Aplikace nemá jiné alternativy (např. mobilní verzi), kde by bylo možné nalézt odlišné způsoby autentizace a podobných funkcionalit. Přihlášení je možné pomocí SSO (doporučený způsob přihlašování) – testování této služby není zaměřením práce, nicméně je možné předpokládat, že je z hlediska bezpečnosti řešená lépe než interní autentizace.

Další možností pro uživatele s interním přihlášením (tedy bez SSO) je kontaktování správce aplikace (použití tohoto postupu pro tvorbu účtu je popsáno v sekci 3.3.2).

3.5 Authorization Testing

Tato sekce se zaměřuje na testování autorizace, tedy oprávnění uživatele provádět danou akci. Testy se zaměřují např. na eskalaci oprávnění, na přístupnost zdrojů bez kontroly autorizace nebo na Directory Traversal.

3.5.1 Testing Directory Traversal File Include

Tato zranitelnost spočívá ve špatné validaci uživatelského vstupu – konkrétně názvu souboru (např. `../file.txt`). Tím je docíleno uložení nebo přečtení z nepřístupného adresáře.

V této aplikace není možnost nahrávat ani číst soubory, zneužití této zranitelnosti tedy nehrozí.

3.5.2 Testing for Bypassing Authorization Schema

Autorizace požadavků pomocí dotazu do databáze (např. učitel může komentovat předmět, který vyučuje – dochází ke kontrole tohoto faktu v databázi). Fragment jedné z funkcí (pouze hlavička), které toto ověřují, se nachází ve výpisu kódu 3.2. Ověření se nachází v každé funkci, která získává nebo upravuje nějaká data. Zjištění identity uživatele je provedeno pomocí Session Cookie – JWT tokenu (více o testování JWT v sekci 3.6.1).

Výpis kódu 3.2 Fragment funkce pro ověření autorizace na BE (celá funkce včetně databázového dotazu se nachází v `/attach/code/hasAccessToCourseReaction.kt`) [4]

```
@Query(...)
suspend fun hasAccessToCourseReaction(
    @Param("semesterCode") semesterCode: String,
    @Param("surveyId") surveyId: Long,
    @Param("teacherId") teacherId: Long,
    @Param("courseId") courseId: Long,
    @Param("username") username: String,
): Boolean
```

3.5.3 Testing for Privilege Escalation

Oprávnění uživatele (přesněji autorizace provádět dané akce) je kontrolováno pomocí uživatelského jména (to je uloženo a digitálně podepsáno v Session Cookie) a záznamů v databázi (např. jestli student má tento předmět zapsán v daném semestru). Jako jedna z možností eskalace privilegií se jeví podvržení podpisu v Session Cookie, realizovatelnost tohoto způsobu se však není prokázána.

3.5.4 Testing for Insecure Direct Object References

Uživatelem upravitelný vstup je použit například jako název souboru nebo index do databáze; je tak možné přistupovat ke zdroji, ke kterému nemá uživatel oprávnění (např. editace uživatele pomocí `/edit?user={my-username}`). [11]

V aplikaci nebyla nalezena žádná situace, která by byla řešena tímto způsobem (tedy parametrem nebo URL, u kterého není počítáno s tím, že by jej mohl uživatel změnit). Zranitelnost IDOR se zde nenachází.

3.6 Session Management Testing

Tato sekce se zabývá testováním Session Managementu – tedy správy uživatelských relací. Součástí těchto testů je analýza použitých technologií, identifikátorů Session, náhodnosti hodnot v těchto identifikátorech. Dále je testováno také nastavení parametrů Cookies, odhlášení uživatele, obrana proti získání Session ID útočníkem a dalším útokům.

3.6.1 Testing for Session Management Schema

Ke správě Sessions jsou používány JWT tokeny; ty v sobě obsahují i informace o časové platnosti. Tyto tokeny jsou digitálně podepsány, lze tedy ověřit, že byly vytvořeny daným subjektem s daným privátním klíčem (v tomto případě serverem). Formát tokenu je `{header}.{payload}.{signature}` zakódovaný v Base64 (celé Cookie z testovacího prostředí se nachází v obrázku 3.8). Tvorba tokenu se nachází v ukázce kódu 3.3. [22]

Na straně BE není nijak kontrolována časová expirace. Je tedy možné použít jakkoliv starý JWT jako Session Cookie. Absence této kontroly je zřejmá v ukázce kódu 3.4, kde dochází pouze k verifikaci podpisu a kontrole, zda má daný uživatel povolen přístup.

Absence kontroly platnosti i problematičnost samotného designu Session Managementu je vyhodnocena v sekci 4.1.

Nastavení atributů Session Cookies je věnována sekce 3.6.2.

Obrázek 3.8 JWT token zakódovaný v Base64 a dekódovaný token

```
Base64 encoded:
eyJhbGciOiJIUzUxMiJ9.
eyJqdGkiOiJjMjg2NWYwOS0wMjZiLTRjOTctODg0MS1jOGVmOGMwY
2RiMzUiLCJpc3MiOiJhbmtldGEtZGV2ZWwiLCJzdWIiOiJ6YWJvcn
ZvaiIsImlhdCI6MTcxMjYwNzA4MSwiZXhwIjoxNzEyNzc5ODgxfQ.
bNz1OSb_NPqEZ6-og1rNwlOAWHtUGI_o3jl-ii-wCzZDIBF9pjZN6
48I2wQxRb8izj9jqP82x9lqUkec2GQGxg

Decoded header and payload:
header:
{"alg":"HS512"}
payload:
{"jti":"c2865f09-026b-4c97-8841-c8ef8c0cdb35",
"iss":"anketa-devel","sub":"zaborvoj","iat":1712607081,
"exp":1712779881}
```

Výpis kódu 3.3 Tvorba JWT tokenu na BE (celý kód je k dispozici v `/attach/code/SecurityTokenServiceImpl.kt`) [4]

```
override suspend fun provideToken(subject: String): String {
    val now = Date()
    val validity =
        Date(now.time + jwtProperties.validityInMillis)
    return Jwts.builder()
        .id(UUID.randomUUID().toString())
        .issuer(jwtProperties.issuer)
        .subject(subject)
        .issuedAt(now)
        .expiration(validity)
        .signWith(jwtProperties.secretKey)
        .compact()
}
```

3.6.2 Testing for Cookies Attributes

Atribut `HttpOnly` je nastaven na `false`, neboť je vyžadováno využívání Cookies pomocí JS (přístup JS ke Cookies je patrný ve výpisu kódu 3.5). Povolení přístupu ke Cookies pro JS zvyšuje úspěšnost získání těchto Cookies potenci-

Výpis kódu 3.4 Kontrola JWT tokenu na BE (celý kód je k dispozici v `/attach/code/SecurityTokenServiceImpl.kt`) [4]

```
override suspend fun getAuthorizedUser(
    credential: String,
): LoggedUser {
    val tokenSubject = runCatching {
        Jwts.parser()
            .verifyWith(jwtProperties.secretKey)
            .build()
            .parseSignedClaims(credential)
            .payload
            .subject
    }
    return if (tokenSubject.isSuccess) {
        if (allowedUserRepository.hasAccess(...)) {
            RealLoggedUser(...)
        } else {
            logger.warn("User {} is not allowed ...", ...)
            AnonymousLoggedUser
        }
    } else {
        logger.warn("Invalid JWT token ...", ...)
        return AnonymousLoggedUser
    }
}
```

onálním útočníkovým kódem. Jedná se především o JavaScript a zranitelnost XSS (útočník získá Cookies pomocí `document.cookie` v JS).

Atribut `Secure` je nastaven na `false` (parametr je takto nastaven i v produkčním prostředí se starou verzí aplikace, které podporuje HTTPS); toto nastavení nezabraňuje prohlížeči odesílat Cookies přes nešifrovaný protokol HTTP. Atribut `Domain` je nastaven správně na doménu aplikace. Atribut `Path` není nastaven správně; jeho hodnota je `/`, což umožňuje odesílání Cookies i jiným aplikacím než té na `/dev` – zde je umístěna vývojová verze aplikace. Tuto chybu nelze pozorovat v produkční verzi, neboť ta běží na doméně samostatně. Parametr `Expires` u Session Cookies nastavuje platnost po dobu 2 dnů od vytvoření (samotná platnost ale není správně kontrolována na BE, jak je popsáno v sekci 3.6.1). Parametr `SameSite` je nastaven na `None`; toto lze zneužít pro Cross Site Request Forgery (CSRF), tomu je věnována sekce 3.6.5. Parametry Session Cookie ve vývojovém prostředí jsou zobrazeny na obrázku 3.9.

Chybné nastavení těchto parametrů je vyhodnoceno v sekci 4.2.

Obrázek 3.9 Parametry Session Cookie na webu bez HTTPS

```
Authorization:"eyJhbGciOiJIUzUxMiJ9.eyJqdGkiOiJ..."
    Created:            "Mon, 08 Apr 2024 20:11:20 GMT"
    Domain:             "anketa-vyvoj.cvut.cz"
    Expires / Max-Age:  "Wed, 10 Apr 2024 20:11:20 GMT"
    HostOnly:           true
    HttpOnly:           false
    Last Accessed:      "Mon, 08 Apr 2024 20:11:20 GMT"
    Path:               "/"
    SameSite:           "None"
    Secure:             false
    Size:               279
```

3.6.3 Testing for Session Fixation

Tato zranitelnost spočívá v použití stejných atributů Session před a po autentizaci. Útočník vnutí oběti svoje Session atributy a oběť se následně autentizuje; útočník se pak vydává za autentizovanou oběť. [11]

V aplikaci se nepoužívají žádné identifikátory Session před autentizací uživatele; nehrozí tedy zneužití těchto údajů.

3.6.4 Testing for Exposed Session Variables

Posíláním Session ID nebo Session Cookies nešifrovaně vzniká riziko jeho získání a zneužití útočníkem. [11]

Atributy Session (v tomto případě Session Cookies) se posílají v hlavičce požadavku ve formě parametru `Authorization`. Jako ochrana proti odeslání přes HTTP a následnému získání těchto údajů útočníkem by měl sloužit parametr `Secure` u Session Cookies – ten však není nastaven, jak je zmíněno v sekci 3.6.2. Samotné odeslání Session Cookie neproběhne přes HTTP, neboť je nastaven HSTS (více informací je v sekci 3.2.7); toto řešení ale není dostatečné.

3.6.5 Testing for Cross Site Request Forgery

Kvůli špatnému nastavení atributu `SameSite` (a to konkrétně na hodnotu `None`, o čemž pojednává sekce 3.6.2) je tato aplikace náchylná k provedení Cross Site Request Forgery. [23]

3.6.6 Testing for Logout Functionality

Odhlášení nezneplatňuje danou Session – použití Session Cookies je možné i po kliknutí na tlačítko *Odhlásit se*. Odhlášení totiž pouze vymaže Cookies z paměti prohlížeče; jejich samotné hodnoty jsou však i nadále platné a je možné pomocí nich provádět validní požadavky.

V ukázce kódu 3.5 je vidět, že probíhá pouze odstranění Cookie; žádný požadavek na zneplatnění se na server neodesílá. Z principu použití JWT není zneplatnění ani možné, neboť JWT jsou pouze digitálně podepsaná data, která se na serveru neukládají (více detailů je v sekci 3.6.1). [22]

Zranitelnost spočívající v nemožnosti zneplatnění dané Session je vyhodnocena v sekci 4.1.

Výpis kódu 3.5 Odhlášení na FE [5]

```
handleLogout = () => {
    cookie.remove('Authorizaton', { path: '/' });
    this.props.history.push('/');
    this.props.logout();
    window.location.reload();
};
```

3.6.7 Testing Session Timeout

V aplikaci se nenachází žádný Timeout po určité době neaktivity. Platnost Session Cookies v rámci prohlížeče je 2 dny. O samotném odhlášení uživatele pojednává sekce 3.6.6.

3.6.8 Testing for Session Puzzling

Pro správu Session se používají Cookies, konkrétně JWT tokeny (podrobněji v sekci 3.6.1). Jejich součástí je i náhodně vygenerovaná hodnota, která zabraňuje opakované tvorbě stejného tokenu.

Nové tokeny pro Session se vytváří jen a pouze při autentizaci uživatele; k jejich tvorbě je vždy nutné jméno i heslo (pro interní a LDAP uživatele; pro SSO uživatele jsou tyto údaje také nutné, zadávají se však v rámci SSO). Není zde tedy místo pro zranitelnost typu Session Puzzling.

3.6.9 Testing for Session Hijacking

Problém, který může vést k Session Hijacking je špatně nastavený atribut `Secure` nebo hodnota atributu `HttpOnly` (oboje popsáno detailněji v sekci 3.6.2).

Jiné komponenty aplikace, které by byly potencionálně zranitelné, se mi nalézt nepodařilo.

3.7 Input Validation Testing

Tato sekce se zabývá testováním validace uživatelských vstupů. Nesprávná kontrola těchto vstupů může vést ke Cross Site Scripting, SQL Injection, XML Injection, Code Injection, Command Injection, Cross Site Request Forgery nebo dalším zranitelnostem – na každou z nich je zaměřen konkrétní test z této sekce.

3.7.1 Testing for Reflected Cross Site Scripting

Reflected XSS je typ útoku, při kterém útočník vkládá škodlivý kód do URL adresy nebo formuláře na webové stránce a tento kód je poté vykonán v prohlížeči uživatele, čímž umožňuje útočníkovi např. získat Session Cookies. [11]

Nepodařilo se mi nalézt žádný uživatelský vstup, který by se po odeslání požadavku následně zobrazoval uživateli, a to ani jako část URL. Při nesprávném požadavku se samotný požadavek ani jeho část uživateli také nezobrazuje (např. nevalidní kód semestru, neexistující stránka, …).

Ve vývojovém prostředí se mi podařilo najít (při testování této kapitoly) špatnou validaci uživatelských vstupů (nejedná se o XSS) v přihlašovacím formuláři (ve vývojovém prostředí je možné se přihlásit jako kdokoliv; dokonce i jako neexistující uživatel). Zatímco použití uživatelského jména `nonExistingUser` funguje (proběhne autentizace, jen nejsou uživatelská data), tak použití `slash/User` vrací `404 Not Found`. Vstup tedy není validován ani zakódován, a je vkládán přímo do URL `/auth/test/{username}` (toto je dostupné pouze ve vývojovém prostředí) – chyba ve zpracování tohoto vstupu je zřejmá z ukázky kódu 3.6.

Výpis kódu 3.6 Autentizace jako libovolný uživatel ve vývojovém prostředí [4]

```
@GetMapping("/auth/test/{asUserName}")
@Profile("local", "devel", "stage")
suspend fun authorizeUser(@PathVariable("asUserName")
    userName: String):
ResponseEntity<Unit> {
    val token = tokenService.provideToken(userName)
    val location = frontendProperties.tokenHandler.format(token)
    return ResponseEntity.status(HttpStatus.MOVED_PERMANENTLY)
        .header(HttpHeaders.LOCATION, location)
        .build()
}
```

3.7.2 Testing for Stored Cross Site Scripting

Stored XSS je útok, při kterém útočník vkládá škodlivý kód nějakého uživatelského vstupu, který je posléze uložen na serveru (typicky v databázi). Tento vstup je pak předán uživateli, v jehož prohlížeči dojde ke spuštění škodlivého kódu. Narozdíl od Reflected XSS (popsaného v sekci 3.7.1) je tento typ útoku perzistentní. [11]

Obrázek 3.10 Vstupy do aplikace a výstupy HTML kódu při testování Stored XSS

```
User input:     <script>alert("XSS")</script>
HTML output:    &lt;script&gt;alert("XSS")&lt;/script&gt;
```

Uživatelskými vstupy, které jsou uloženy a později zobrazovány uživatelům, jsou především textová hodnocení. Cross Site Scripting se realizovat nepodařilo. Potencionálně škodlivé znaky (<, >, &) jsou nahrazeny jejich escape variantou (např. sekvencí < pro symbol <); toto je patrné z obrázku 3.10. Nedojde tak ke spuštění JS kódu. Zneužití této zranitelnosti není nejspíše realizovatelné. Obdobným způsobem byly otestovány i další uživatelské vstupy – komentáře k hodnocení předmětů nebo vyučujících.

3.7.3 Testing for HTTP Verb Tampering

Tato část je dle metodologie přesunuta do sekce 3.2.6.

3.7.4 Testing for HTTP Parameter Pollution

Pro přenos dat mezi klientem a server je využíván formát JSON (JavaScript Object Notation). Veškeré požadavky mají své parametry přímo jako součást URL (např. `/semesters/B222`), při duplikaci názvů parametrů je použit první výskyt parametru. Další varianta parametrů je v těle požadavku ve formátu JSON, při duplikaci názvů parametrů je naopak použit poslední výskyt parametru. Nepodařilo se tedy nalézt způsob, jak toto zneužít.

3.7.5 Testing for SQL Injection

Zranitelnost SQL Injection spočívá ve špatné validaci a nakládání s uživatelským vstupem, který vstupuje do SQL dotazu (ukázka vstupu, zranitelného dotazu a výsledného dotazu je v obrázku 3.11). Útočník pak může manipulovat se samotnou strukturou dotazu a získávat data z databáze nebo obcházet logiku aplikace. [11]

Obrázek 3.11 Ukázka SQL Injection [24]

```
SQL query with variable:
    SELECT * FROM Users WHERE UserId = $userId;
User input:
    105 OR 1=1
Final SQL query:
    SELECT * FROM Users WHERE UserId = 105 OR 1=1;
```

Na BE se nenachází žádná místa zranitelná SQL Injection. K provádění databázových dotazů se používá `org.springframework.data.r2dbc.repository.Query`. Každá metoda má pak uvedený dotaz v anotaci `@Query` s odpovídajícími placeholdery; ty jsou pak použity i v samotné metodě (lze vidět v ukázce kódu 3.7). Tento způsob spolehlivě eliminuje riziko SQL Injection.

Výpis kódu 3.7 Ukázka funkce s SQL dotazem na BE [4]

```
@Query(
    """SELECT * FROM EXT_USER_PASSWORD WHERE ID = :userId""",
)
suspend fun getExternalRegistration(
    @Param("userId") id: Long,
): ExternalUserRecord?
```

3.7.6 Testing for LDAP Injection

LDAP se používá jako jedna z možností autentizace uživatelů. Nepodařilo se mi v tomto nalézt žádnou bezpečnostní chybu umožňující LDAP Injection. Implementace zpracování LDAP autentizace je k dispozici v ukázce kódu 3.8.

3.7.7 Testing for XML Injection

XML Injection je typ útoku, při kterém útočník vkládá nežádoucí kód do XML dat, což může mít za následek zneužití a neautorizovaný přístup k systému nebo datům. [11]

Aplikace nevyužívá XML pro přenos dat mezi uživatelem a serverem, ani není možné tento soubor na server nahrát; aplikace není tedy takto zranitelná.

Výpis kódu 3.8 Použití LDAP při autentizaci [4]

```
override suspend fun authorizeExternalLdapUser(
    authLdapPostRequest: AuthLdapPostRequest,
): ResponseEntity<AuthPostResponse> {
    val user = authLdapPostRequest.username
    val properties = Properties().apply {
        this[Context.INITIAL_CONTEXT_FACTORY] =
            ldapProperties.context
        this[Context.PROVIDER_URL] =
            ldapProperties.url
        this[Context.SECURITY_PRINCIPAL] =
            ldapProperties.principal.format(user, user.first())
        this[Context.SECURITY_CREDENTIALS] =
            authLdapPostRequest.password
    }

    val ldapLoginResult =
        runCatching { InitialDirContext(properties).close() }
    return if (ldapLoginResult.isSuccess) {...} else {...}
}
```

3.7.8 Testing for SSI Injection

Server-Side Includes (SSI) Injection je typ útoku, při kterém útočník vkládá kód do SSI direktiv, což může vést k neautorizovanému zobrazení obsahu, spuštění skriptů, … [11]

Aplikace nepoužívá SSI – nejsou nalezeny žádné známky použití ve zdrojovém kódu ani není potvrzena existence souborů s příponou `shtml`.

3.7.9 Testing for XPath Injection

XPath Injection je velice podobná SQL Injection (testováno v sekci 3.7.5), hlavním rozdílem je použití XML souboru jako databáze. Opět se jedná o špatnou validaci uživatelských vstupů. [11]

Aplikace nepoužívá XML pro přenos požadavků; není v ní tedy možné nalézt tuto zranitelnost.

3.7.10 Testing for IMAP SMTP Injection

Aplikace nepoužívá žádnou službu pro odesílání nebo přijímání e-mailů. Díky tomu se v aplikaci nemůže nacházet žádná taková zranitelnost.

3.7.11 Testing for Code Injection

Při Code Injection útočník odesílá vstup, který je pak zpracován webovým serverem jako dynamický kód nebo vložený soubor. [11]

V aplikaci nebyly nalezeny žádné komponenty, kde by byl uživatelský vstup použit v nějakém spustitelném souboru. Zranitelnost injekce kódu se v aplikaci nenachází.

3.7.11.1 Testing for Local File Inclusion

Zranitelnost File Inclusion umožňuje útočníkovi zahrnout soubor nacházející se na serveru – a to především kvůli použití uživatelem dodaného vstupu bez správné validace. [11]

V testované aplikaci neexistuje žádný požadavek, jehož parametr by byl posléze použit jako cesta k souboru. Tuto zranitelnost nelze v aplikaci nalézt.

3.7.11.2 Testing for Remote File Inclusion

Princip této zranitelnosti je stejný jako v sekci 3.7.11.1. Jediným rozdílem je nahrání souboru z jiného webu místo nahrání z serveru. [11]

Ani pro tuto zranitelnost nebyla nalezena žádná komponenta, která by umožnila její provedení.

3.7.12 Testing for Command Injection

Command Injection je technika, při níž útočník prostřednictvím HTTP požadavků provádí systémové příkazy na serveru; toto je umožněno především absencí validace těchto vstupů. [11]

Ani pro tuto zranitelnost neexistuje v aplikaci komponenta, která by používala uživatelem kontrolovaný vstup v rámci příkazu. Zranitelnost zde tedy nelze nalézt.

3.7.13 Testing for Format String Injection

Format String Injection je zranitelnost, při které útočník využívá špatné manipulace s formátovacími řetězci na serveru, aby získal citlivé informace, provedl další útoky atd. [11]

Nepodařilo se mi ve zdrojovém kódu BE najít nějakou část, kde by bylo možné uživatelský vstup zneužít jako formátovací řetězec.

3.7.14 Testing for Incubated Vulnerability

Nebyla nalezena zranitelnost ve validaci uživatelských vstupů, která by umožňovala provedení tohoto typu útoku.

3.7.15 Testing for HTTP Splitting Smuggling

Tato část popisuje útoky využívajících specifické vlastnosti protokolu HTTP (oddělení jednotlivých položek v hlavičce HTTP požadavku pomocí odřádkování nebo odlišné zpracování HTTP požadavků nebo odpovědí na různých komponentách, např. prohlížeč a aplikační firewall) Tyto vlastnosti jsou pak zneužitelné kvůli špatné validaci uživatelských vstupů. [11]

V aplikaci není používán uživatelský vstup pro generování HTTP odpovědí, nelze zde tedy nalézt zranitelnost spočívající v jejich chybné validaci.

3.7.16 Testing for HTTP Incoming Requests

Pomocí *WireShark* byla zachycena komunikace mezi uživatelem a serverem; nebylo nalezeno nic neobvyklého (např. podezřelé požadavky). Komunikace probíhá ve vývojovém prostředí nešifrovaně; v produkčním je šifrována.

3.7.17 Testing for Host Header Injection

Server zpracuje i požadavek se změněným parametrem `Host`; přesměrování na škodlivou URL neproběhne. Ani pomocí parametru `X-Forwarded-Host` není požadavek přesměrován. Server přijímá veškeré požadavky a nezohledňuje parametr `Host` (na IP adrese je v provozu jen 1 doména, proto toto řešení funguje). Aplikace tedy není náchylná k přesměrování na jinou doménu pomocí parametru `Host`.

3.7.18 Testing for Server-side Template Injection

Aplikace nepoužívá technologie pro generování dynamického HTML (*Jinja2*, *Twig*, *FreeMaker*, …). Aplikace je vytvořena jako FE, který získává data pomocí API z BE. Tento typ zranitelnosti zde tedy nelze nalézt.

3.7.19 Testing for Server-Side Request Forgery

Princip SSRF spočívá v tom, že útočník manipuluje server svými požadavky tak, aby prováděl neautorizované síťové požadavky na jiné zdroje, což může vést k zneužití služeb dostupných z vnitřní sítě atd. [11]

Nepodařilo se mi nalézt uživatelský vstup, který by byl na straně serveru použit pro přístup k jiným zdrojům (např. přesměrování nebo načtení jiné stránky v HTTP parametru, spuštění příkazu na serveru, …). Není tedy možné provést tento typ útoku.

3.8 Testing for Error Handling

Tato sekce se zabývá testováním toho, zda chybové hlášky v aplikaci nebo v odpovědi od serveru mohou útočníkovi pomoci odhalit strukturu aplikace nebo zda obsahují citlivé informace.

3.8.1 Testing for Improper Error Handling

Špatné zacházení s chybovými hláškami může vést k odhalení vnitřní struktury aplikace útočníkovi (výpisy zásobníku volání, výpisy paměti, …) [11].

Při manipulaci s payloadem v *POST* požadavku (odstranění `courseID` nebo `surveyID`) server vrátí `400 Bad Request`, odpověď se nachází v obrázku 3.12. Při manipulaci s URL požadavků, ze kterých jsou pak načtena data (`/results`, `/noticeBoard`, …), je vráceno `404 Not Found` a FE vypíše hlášku *Problém s připojením k síti!*

Obrázek 3.12 Odpověď serveru při chybném požadavku (kompletní výpis lze nalézt v `attach/tool-output/http-bad-request.txt`)


```
{
    "timestamp":"2024-04-09T06:52:04.183+00:00",
    "path":"/surveys",
    "status":400,
    "error":"Bad Request",
    "requestId":"8d418e02-3724"
}
```


Nebylo nalezeno žádné místo, kde bylo možné získat chybovou hlášku odhalující vnitřní strukturu aplikace, která by mohla pomoci útočníkovi.

3.8.2 Testing for Stack Traces

Tato část je dle metodologie přesunuta do sekce 3.8.1.

3.9 Testing for Weak Cryptography

Testy v této sekci se zaměřují na správnost implementace kryptografických operací nebo mechanismů. Testy se zaměřují na Transport Layer Security (TLS), Padding Oracle, přenášení dat bez šifrování nebo chybné použití nebo nastavení kryptografických algoritmů.

3.9.1 Testing for Weak Transport Layer Security

Tato část testu je prováděna na produkčním prostředí; vývojové prostředí nepodporuje HTTPS, a tedy ani TLS.

Pomocí nástroje *sslscan* bylo otestováno, které verze TLS jsou podporovány – server podporuje pouze verze 1.2 a 1.3.

Obrázek 3.13 Použití nástroje sslscan (kompletní výpis lze nalézt v `attach/tool-output/sslscan.txt`)

```
$ sslscan anketa.is.cvut.cz
...
SSL/TLS Protocols:
...
    TLSv1.2   enabled
    TLSv1.3   enabled
...
Heartbleed:
    TLSv1.3 not vulnerable to heartbleed
    TLSv1.2 not vulnerable to heartbleed
...
SSL Certificate:
    Signature Algorithm: sha384WithRSAEncryption
    RSA Key Strength:    2048
...
Issuer:   GEANT OV RSA CA 4
...
Not valid before: Aug 18 00:00:00 2023 GMT
Not valid after:  Aug 17 23:59:59 2024 GMT
```

Dále byly zkontrolovány parametry certifikátu serveru. Délka klíče je dostatečná – 2048 bitů. Algoritmus podpisu je `sha384WithRSAEncryption` – použití tohoto algoritmu je validní. Certifikát je validní a doba jeho platnosti nepřesahuje 398 dní. Certifikát je podepsán důvěrnou autoritou – `GEANT OV RSA CA 4`. `Subject Alternate Name` je nastaven na správné domény (`anketa.is.cvut.cz`, `anketa.cvut.cz` a `www.anketa.cvut.cz`).

Při zaměření na problémy, které přináší používání TLSv1.2, bylo ověřeno, že server nepodporuje šifrovací sady, které jsou součástí TLSv1.2 a které byly zároveň označeny jako zastaralé (prolomené nebo nedostatečně bezpečné).

3.9.2 Testing for Padding Oracle

Data mezi klientem a serverem jsou přenášena šifrovaně pomocí TLS. Aplikace nepoužívá jiný způsob šifrování při přenosu dat. Testování implementace TLS, používané touto aplikací, není v této práci realizováno.

3.9.3 Testing for Sensitive Information Sent via Unencrypted Channels

Tento test je prováděn v produkčním prostředí; testovací prostředí nepodporuje HTTPS.

Šifrování dat při přenosu je vynuceno pomocí HSTS (více detailů se nachází v sekci 3.2.7). Odesílání Cookies pouze šifrovaně lze vynutit atributem `Secure`, ten však není nastaven (testování parametrů Cookies je se věnuje sekce 3.6.2). Odesílání Cookies pouze šifrovaně není vynuceno.

3.9.4 Testing for Weak Encryption

Jediným místem, kde je v aplikaci řešena kryptografie je hashování hesel uživatelů. Šifrování dat během přenosu mezi serverem a klientem není záležitostí samotné aplikace, ale serveru a protokolu TLS (testování TLS se nachází v sekci 3.9.1).

K hashování hesel je použit algoritmus `PBKDF2WithHmacSHA1`. Použití tohoto algoritmu je jednou z doporučených možností. [25] Samotné hashování hesel se nachází v ukázce kódu 3.9.

Výpis kódu 3.9 Hashování hesel [4] (kompletní kód lze nalézt v `attach/code/HashUtils.kt`)

```
private const val ITERATION_COUNT = 50000
private const val KEY_LENGTH = 40 * 8
private const val SALT_LENGTH = 32
private const val RADIX = 16
...
fun getPasswordHash(password: String, salt: String): String {
    val skf = SecretKeyFactory.getInstance("PBKDF2WithHmacSHA1")
    val spec = PBEKeySpec(password.toCharArray(),
        salt.toByteArray(), ITERATION_COUNT, KEY_LENGTH)
    return toHexString(skf.generateSecret(spec).encoded)
}
```

Počet iterací při hashování není dostatečný, pouze 50 000. Při použití `PBKDF2` s `SHA1` je doporučeno alespoň 1 300 000 iterací, což je 26krát více než je v aplikaci použito.

Tento bezpečnostní problém je vyhodnocen v sekci 4.10.

3.10 Business Logic Testing

Tato sekce se zaměřuje na testování Business Logic. Důraz je kladen na validaci dat a jejich integritu, nahrávání souborů nebo kontrolu počtu použití konkrétní funkce. Dále je také testována možnost útoku časovým postranním kanálem.

3.10.1 Test Business Logic Data Validation

Veškerá data vkládaná uživatelem jsou pouze textová hodnocení, tyto vstupy není možné z hlediska Business logiky validovat.

3.10.2 Test Ability to Forge Requests

Zfalšované požadavky jsou metodou, kterou útočníci používají k obejití uživatelského rozhraní aplikace. Cílem je např. poslat HTTP požadavky s hodnotami, které nejsou podporovány nebo očekávány webovou aplikací. [11]

Pomocí *BurpSuite* nebyly odhaleny žádné HTTP požadavky obsahující skryté parametry nebo snadno uhodnutelné hodnoty, jejichž úprava by vedla k neoprávněnému přístupu k nějakým datům nebo funkcím. Dle tohoto testu tedy žádná zranitelnost nalezena nebyla.

3.10.3 Test Integrity Checks

V aplikaci nebylo nalezeno žádné místo, kde by nebyla správně ověřována integrita odeslaných dat (většina dat je pouze text, který není třeba validovat v tomto smyslu). U bodového hodnocení je na BE kontrolováno, že se nachází ve stanoveném rozsahu. V aplikaci se také nenachází žádné skryté parametry.

3.10.4 Test for Process Timing

Útočník může při přihlášení pomocí měření času odhalit rozdíl mezi použitím chybného uživatelského jména a použitím validního uživatelského jména a chybného hesla. To vše kvůli tomu, že aplikace nejprve získává data uživatele na základě jeho uživatelského jména; pokud uživatel neexistuje, je vráceno `unauthorized`. Poté probíhá časově náročnější validace hesla. Útočník je tak schopen rozeznat, zda proběhla kontrola hesla či nikoliv. Z toho může zjistit,

jestli dané uživatelské jméno existuje. Toto je zřejmé z okomentované ukázky kódu 3.10.

Tato zranitelnost je podrobně vyhodnocena v sekci 4.4.

Výpis kódu 3.10 Okomentovaný kód autentizace ukazující využití zpoždění odezvy (plná verze kódu se nachází v `/attach/code/authorizeExternalUser.kt`) [4]

```
override suspend fun authorizeExternalUser(...): ... {
    val user =
        authRepository
        .getExternalRegistration(authPostRequest.userId)
        ?: return unauthorized<...>() // returns immediately

    val validPassword = HashUtils.validatePassword(
        // this function takes some time
        authPostRequest.password,
        user.salt,
        user.password,
    )
    return if (validPassword) {
        ok{...}
    } else {
        ... // returns after a longer period
    }
}
```

3.10.5 Test Number of Times a Function Can Be Used Limits

Funkcí, která může být použita pouze jednou, je v této aplikaci např. odeslání hodnocení konkrétního předmětu. Toto je zajištěno zaznamenáním vyplnění ankety v databázi (separátně od samotného hodnocení z důvodu anonymity). Při zpracování požadavku na tvorbu hodnocení je toto ověřeno pomocí `V.SURVEY_FILLED = 'N'`, což je patrné v ukázce kódu 3.11.

3.10.6 Testing for the Circumvention of Work Flows

V aplikaci nebyl nalezen žádný proces, který by se skládal z více kroků a u kterého by bylo možné některý krok vynechat za účelem zneužití nějaké funkcionality.

Výpis kódu 3.11 Ověření, zda uživatel nevyplnil ankety již v minulosti (plná verze kódu se nachází v `/attach/code/getCourseAllowedStudentByUsername.kt`) [4]

```
@Query(
    """
    SELECT
        ...
    FROM
        MV_STUDENT S
    LEFT JOIN
        V_SURVEY_COURSE_STUDENT V ON S.ID_STUDENT = V.ID_STUDENT
    WHERE
        ...
        AND V.SURVEY_FILLED = 'N'
    """,
)
suspend fun getCourseSurveyAllowedStudentByUsername(
    ...
): StudentRecord?
```

3.10.7 Test Defenses Against Application Misuse

Při testování aplikace nebyla detekována žádná obrana aplikace (např. zablokování uživatele při podezřelém chování). Obranné mechanismy nad rámec nepřijetí nevalidního požadavku nebyly nalezeny ani ve zdrojovém kódu. Jedinou, avšak nedostatečnou, obranou je logování – to je implementováno pouze ve funkci kontrolující Session Cookie, logování je vidět v ukázce kódu 3.4. Logování je však nutné také při nastavování nového hesla pomocí tokenu – není zde ale implementováno. Aktivní obrana se v aplikaci nenachází vůbec.

Nedostatečné logování a chybějící aktivní obrana proti zneužití útočníkem jsou vyhodnoceny v sekci 4.3.

3.10.8 Test Upload of Unexpected File Types

V aplikaci se nenachází žádná komponenta, která by umožňovala nahrání souboru. Tento typ zranitelnosti v ní tedy nelze nalézt.

3.10.9 Test Upload of Malicious Files

Závěr této sekce je obdobný závěru předchozí sekce – v aplikaci není možnost nahrát soubor, a proto se zde tato zranitelnost nenachází.

3.11 Client-side Testing

Tato sekce se zaměřuje na testování zranitelností, které jsou realizovatelné na straně klienta. Mezi ně patří: spuštění JavaScriptu, HTML Injection, URL přesměrování, Cross Origin Resource Sharing (CORS), Clickjacking, Cross Site Script Inclusion (XSSI) a další.

3.11.1 Testing for DOM-Based Cross Site Scripting

DOM-Based XSS je typ útoku, při kterém útočník využívá zranitelnosti webové stránky k vložení a vykonání kódů JavaScriptu, které ovlivňují obsah stránky na straně klienta, a to pomocí manipulace s DOM (Document Object Model). [11]

Nebylo nalezeno žádné místo, kde by byla část URL použita pro úpravu DOM nebo tvorbu jeho objektů. Zranitelnost tedy není možné zneužít.

3.11.2 Testing for JavaScript Execution

JavaScript Execution je zranitelnost, která umožňuje vložení JavaScript kódu, který je vykonán v prohlížeči oběti; často v důsledku nedostatečné validace vstupů a výstupů aplikace. [11]

V aplikaci nebylo nalezeno žádné místo, kde by uživatelský vstup byl použit jako součást JavaScriptu.

3.11.3 Testing for HTML Injection

Jedná se o nesprávné ověřování uživatelských vstupů, které jsou posléze vloženy do HTML kódu stránky. [11]

Ve zdrojovém kódu nebyl nalezen žádný případ, kdy by byl nějaký uživatelem kontrolovaný vstup použit jako součást vytvořeného HTML kódu. Hledání ve zdrojovém kódu je zaměřeno na výrazy jako `innerHTML`, `document`, `write` nebo `createElement`.

3.11.4 Testing for Client-side URL Redirect

Pokud je adresa stránky pro přesměrování součástí URL a tento parametr není validován, může útočník vytvořit URL, která bude přesměrována na škodlivý web. [11]

V aplikaci se nachází přesměrování při přihlášení. Uživatel chce navštívit stránku `/teachers`, do Session Storage se uloží `to: /teacher` a po autentizaci dojde k přesměrování – výsledná adresa je jen sloučení s řetězcem `/teachers`. Při vložení `https://www.google.com` jako hodnoty do Session Storage dojde k přesměrování na `http://anketa-vyvoj.cvut.cz/dev/https://www.google.com`, což nezpůsobuje žádné bezpečnostní problémy.

Výpis kódu 3.12 Přesměrování na FE při přihlášení (kompletní kód lze nalézt v `attach/code/LandingPage.jsx`) [5]

```
if (sessionStorage.getItem('from') === '/') {
    this.props.history.push('/');
    window.location.reload();
} else {
    this.props.history.push(sessionStorage.getItem('from'));
    sessionStorage.setItem('from', '');
    window.location.reload();
}
```

3.11.5 Testing for CSS Injection

Zranitelnost CSS Injection spočívá v možnosti vložit libovolný CSS kód do webové stránky, která je pak zobrazena v prohlížeči oběti, což může vést např. k získání Session Cookies. [11]

Aplikace neobsahuje žádnou funkcionalitu, která by umožňovala úpravu Cascading Style Sheets (CSS).

3.11.6 Testing for Client-side Resource Manipulation

Zdrojový kód FE byl prohledán za účelem zjištění, zda se v něm nenachází načítání zdroje, jehož URL je kontrolována uživatelem. Hledání bylo zaměřeno na fráze jako `script`, `src` nebo `href`. Nic zneužitelného však nalezeno nebylo.

3.11.7 Testing Cross Origin Resource Sharing

Bylo otestováno chování BE při odpovídání na požadavky. V odpovědi se nachází atribut, který sděluje, že odpověď může číst pouze `http://anketa-vyvoj.cvut.cz`. Tato zranitelnost se tedy v aplikaci nenachází.

Obrázek 3.14 Atribut určující možnosti CORS v odpovědi od serveru (kompletní odpověď lze nalézt v `attach/tool-output/cors.txt`)

```
Access-Control-Allow-Origin: http://anketa-vyvoj.cvut.cz
```

3.11.8 Testing for Cross Site Flashing

V aplikaci není používán *Action Script*, tato zranitelnost tedy nebude testována.

3.11.9 Testing for Clickjacking

Principem Clickjackingu je, že útočník načte webovou stránku A např. jako `iframe` na své podvodné stránce B, uživatel pak kliká skrze elementy na útočníkově stránce B na komponenty v aplikaci A, přičemž nemá ponětí, že interaguje se stránkou A. [11]

Aplikace neobsahuje žádnou ochranu pro Clickjackingu. Podařilo se mi tak vytvořit a otestovat jednoduché HTML, které je dostupné v `attach/exploit/clickjacking.html`). Testování bylo prováděno v prohlížečí *Firefox* – neúspěšně, Firefox odesílá v rámci požadavku informaci o tom, že se jedná o `iframe` a server požadavek zamítne (parametry požadavku v obrázku 3.15). Jedná se ale o ochranu na straně prohlížeče a nikoliv testované aplikace. Dále byl test proveden v prohlížeči *Edge* – zde tato ochrana není a webová stránka je tak načtena v rámci `iframe`.

Tato zranitelnost je vyhodnocena v sekci 4.11.

Obrázek 3.15 Parametry GET požadavku na načtení stránky v iframe v prohlížeči Firefox (celá hlavička požadavku je v `/attach/tool-output/iframe.txt`)

```
Sec-Fetch-Dest: iframe
Sec-Fetch-Mode: navigate
Sec-Fetch-Site: cross-site
```

3.11.10 Testing WebSockets

WebSockets nejsou v této aplikaci používány – nebyly nalezeny žádné známky použití v síťovém provozu (`ws://` nebo `wss://`) ani ve zdrojovém kódu.

3.11.11 Testing Web Messaging

Web Messaging není v této aplikaci používán. Ve zdrojovém kódu FE nebyla nalezena žádná známka použití funkce `postMessage` ani `addEventListener`.

3.11.12 Testing Browser Storage

Místní úložiště ani indexovaná databáze nejsou aplikací využívány. Session Storage je použito pro uložení URL pro přesměrování po přihlášení – tato

URL není citlivou informací a její zpracování je bezpečné (více detailů je v sekci 3.11.4). Testování Cookies je věnována sekce 3.6.2.

3.11.13 Testing for Cross Site Script Inclusion

Jedná se o útok na straně klienta podobný Cross Site Request Forgery. Zatímco CSRF využívá autentizovaný uživatelský kontext k provedení určitých akcí měnících stav (např. odeslání peněz), XSSI místo toho používá JavaScript k úniku citlivých dat z autentizovaných relací. [11]

Soubory JavaScriptu, které jsou získávány ze serveru, jsou statické a přístupné bez autentizace. Není tedy možné pomocí XSSI získat nějaké citlivé informace.

3.12 API Testing

Tato sekce se zaměřuje na testování technologií využívaných API. Jedinou testovanou technologií je GraphQL.

3.12.1 Testing GraphQL

GraphQL není v aplikaci použito, není tedy nutné provádět toto testování.

Kapitola 4

Vyhodnocení a oprava nalezených zranitelností

Tato kapitola obsahuje výčet nalezených zranitelností a chyb. Každou takovou chybu je popsána a je ohodnocena její závažnost. Je také navržen způsob, jakým by bylo možné ji opravit. Veškeré zranitelnosti jsou řazeny dle závažnosti, a to od těch nejzávažnějších po ty méně závažné.

Tabulka 4.1 Nalezené zranitelnosti včetně hodnocení jejich závažnosti

ID	Zranitelnost nebo chyba	CVSS	Závažnost
1	Session Management	9,1	Critical
2	Cookie parametry	8,6	High
3	nedostatečné logování a obrana proti zneužití	6,9	Medium
4	Process Timing při autentizaci	6,9	Medium
5	chybějící Lock Out mechanismus	6,3	Medium
6	Password Policy	6,3	Medium
7	platnost tokenů pro tvorbu hesla	6,3	Medium
8	přístupnost vývojového prostředí a použitá data	6,3	Medium
9	verze webserveru	6,3	Medium
10	hashování hesel	5,9	Medium
11	načtení v iframe	5,3	Medium
12	chybné napojení FE a BE		
13	nevalidace vstupů ve vývojovém prostředí		
14	proces registrace uživatelů		

4.1 Session Management

Tato zranitelnost byla nalezena v sekcích 3.6.1 a 3.6.6.

Hodnocení zranitelnosti: 9,1 – Critical

Vektor CVSS 4.0 Base Score:
AV:N/AC:H/AT:N/PR:N/UI:N/VC:H/VI:H/VA:N/SC:N/SI:N/SA:N

Vyhodnocení: Použití JavaScript Web Tokenu (JWT) jako Session Cookie je nevhodné především z důvodu nemožnosti jeho zneplatnění. Součástí JWT je digitální podpis – to odstraňuje nutnost mít údaje z tokenu uložené na straně serveru. Stačí totiž pouze zkontrolovat validitu digitálního podpisu. Problémem je však již výše zmíněná nemožnost zneplatnění tohoto tokenu. Nejedná se však o chybu v JWT, ale o volbu využít tuto technologie pro účely Session Managementu. [26]

Součástí JWT vytvořeného serverem je i časová platnost, která je nastavena na 2 dny od okamžiku vytvoření. Tento údaj však není při zpracování serverem ověřován. Je tedy možné použít jakkoliv starý JWT.

Nemožnost zneplatnění tokenu je problematická především z důvodu nemožnosti se správně odhlásit z aplikace, jak je zmíněno v [26]. Session Cookie je sice odstraněno z paměti prohlížeče, ale i nadále zůstává platné. V případě zachycení tohoto Cookie útočníkem nebo jiné kompromitace, tak nemá uživatel jakoukoliv možnost tento token zneplatnit a zamezit tak přístup útočníkovi.

Absence kontroly časové platnosti tohoto tokenu tak umožňuje zneužít token nejen v době jeho platnosti (2 dny od vytvoření), ale i kdykoliv později.

Návrh opravy: Opravou tohoto problému je především vytvoření a implementace takového Session Managementu, který umožní zneplatnění Session při odhlášení uživatele. Z principu této funkce je nutné ukládat informace o aktivních Sessions především na straně serveru.

Jako příklad lze uvést použití dostatečné dlouhé náhodné hodnoty jako Session ID – pouze tato hodnota bude použita jako Session Cookie. Veškeré další údaje o uživateli a Session (uživatelské jméno, časová platnost Session) budou spolu s Session ID uloženy na straně serveru. Další doporučení lze nalézt v [27].

Částečnou opravou, která se jeví méně časově náročnou, je doplnění kontroly časové platnosti tokenu. Je nutné však upozornit, že toto nevyřeší podstatu problému Session Managementu, ale pouze sníží možnost zneužití této zranitelnosti.

4.2 Cookie parametry

Tato zranitelnost byla nalezena v sekci 3.6.2.

Hodnocení zranitelnosti: 8,6 – High

Vektor CVSS 4.0 Base Score:
AV:A/AC:L/AT:N/PR:N/UI:N/VC:H/VI:H/VA:N/SC:N/SI:N/SA:N

Vyhodnocení: Chybné nastavení atributů a parametrů Cookies může vést k získání Cookies útočníkem, získání Session, neoprávněnému provádění požadavků nebo získání citlivých údajů.

Prvním chybně nastaveným parametrem je parametr `Secure`. Ten sděluje prohlížeči, že toto Cookie je možno odeslat pouze přes šifrovaný protokol – HTTPS. Tím je zajištěno, že útočník nezíská toto Cookie, pokud jej zachytí v síťovém provozu. Tento parametr však nastaven není (jeho hodnota je `false`) – není tedy adekvátně zabráněno odeslání Session Cookie přes nešifrované HTTP.

Dalším parametrem, který není správně nastaven, je parametr `SameSite`. Ten určuje, při jakých požadavcích se odesílají Cookies. Tento parametr je nastaven na hodnotu *None* – tato volba odesílá Cookies i v rámci např. linku, *POST* formuláře nebo `iframe`. [28] Toto nastavení umožnuje nebo alespoň usnadňuje provedení útoku typu Cross Site Request Forgery.

Parametr `HttpOnly` je také nastaven nevhodně. Hodnota `false` povoluje JavaScriptu přístup k tomu Cookie. To je problematické v případě provedení útoku typu Cross Site Scripting – útočník pak může v případě úspěšné realizace tohoto útoku získat velice snadno Session Cookies uživatele.

Nastavení parametru `Path` je také chybné. Aplikace se nachází na `http://anketa-vyvoj.cvut.cz/dev/`. Aktuální nastavení `Path` na hodnotu `/` umožňuje odesílání tohoto Cookie při jakémkoliv požadavku na `http://anketa-vyvoj.cvut.cz/`, tedy i mimo samotnou aplikaci. V produkční verzi, která se nachází na `https://anketa.is.cvut.cz/`, toto není pozorovatelné (nenachází se v žádném podadresáři).

Návrh opravy: Nastavit atribut `Secure` na hodnotu `true`, čímž bude vynucen přenost pouze přes HTTPS.

Atribut `SameSite` by měl být nastaven na hodnotu `Strict` (Cookie se neodešle při žádném požadavku z jiné domény), případně `Lax` (odešle se při požadavku z jiné domény pouze v metodě *GET* nebo *HEAD*). [28, 29]

Dále je nutné změnit hodnotu parametru `Path` tak, aby odpovídal reálnému umístění aplikace. Pro vývojovou verzi na `http://anketa-vyvoj.cvut.cz/dev/` musí mít hodnotu `/dev`; na produkční verzi nelze tuto chybu pozorovat, neboť se nenachází v žádném podadresáři.

Oprava atributu `HttpOnly` je komplikovanější, neboť souvisí se nevhodným návrhem Session Managementu – k odstranění Cookies je nevhodně použit JavaScript (detailnější informace o nevhodném Session Managementu a odstraňování Cookies jsou v sekci 4.1). Po opravě Session Managementu je nutné nastavit atribut na hodnotu `true`.

4.3 Nedostatečné logování a obrana proti zneužití

Tato zranitelnost byla nalezena v sekci 3.10.7.

Hodnocení zranitelnosti: 6,9 – Medium

Vektor CVSS 4.0 Base Score:
AV:N/AC:L/AT:N/PR:N/UI:N/VC:L/VI:L/VA:N/SC:N/SI:N/SA:N

Vyhodnocení: Logování není v této aplikaci prováděno dostatečně. Jedinou událostí, která je zaznamenána, je použití neplatného Session Cookie. V dalších kritických místech, jako například autentizace nebo tvorba hesla, není logování implementováno.

Návrh opravy: Přidat logování při neúspěšných akcích. Mezi ně patří neúspěšné přihlášení nebo použití nevalidního tokenu pro tvorbu hesla. Dále je vhodné doplnit aktivní obranu proti zneužití. Tedy blokace uživatele při provádění podezřelých akcí (posílání nevalidních požadavků na server, snaha o SQL Injection, snaha o Cross Site Scripting, …).

4.4 Process Timing při autentizaci

Tato zranitelnost byla nalezena v sekci 3.10.4.

Hodnocení zranitelnosti: 6,9 – Medium

Vektor CVSS 4.0 Base Score:
AV:N/AC:L/AT:N/PR:N/UI:N/VC:L/VI:N/VA:N/SC:N/SI:N/SA:N

Vyhodnocení: Díky rozdílnosti časů odpovědi při pokusu o autentizaci je možné zjišťovat existenci testovaných uživatelských jmen. Rozdíl časů odpovědí se ještě navýší při provedení úpravy dle sekce 4.10. Reálná závažnost této zranitelnosti je výrazně nižší, neboť uživatelská jména jsou vyhledatelná po přihlášení na `https://usermap.cvut.cz`. Použití *CVSS Base Metrics* neumožňuje tento fakt zohlednit.

Návrh opravy: Počítat hash neexistujícího hesla i v případě, že uživatel neexistuje. Tím dojde ke srovnání časů v obou případech (správné a nesprávné uživatelské jméno). Útočník tak nebude schopen využít tento časový postranní kanál.

4.5 Chybějící Lock Out mechanismus

Tato zranitelnost byla nalezena v sekci 3.4.3.

Hodnocení zranitelnosti: 6,3 – Medium

Vektor CVSS 4.0 Base Score:
AV:N/AC:H/AT:N/PR:N/UI:N/VC:L/VI:L/VA:N/SC:N/SI:N/SA:N

Vyhodnocení: Absence Lock Out mechanismu – tedy zablokování uživatele při určitém počtu neúspěšných pokusů o přihlášení – umožňuje útočníkovi odesílat libovolné množství požadavků o autentizaci. Může se tak i úspěšně snažit o prolomení hesla určitého uživatele. Tyto pokusy nejsou navíc ani zaznamenávány (více podrobností je v sekci 4.3).

Návrh opravy: Doplnit Lock Out mechanismus. Aplikace zablokuje daný uživatelský účet na určitou dobu, pokud došlo v určitém časovém intervalu k nějakému počtu neúspěšných pokusů o přihlášení (např. při 10 neúspěšných pokusech v rámci 10 minut, zablokovat uživatele na dobu 1 hodiny). Je však nutné zajistit, aby tento mechanismus nemohl být zneužit pro odmítnutí služby – Denial of Service [30]

4.6 Password Policy

Tato zranitelnost byla nalezena v sekci 3.4.7.

Hodnocení zranitelnosti: 6,3 – Medium

Vektor CVSS 4.0 Base Score:
AV:N/AC:H/AT:N/PR:N/UI:N/VC:L/VI:L/VA:N/SC:N/SI:N/SA:N

Vyhodnocení: Požadavky na komplexitu hesel výrazně zvyšují bezpečnost této autentizační metody (pomocí uživatelského jména a hesla). Jediným požadavkem v této aplikaci je splnění minimální délky hesla – 6 znaků. Pouze požadavek na délku, v tomto případě nedostatečný, není žádnou zárukou bezpečnosti.

Návrh opravy: Je nutné zvýšit počet vyžadovaných znaků hesla alespoň na 8. Velmi užitečná je i funkcionalita zjišťující bezpečnost hesla, např. kontrolou, zda se nenachází v databázích uniklých hesel nebo pouze lokálním posouzením jeho komplexnosti a délky. Tento nástroj je vhodné zpřístupnit uživateli tak, aby při tvorbě hesla mohl zjišťovat jeho bezpečnost. [31]

4.7 Platnost tokenů pro tvorbu hesla

Tato zranitelnost byla nalezena v sekci 3.3.2.

Hodnocení zranitelnosti: 6,3 – Medium

Vektor CVSS 4.0 Base Score:
AV:N/AC:H/AT:P/PR:N/UI:N/VC:L/VI:N/VA:N/SC:N/SI:N/SA:N

Vyhodnocení: Jako jedno z doporučení pro tokeny určené pro reset hesla je uváděno časové omezení jejich platnosti – expirace. [32] Tento mechanismus se však v aplikaci nenachází, žádné informace o časové platnosti nejsou asociovány s konkrétním tokenem. Je tedy možné využít token jakkoliv dlouho po jeho vytvoření, což usnadňuje útoky spočívající v zneužití tohoto tokenu. Použití neplatného tokenu není navíc ani zaznamenáváno, více v sekci 4.3.

Návrh opravy: Časově omezit platnost tokenů určených pro tvorbu hesla. Jako řešení se nabízí ukládání časové expirace do databáze spolu s tokenem nebo využití JWT [33], které je pro tuto funkcionalitu vhodnější než pro Session Management (použití JWT pro Session Management je detailně popsáno v sekci 4.1). [32]

4.8 Přístupnost vývojového prostředí a použitá data

Tato zranitelnost byla nalezena v sekci 3.1.1.

Hodnocení zranitelnosti: 6,3 – Medium

Vektor CVSS 4.0 Base Score:
AV:N/AC:L/AT:P/PR:N/UI:N/VC:L/VI:N/VA:N/SC:N/SI:N/SA:N

Vyhodnocení: Veřejně přístupné vývojové prostředí, na které je odkazováno i z dokumentace aplikace Anketa ČVUT, používá pro účely testování uživatelská data z produkčního prostředí. Informace o použití těchto dat je také dostupná v dokumentaci aplikace, což může z vývojového prostředí vytvářet lákavý cíl pro útočníky. Citlivá uživatelská data jsou tak – bez jakékoliv modifikace – vystavena vysokému riziku jejich úniku. [34]

Návrh opravy: Nepoužívat produkční data bez souhlasu uživatelů a bez jakékoliv modifikace pro účely testování. Pokud je nutné používat produkční data, je doporučené tato data pozměnit tak, aby z nich nebylo možné získat citlivé informace. Nejbezpečnějším řešením je však tvorba dat určených pouze pro účely vývoje a testování. [34] Dále je vhodné omezit přístupnost vývojového prostředí pouze pro uživatele, kteří jsou zapojeni do vývoje a testování této aplikace. Toho lze docílit použitím VPN pro přístup k vývojovému prostředí.

4.9 Verze webserveru

Tato zranitelnost byla nalezena v sekci 3.1.2.

Hodnocení zranitelnosti: 6,3 – Medium

Vektor CVSS 4.0 Base Score:
AV:N/AC:H/AT:N/PR:N/UI:N/VC:N/VI:N/VA:L/SC:N/SI:N/SA:N

Vyhodnocení: Webserver *Apache/2.4.57* obsahuje středně závažnou zranitelnost *CVE-2023-45802*. Tato zranitelnost je opravena v následující verzi. [18] Verze používaná v produkčním prostředí je ještě starší.

Návrh opravy: Je nutné pravidelně aktualizovat software webového serveru. Rychlé provedení aktualizace je nutné především při vydání opravy některé kritické chyby v tomto softwaru.

4.10 Hashování hesel

Tato zranitelnost byla nalezena v sekci 3.9.4.

Hodnocení zranitelnosti: 5,9 – Medium

Vektor CVSS 4.0 Base Score:
AV:N/AC:H/AT:N/PR:H/UI:N/VC:L/VI:H/VA:N/SC:N/SI:N/SA:N

Vyhodnocení: Při hashování hesel není nastaven dostatečný počet iterací. To umožňuje útočníkovi rychlejší počítání hashů hesel při pokusu o jejich prolomení – jedná se tedy o situaci, kdy útočník zná hash a snaží se získat heslo. Dle [35] je nutné nastavit počet iterací na 1 300 000, zatímco v aplikaci je použito pouze 50 000 iterací.

Návrh opravy: Je nutné navýšit počet iterací na alespoň 1 300 000. [35] Další variantou je využít jiný bezpečný algoritmus pro hashování hesel. Je nutné podotknout, že zvýšení počtu iterací zvýší časovou náročnost samotného výpočtu a tím pádem výrazně usnadní detekci existence uživatelských účtů; je nutné implementovat obranu proti tomuto útoku postranním kanálem (o této zranitelnosti pojednává sekce 4.4).

4.11 Načtení webu v iframe

Tato zranitelnost byla nalezena v sekci 3.11.9.

Hodnocení zranitelnosti: 5,3 – Medium

Vektor CVSS 4.0 Base Score:
AV:N/AC:L/AT:N/PR:N/UI:P/VC:N/VI:L/VA:N/SC:N/SI:N/SA:N

Vyhodnocení: Útočník může v rámci své podvodné stránky načítat aplikaci Anketa ČVUT jako `iframe`. Dále může uživatele manipulovat takovým způsobem, že si uživatel myslí, že interaguje s útočníkovou stránkou, zatímco interaguje s aplikací Anketa ČVUT. Tuto zranitelnost nelze zneužít pro získání informací, ale pouze pro provedení nechtěných akcích uživatelem (uživatel se např. domnívá, že kliká na prvek na útočníkově stránce, zatímco kliká na prvek v aplikaci Anketa ČVUT). Jednou z akcí, na které by mohl útočník cílit, je vyplnění anketního lístku. Donutit uživatele k nevědomému hodnocení ankety bez textového hodnocení je proveditelné. Vyplnění textového hodnocení je naopak nejspíše mimo možnosti tohoto typu útoku.

Návrh opravy: Nastavit v odpovědi od serveru parametr `X-Frame-Options` na hodnotu `DENY`, který sděluje webovému prohlížeči, že tato stránka nesmí být načítána v rámci `iframe`. [36]

4.12 Chybné napojení FE a BE

Tato chyba byla nalezena v sekci 3.1.6.

Vyhodnocení: Webová stránka určená pro tvorbu hesla na základě tokenu odesílá požadavek na endpoint `/login/create` na BE. Tento endpoint však neexistuje; aplikace má správně odesílat požadavek na `/user` na BE. Jedná se nejspíše o problém způsobený nekonzistencí názvů endpointů mezi jednotlivými verzemi aplikace.

Návrh opravy: Upravit FE tak, aby byl požadavek odesílán na korektní endpoint. Dále je nutné ověřit, že po opravě tento mechanismus funguje. Nesmí chybět ani pátrání po tom, proč tato chyba nebyla odhalena v rámci vývoje a testování aplikace.

4.13 Nevalidace vstupů ve vývojovém prostředí

Tato chyba byla nalezena v sekci 3.7.1.

Vyhodnocení: Ve funkcionalitě, která umožňuje přihlášení jako jiný uživatel (pouze ve vývojovém prostředí), nedochází k validaci uživatelských vstupů. Uživatelským vstupem je v tomto případě uživatelské jméno, které je posléze použito jako část URL. Pokud použijeme jako uživatelské jméno řetězec, který obsahuje `/`, nebude požadavek odeslán na `/auth/test/{username}`, ale na `/auth/test/{1st_part}/{2nd_part}` – lomítko tedy zafunguje jako separátor úrovní adresářů. Použitím vstupu `../../../test` lze dosáhnout Path Traversal – tedy uniknutí z definované URL cesty. Vše toto je však možné provést pouze ve vývojovém prostředí, chyba tedy nemá reálný dopad a nejedná se o bezpečnostní riziko.

Návrh opravy: Přidat validaci tohoto vstupu. Jako řešení se nabízí povolit pouze alfanumerické znaky (jiné znaky se v uživatelských jménech nepoužívají).

4.14 Proces registrace uživatelů

Tato chyba byla nalezena v sekci 3.3.2.

Vyhodnocení: Samotný fakt, že ne všichni uživatelé se přihlašují pomocí SSO, přináší velké množství problémů. Jedná se především o nutnost implementace správy uživatelských účtů (hashování hesel, které není korektně nastaveno; více detailů je v sekci 4.10). Mezi další funkcionality, které souvisí s interní správou uživatelů, patří tvorba hesel, která není řešena správně (toto je popsáno v sekcích 4.7 a 4.6). Obecně lze říct, že toto řešení výrazně zvětšuje prostor pro případné bezpečnostní chyby.

Návrh opravy: Prosadit odstranění nutnosti existence interní správy uživatelů. Všichni uživatelé se tedy budou muset přihlašovat pomocí SSO. Posléze je možné odstranit komponenty, které se týkají interní správy uživatelů a docílit tak zjednodušení celé aplikace. Jednoduchost samotné aplikace má totiž výrazný vliv na její bezpečnost, toho je docíleno především díky tomu, že v komplikovaných aplikací je mnohem více prostoru pro vytvoření chyb. [37]

Kapitola 5

Doporučení pro vývojáře Ankety ČVUT

Tato kapitola je věnována doporučením pro budoucí vývojáře Ankety ČVUT. Důraz je kladen na oblasti s větším množstvím chyb a zranitelností nebo na oblasti s závažnými zranitelnostmi.

Během testování nebylo možné si nevšimnout vyšší koncentrace chyb v některých sekcích, nejvyšší koncentrace se nachází v sekci Session Managementu. Jedná se především o podcenění nutnosti správného nastavení parametrů nebo podcenění návrhu některé funkcionality, a to zejména z bezpečnostního hlediska.

Je nutné se při návrhu aplikace zamyslet nad návrhem Session Managementu. Je důležité vytvářet návrh nejen se zaměřením na snadnosti jeho implementace, ale také na jeho bezpečnost. Mezi důležité otázky při návrhu musí patřit i to, zda navržený mechanismus splňuje veškeré požadavky kladené na bezpečný Session Management. Jedná se především o náhodnost, nepredikovatelnost a dostatečnou délku Session ID, časovou validitu, zneplatnění Session po odhlášení a také vynucení přenosu Session Cookies zabezpečeným kanálem. [27]

Dále je nutné se zaměřit na korektní nastavení parametrů Cookies v prohlížeči. Jedná se o parametry určující, jakým způsobem a v jakých situacích je možné Cookies odesílat; dále také nastavení parametru, který znemožňuje získání Cookies pomocí JavaScriptu. Správnost tohoto nastavení zabezpečuje aplikaci proti získání Session Cookies útočníkem, proti provedení Cross Site Request Forgery, zmírňuje dopady potencionálního Cross Site Scripting a zabraňuje dalším útokům [27]

Bylo nalezeno více chyb pramenících z nutnosti implementace interní správy uživatelů. Jedná se především o tvorbu a ukládání hesel těchto uživatelů. Lze jednoznačně říct, že existence funkcionalit nutných pro tuto správu přináší

určitou pravděpodobnost vytvoření chyb v jejich návrhu a implementaci. Je tedy vhodné se zamyslet nad tím, zda jsou tyto komponenty v aplikaci nezbytně nutné a zda nepřinášejí větší množství problémů než užitku.

Je nutné zdůraznit, že při návrhu a vývoji aplikace nebo jejích úpravách je nutné se zaměřit nejen na výše uvedené oblasti. Zaměření se pouze na uvedené oblasti by totiž mohlo vést k vytvoření chyb v návrhu nebo implementaci v oblastech jiných.

Kapitola 6
Závěr

Cílem práce bylo provést bezpečnostní analýzu aplikace Anketa ČVUT. K provedení této analýzy byla zvolena metodologie pro testování webových aplikací *OWASP Web Security Testing Guide*. Testování bylo provedeno s přístupem k zdrojovým kódům aplikace.

Úvodní část práce se věnuje stručnému popisu aplikace Anketa ČVUT. Následující část se zabývá existujícími metodologiemi pro provádění bezpečnostních analýz, včetně výběru konkrétní metodologie pro tuto práci. Hlavní náplní praktické části bylo samotné provedení bezpečnostní analýzy webové aplikace. Tato kapitola tedy obsahuje dokumentaci postupu testování dle vybrané metodologie včetně dokumentace nalezených chyb a zranitelností.

Výsledkem je výčet chyb a zranitelností, které byly během této analýzy nalezeny. Pro každou z nich je popsán princip jejího potencionálního zneužití nebo její problematičnost z bezpečnostního hlediska. Dále je také popsán způsob její opravy. Pro prioritizaci nápravy byla pro nalezené zranitelnosti určena jejich závažnost dle *Common Vulnerability Scoring System*. Mezi nalezenými zranitelnostmi se nachází 1 kriticky závažná, 1 vysoce závažná a větší množství středně závažných zranitelností. Dále bylo vytvořeno stručné doporučení pro budoucí vývojáře aplikace, které upozorňuje na problematické oblasti v návrhu a implementaci této aplikace. Nalezené zranitelnosti poukázaly na nutnost věnovat bezpečnosti více prostoru při návrhu a implementaci aplikace.

Jelikož je aplikace Anketa ČVUT stále v procesu vývoje, bude nutné se věnovat testování její bezpečnosti i v budoucnu. Vzestupný trend počtu kybernetických útoků [3] spolu se stoupajícím počtem nalezených zranitelností [1] vynucuje častější a důkladnější bezpečnostní testování nejen v kontextu aplikace Anketa ČVUT.

Volbu metodologie *OWASP Web Security Testing Guide*, která byla použita pro samotnou bezpečnostní analýzu, musím ohodnotit kladně, neboť pokryla veškeré případy, které jsou relevantní ve vztahu k bezpečnosti webových aplikací.

Bibliografie

1. STATISTA INC. Number of common IT security vulnerabilities and exposures (CVEs) worldwide from 2009 to 2024 YTD. *Statista* [online]. 2024 [cit. 2024-03-26]. Dostupné z: `https://www.statista.com/statistics/500755/worldwide-common-vulnerabilities-and-exposures/`.
2. VYSOKESKOLY.CZ. Školy čelí kyberútokům. Útočníci někdy chtějí výkupné. *VysokeSkoly.cz* [online]. 2024 [cit. 2024-03-26]. Dostupné z: `https://www.vysokeskoly.cz/clanek/skoly-celi-kyberutokum-utocnici-nekdy-chteji-vykupne`.
3. RADIMERSKÝ, David. Jednu z fakult ČVUT napadli hackeři, škola teď zjišťuje škody. *ČT24* [online]. 2023 [cit. 2024-03-26]. Dostupné z: `https://ct24.ceskatelevize.cz/clanek/regiony/jednu-z-fakult-cvut-napadli-hackeri-skola-ted-zjistuje-skody-342757`.
4. CHMEL, Oleksandr. *Anketa ČVUT - refaktoring backend*. Praha, Česko, 2024. Dostupné také z: `https://dspace.cvut.cz/handle/10467/113754`. Diplomová práce. České vysoké učení technické v Praze, Fakulta informačních technologií.
5. FURKAN, Sengül. *anketa-cvut-v3 / anketa*. 2024. Dostupné také z: `https://gitlab.fit.cvut.cz/anketa-cvut-v3/anketa/-/tree/feature/update-react2`.
6. HELIKAROVÁ, Eliška. *Analysis of the CTU Teaching Survey's Anonymity*. Prague, Czechia, 2023. Dostupné také z: `https://dspace.cvut.cz/handle/10467/109683`. Bachelor's thesis. Czech Technical University in Prague, Faculty of Information Technology.
7. ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE. *Anketa ČVUT: dokumentace a metodika provozu systému*. Praha, Česko, 2022. Č. Čj. CVUT00018824/2022. Dostupné také z: `https://student.cvut.cz/sites/default/files/content/d1dc93cd-5894-4521-b799-c7e715d3c59e/cs/20221007-metodicky-pokyn-c-32022.pdf`.

8. KNAP, David. *Návrh nového řešení aplikace Anketa ČVUT*. Praha, Česko, 2018. Dostupné také z: https://dspace.cvut.cz/handle/10467/76485. Diplomová práce. České vysoké učení technické v Praze, Fakulta elektrotechnická.
9. FOLLIN, Liam. Penetration Testing Methodologies. *PentestPeople* [online]. [B.r.] [cit. 2024-02-07]. Dostupné z: https://www.pentestpeople.com/blog-posts/penetration-testing-methodologies.
10. ODOM, Chris. The Best Penetration Testing Methodology: Our Review of the Top Four. *Emagined* [online]. 2024 [cit. 2024-02-07]. Dostupné z: https://www.emagined.com/blog/penetration-testing-methodologies.
11. OWASP FOUNDATION, INC. *OWASP Web Security Testing Guide* [online]. Maryland, USA, 2020 [cit. 2024-02-02]. Dostupné z: https://owasp.org/www-project-web-security-testing-guide/.
12. PTES. *Penetration Testing Execution Standard (PTES)* [online]. 2014. [cit. 2024-02-10]. Dostupné z: http://www.pentest-standard.org/index.php/Main_Page.
13. THE INSTITUTE FOR SECURITY AND OPEN METHODOLOGIES. *Open Source Security Testing Methodology Manual*. 2010. Dostupné také z: https://www.isecom.org/OSSTMM.3.pdf.
14. NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. *NIST Special Publication 800-115, Technical Guide to Information Security Testing and Assessment*. Gaithersburg, USA, 2021. Dostupné také z: https://www.nist.gov/privacy-framework/nist-sp-800-115.
15. RAPID7. Vulnerability Management Process. *Rapid7* [online]. [B.r.] [cit. 2024-02-11]. Dostupné z: https://www.rapid7.com/fundamentals/vulnerability-management-and-scanning/.
16. FLEETHAM, Charlie. Vulnerability management metrics: how to measure success. *Intruder* [online]. 2023 [cit. 2024-02-11]. Dostupné z: https://www.intruder.io/blog/vulnerability-management-metrics.
17. FIRST.ORG, INC. *CVSS v4.0 Specification Document* [online]. 2023. [cit. 2024-02-04]. Dostupné z: https://www.first.org/cvss/v4.0/specification-document.
18. THE APACHE SOFTWARE FOUNDATION. *Apache HTTP Server 2.4 vulnerabilities* [online]. 2024. [cit. 2024-03-02]. Dostupné z: https://httpd.apache.org/security/vulnerabilities_24.html.
19. THE APACHE SOFTWARE FOUNDATION. *APACHE 2.4 STATUS* [online]. 2024. [cit. 2024-03-02]. Dostupné z: https://svn.apache.org/repos/asf/httpd/httpd/branches/2.4.x/STATUS.

20. ORENSTEIN, Gary. How long should my password be? *Bitwarden* [online]. 2022 [cit. 2024-04-11]. Dostupné z: https://bitwarden.com/blog/how-long-should-my-password-be/.

21. MICROSOFT CORPORATION. Create and use strong passwords. *Microsoft* [online]. [B.r.] [cit. 2024-04-11]. Dostupné z: https://support.microsoft.com/en-us/windows/create-and-use-strong-passwords-c5cebb49-8c53-4f5e-2bc4-fe357ca048eb.

22. AUTH0 INC. Introduction to JSON Web Tokens. *JWT* [online]. [B.r.] [cit. 2024-04-02]. Dostupné z: https://jwt.io/introduction.

23. LHOTSKÁ, Karolína. *Cross-site zranitelnosti v prohlížečích*. Praha, Česko, 2022. Dostupné také z: https://dspace.cvut.cz/handle/10467/101802. Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií.

24. W3SCHOOLS. SQL Injection. *W3Schools* [online]. 2024 [cit. 2024-03-31]. Dostupné z: https://www.w3schools.com/sql/sql_injection.asp.

25. OBERAI, Aditya. Demystifying password hashing algorithms. *Appwrite* [online]. 2023 [cit. 2024-04-08]. Dostupné z: https://appwrite.io/blog/post/password-hashing-algorithms.

26. COPES, Flavio. JWT authentication: Best practices and when to use it. *LogRocket* [online]. 2023 [cit. 2024-04-19]. Dostupné z: https://blog.logrocket.com/jwt-authentication-best-practices/.

27. OWASP FOUNDATION, INC. Session Management Cheat Sheet. *OWASP Cheat Sheet Series* [online]. 2024 [cit. 2024-04-16]. Dostupné z: https://cheatsheetseries.owasp.org/cheatsheets/Session_Management_Cheat_Sheet.html.

28. GRUDL, David. Co jsou SameSite cookie a proč je potřebujeme? *phpFashion* [online]. 2020 [cit. 2024-04-16]. Dostupné z: https://phpfashion.com/co-jsou-samesite-cookie-a-proc-je-potrebujeme.

29. MEREWOOD, Rowan. SameSite cookies explained. *Web.dev* [online]. 2023 [cit. 2024-05-10]. Dostupné z: https://web.dev/articles/samesite-cookies-explained.

30. OWASP FOUNDATION, INC. Authentication Cheat Sheet. *OWASP Cheat Sheet Series* [online]. 2024 [cit. 2024-04-18]. Dostupné z: https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html.

31. GRACY, Meeba. NIST Password Guidelines: All You Need to Know. *Sprinto* [online]. 2024 [cit. 2024-04-19]. Dostupné z: https://sprinto.com/blog/nist-password-guidelines/.

32. OWASP FOUNDATION, INC. Forgot Password Cheat Sheet. *OWASP Cheat Sheet Series* [online]. 2024 [cit. 2024-04-22]. Dostupné z: `https://cheatsheetseries.owasp.org/cheatsheets/Forgot_Password_Cheat_Sheet.html`.

33. MUNRO, Jamie. Creating Secure Password Resets With JSON Web Tokens. *Smashing Magazine* [online]. 2017 [cit. 2024-04-22]. Dostupné z: `https://www.smashingmagazine.com/2017/11/safe-password-resets-with-json-web-tokens/`.

34. VIRTO COMMERCE. Best Practice for Handling Production Data in Local/Dev/Test Environments by Virto DevLabs. *Medium* [online]. 2022 [cit. 2024-04-22]. Dostupné z: `https://virtocommerce.medium.com/best-practice-for-handling-production-data-in-local-dev-test-environments-by-virto-devlabs-600c5d9980d`.

35. OWASP FOUNDATION, INC. Password Storage Cheat Sheet. *OWASP Cheat Sheet Series* [online]. 2024 [cit. 2024-04-08]. Dostupné z: `https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html`.

36. MOZILLA. X-Frame-Options. *MDN Web Docs* [online]. [B.r.] [cit. 2024-04-23]. Dostupné z: `https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options`.

37. NAMIRIAL. The K.I.S.S. principle for cyber security. *Namirial* [online]. 2023 [cit. 2024-04-23]. Dostupné z: `https://focus.namirial.global/kiss-principle-cyber-security/`.

Obsah příloh

/
- `thesis/` ..... zdrojové soubory textu práce ve formátu LaTeX
- `attach/` ..... přílohy práce
  - `code/` ..... zdrojové kódy testované aplikace
    - `frontend.zip` ..... komprimovaný zdrojový kód FE
    - `backend.zip` ..... komprimovaný zdrojový kód BE
  - `exploit/` ..... soubory pro demonstraci zranitelností
  - `tool-output/` ..... výstup použitých nástrojů
- `thesis.pdf` ..... text práce ve formátu PDF