FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání bakalářské práce

Název:	Bezpečnostní analýza carsharingového systému
Student:	Jakub Sulovský
Vedoucí:	Ing. Filip Ravas
Studijní program:	Informatika
Obor / specializace:	Bezpečnost a informační technologie
Katedra:	Katedra počítačových systémů
Platnost zadání:	do konce letního semestru 2023/2024

Pokyny pro vypracování

1. Seznamte se s bezpečnostními riziky webových aplikací, API a jejich hodnocením.
2. Seznamte se a vyberte vhodný nástroj pro testování bezpečnosti systému Uniqway (www.uniqway.cz).
3. Prozkoumejte a zhodnoťte aplikaci Uniqway z pohledu privilegií využívaných rolí uživatelů aplikace.
4. Proveďte black-box test bezpečnostních rizik aplikace.
5. Proveďte test bezpečnosti API za předpokladu základních znalostí o architektuře projektu.
6. Vyhodnoťte výsledky získané v předchozích krocích a navrhněte nutné změny systému pro mitigaci zranitelností.

Elektronicky schválil/a prof. Ing. Pavel Tvrdík, CSc. dne 15. února 2023 v Praze.

Bakalářská práce

BEZPEČNOSTNÍ ANALÝZA CARSHARINGOVÉHO SYSTÉMU

Jakub Sulovský

Fakulta informačních technologií
Katedra informační bezpečnosti
Vedoucí: Ing. Filip Ravas
15. února 2024

České vysoké učení technické v Praze
Fakulta informačních technologií
© 2023 Jakub Sulovský. Všechna práva vyhrazena.
Tato práce vznikla jako školní dílo na Českém vysokém učení technickém v Praze, Fakultě informačních technologií. Práce je chráněna právními předpisy a mezinárodními úmluvami o právu autorském a právech souvisejících s právem autorským. K jejímu užití, s výjimkou bezúplatných zákonných licencí a nad rámec oprávnění uvedených v Prohlášení, je nezbytný souhlas autora.

Odkaz na tuto práci: Sulovský Jakub. *Bezpečnostní analýza carsharingového systému*. Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií, 2023.

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Chtěl bych poděkovat především vedoucímu práce Ing. Filipu Ravasovi za pomoc, cenné rady a věnovaný čas během psaní této Bakalářské práce. Dále bych chtěl poděkovat své rodině a kamarádu Josefovi za podporu a trpělivost během celého studia.

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. února 2024 .....

Abstrakt

Tato bakalářská práce se zabývá bezpečnostní analýzou carsharingového systému Uniqway. Během této analýzy byly nalezeny bezpečnostní slabiny, ohrožující bezpečnost aplikace a osobních informací uživatelů. Následně byly navrženy úpravy pro jejich mitigaci a zlepšení celkové bezpečnosti systému.

Klíčová slova bezpečnostní analýza, penetrační testování, Uniqway, bezpečnost webových aplikací

Abstract

This bachelor thesis addresses the security analysis of carsharing system Uniqway. During this analysis, security weaknesses were found, threatening the security of the application and users personal information. Subsequently, modifications were proposed to mitigate them and improve the overall security of the system.

Keywords security analysis, penetration testing, Uniqway, web application security

Úvod

Kybernetická bezpečnost a ochrana online prostoru představují klíčové výzvy pro současnou společnost. S postupem času se stále čím dál tím větší část našich životů přesouvá do digitálního světa a s tím roste i množství a hodnota informací uložených elektronicky. V kontextu stále rostoucí digitalizace roste přirozeně i riziko ohrožení elektronických dat a množství kybernetických útoků. Tomuto růstu navíc v posledních letech výrazně přispěly mimo jiné i dvě události, ovlivňující celý svět – globální pandemie Covid-19 a Ruská invaze na Ukrajinu. Ty přinesly do online světa spoustu nových služeb a vytvořily nové cíle pro útoky hackerů. Tyto útoky mohou být motivovány spousty různými faktory. Někteří hackeři provádějí kybernetické útoky s vidinou finančního zisku, jiní zas z politických důvodů, nebo jen tak pro zábavu. Ať už je jejich důvod jakýkoliv, rostoucí počet kybernetických útoků jen podtrhuje potřebu průběžného monitorování, testování a aktualizace bezpečnostních opatření systémů. Analýza a testování bezpečnosti aplikací tak získávají v kybernetickém světě speciální význam.

Důkladná analýza a pravidelné penetrační testování jsou nezbytné pro zajištění bezpečnosti téměř každého informačního systému. Umožňují totiž nasimulovat útok reálného hackera na cílovou aplikaci a objevit případné zranitelnosti, které by mohly být zneužity opravdovým útočníkem.

Tato bakalářská práce se zabývá bezpečnostní analýzou systému Uniqway. Uniqway je první český studentský carsharingový systém a na jeho tvorbě se podíleli studenti ze tří českých univerzit – ČVUT, ČZU a VŠE – ve spolupráci se ŠKODA AUTO. Projekt byl oficiálně spuštěn v říjnu roku 2018 a přinesl studentům a zaměstnancům vybraných vysokých škol nové možnosti v oblasti sdílení vozidel. Uniqway je zároveň první projekt v České republice, vzniklý na akademické půdě, který byl následně překlopen do reálně fungující služby[1].

Hlavním důvod, proč byl pro tuto analýzu zvolen právě systém Uniqway je prostý. Narozdíl od větších systémů a firem, které často mají své vlastní kyberbezpečnostní týmy a pravidelně provádějí bezpečnostní testy svých systémů, Uniqway žádný takový tým nemá. Systém zatím nebyl oficiálně testován a analýza bezpečnosti může přinést úplně nové poznatky ohledně jeho bezpečnostního stavu. Jelikož se navíc jedná o reálně fungující carsharingovou službu, z podstaty věci si uchovává velmi citlivé údaje uživatelů, jako jsou například platební údaje, osobní doklady a další. Únik takových údajů pro uživatele představuje významnou bezpečnostní hrozbu a mohl by mu způsobit značnou finanční a osobnostní újmu. Tento fakt dělá z Uniqway atraktivní cíl pro útočníky a ještě více zvyšuje potřebu důkladné bezpečnostní analýzy.

První kapitola této bakalářské práce zahrnuje krátký úvod do kybernetické bezpečnosti, popisuje bezpečnostní rizika webových aplikací a představuje organizaci OWASP, která se těmito riziky zabývá. V kapitole jsou také představeny některé z projektů, které organizace OWASP vytvořila.

Druhá kapitola uvádí čtenáře do tématiky penetračního testování. Jsou zde popsány základní druhy penetračních testů, jejich fáze a některé z nejpoužívanějších nástrojů, které testerům, ale

i opravdovým útočníkům pomáhají s provedením útoku na cílový systém.

Třetí kapitola se zabývá stručnou analýzou systému Uniqway. Je zde popsána jeho architektura a hlavní komponenty, které společně navzájem komunikují a dohromady tvoří celý systém.

Čtvrtá kapitola se zaměřuje na black-box penetrační testování webových aplikací, které Uniqway provozuje a vystavuje veřejně na internetu. Jsou zde popsány výsledky testování, rozdělené do jednotlivých kategorií dle seznamu 10 nejzávažnějších zranitelností webových aplikací a následně jsou nalezené zranitelnosti zhodnoceny.

Pátá kapitola se zaměřuje na penetrační testování API, které systém Uniqway vystavuje. U tohoto testování je bráno v potaz základní povědomí útočníka o struktuře cílového systému a o testovaném API. Taktéž jsou zde výsledky testování rozdělené do jednotlivých kategorií dle seznamu 10 nejzávažnějších zranitelností API služeb a následně jsou zhodnoceny.

Cíle práce

Cílem teoretické části práce je uvést čtenáře do bezpečnosti webových aplikací a jejího testování, popsat hlavní bezpečnostní rizika webových aplikací a metodiku CVSS, využívanou pro hodnocení závažnosti těchto rizik. Dalším cílem je vybrat a popsat nejpoužívanější nástroje pro testování bezpečnosti webových aplikací.

Cílem praktické části práce je – s pomocí zvolených nástrojů pro penetrační testování – zanalyzovat a otestovat bezpečnost webové aplikace carsharingového systému Uniqway a API, které využívá. Dalším cílem je zhodnotit výsledky provedeného testování a navrhnout změny nutné pro mitigaci případných nalezených zranitelností.

Přínosem práce bude obeznámení autorů systému Uniqway s aktuálním bezpečnostním stavem aplikace a případnými nalezenými zranitelnostmi, jejichž odstraněním budou moct tento stav zlepšit.

Kapitola 1

Bezpečnostní rizika webových aplikací

Podle průzkumů používalo na začátku roku 2023 internet přibližně 5,16 miliardy lidí, tedy cca. 64% světové populace [2]. Webové aplikace se pro většinu z nás staly neodmyslitelnou součástí našeho každodenního života. Umožňují nám rychlý přístup k důležitým informacím, snadnou komunikaci a sdílení dat s lidmi po celém světě a nepřebernou spoustu dalších věcí, které nám mohou velmi usnadnit život.

S rostoucí popularitou webových aplikací však roste i riziko ohrožení bezpečnosti dat, která jsou na internetu sdílena. Spousta webových aplikací umožňuje přístup k důvěrným informacím, které jsou jen jednou z mnoha motivací k útokům hackerů. V případech, kdy jsou tyto útoky úspěšné, mohou mít vážné důsledky, jak pro provozovatele napadené aplikace, tak pro její uživatele. Bezpečnost webových aplikací tedy hraje stále důležitější roli a je potřeba jí během vývoje a provozu aplikace věnovat náležitou pozornost.

1.1 Kybernetická bezpečnost

Kybernetickou bezpečnost můžeme obecně popsat, jako obor, věnující se ochraně kritických systémů a citlivých informací před digitálními útoky. Opatření, která kybernetická bezpečnost přináší, jsou navržena k boji proti hrozbám mířeným na systémy a aplikace, ať už tyto hrozby pocházejí zvnějšku nebo zevnitř organizace.

Vzhledem k tomu, jak je kybernetický svět rozmanitý, existuje spousta různých disciplín kybernetické bezpečnosti. Mezi ty patří například:

- **Síťová bezpečnost**, věnující se ochraně síťové infrastruktury
- **Cloudová bezpečnost**, věnující se ochraně dat, aplikací a infrastruktury v cloudovém prostředí.
- **Aplikační bezpečnost**, věnující se ochraně aplikací v průběhu celého jejich životního cyklu.
- **Informační bezpečnost**, věnující se ochraně citlivých informací.
- **Vzdělávání uživatele**, věnující se dostatečnému obeznámení uživatelů s nutnými bezpečnostními opatřeními.

1.1.1 Obecné pojmy v kybernetické bezpečnosti

Autentizace

Autentizace je proces ověření identity uživatele. Existují tři základní druhy autentizačních faktorů:

1. něco co víme – ověření probíhá pomocí hesla, PIN kódu, nebo jiné informace, kterou uživatel ví.
2. něco co máme – ověření probíhá pomocí fyzického objektu, který má uživatel během autentizace u sebe, např. přístupová karta, USB klíč, mobilní zařízení…
3. něco co jsme – ověření probíhá pomocí biometrické charakteristiky unikátní pro daného uživatele. Tímto faktorem může být například otisk prstu, nebo sken obličeje.

Za účelem zesílení bezpečnosti se také velmi často aplikuje *vícefaktorová autentizace*. Při ní musí uživatel poskytnout minimálně dvě odlišné formy identifikace, což výrazně snižuje riziko neoprávněného přístupu. Pokud by se totiž útočníkovi podařilo prolomit jeden autentizační faktor, musel by se vypořádat s dalšími faktory, což útok výrazně komplikuje.

Autorizace

Autorizace je proces ověření oprávnění uživatele k přístupu k určitým funkcím aplikace. Jejím cílem je zajistit, aby uživatel mohl přistupovat jen k datům a funkcím, ke kterým by měl a na která má dostatečná práva. Autorizační proces je taktéž klíčovým prvkem při ochraně dat. Pomocí pečlivě navržených autorizačních mechanismů může totiž systém efektivně minimalizovat přístup k citlivým informacím a s tím minimalizovat i potenciální dopady v případě, že by došlo k neoprávněnému přístupu.

Kybernetická událost

Kybernetickou bezpečnostní událostí je dle [3] událost, která může způsobit narušení bezpečnosti informací v informačních systémech, bezpečnosti služeb, nebo bezpečnosti a integrity sítí elektronických komunikací. Kybernetickou událostí může být například přihlášení se do systému pod privilegovaným uživatelem „root“.

Kybernetický incident

Kybernetickým bezpečnostním incidentem je dle [3] narušení bezpečnosti informací v informačních systémech, bezpečnosti služeb, nebo bezpečnosti a integrity sítí elektronických komunikací v důsledku kybernetické bezpečnostní události. Kybernetickým incidentem může být například vyřazení webové stránky z provozu pomocí DDoS útoku.

Hrozba

Hrozba je potenciální příčina nežádoucího incidentu, který může mít za následek poškození systému, nebo organizace. Hrozbou může být například útok hrubou silou, nebo phishingový e-mail.

Zranitelnost

Zranitelnost je konkrétní slabé místo aktiva, které může být zneužito jednou, nebo více hrozbami. Zranitelností může být například chyba v konkrétní verzi používaného softwaru.

Obrázek 1.1 Defense in depth – model bezpečnostních opatření dle vrstev

[4]

Exploit

Exploit je kód, který zneužívá konkrétní zranitelnost v systému. Exploity jsou nejčastěji využívány útočníky k získání kontroly nad napadeným systémem, získání citlivých informací, nebo způsobení škod. Mohou být ale použity i penetračními testery, jako „ověření konceptu“ nalezené zranitelnosti.

1.1.2 Důležité koncepty kybernetické bezpečnosti

V kybernetické bezpečnosti se často setkáváme s mnoha různými koncepty, které mají za cíl vytvořit základ pro vývoj a údržbu bezpečných systémů a aplikací. V této sekci se podrobněji podíváme na některé z nich.

Defense in depth

Defense in depth, nebo také vrstvená/víceúrovňová obrana, je koncept, který se opírá o využití několika vrstev bezpečnostních opatření najednou.

Každá z těchto vrstev poskytuje samostatný, nezávislý obranný mechanismus pro zajištění bezpečnosti. Pokud tedy dojde k útoku, nestačí útočníkovi překonat pouze jednu překážku, ale musí jich překonat více, což rapidně zvyšuje náročnost úspěšného útoku.

Bezpečnostní opatření použitá v konceptu *defense in depth* se často rozdělují do následujících 3 oblastí:

1. Fyzická opatření – mají za cíl zajistit fyzické zabezpečení prostor, kde jsou umístěna IT zařízení, proti neoprávněnému přístupu. Příkladem takových opatření mohou být například biometrické systémy, nebo bezpečnostní kamery.

2. Technická opatření – zahrnují hardwarové a softwarové prostředky, zajišťující ochranu proti hrozbám cíleným na aplikace a sítě. Mezi tato opatření patří například firewall, IDS/IPS, nebo různé šifrovací mechanismy.

3. Administrativní opatření – organizační opatření zavedená nejčastěji administrátory / členy bezpečnostního týmu, týkající se řízení přístupu uživatelů k citlivým informacím, pravidel pro používání systému, školení zaměstnanců, atd.

Fail-secure

Fail-secure se zaměřuje na to, aby v případě selhání použitého bezpečnostního mechanismu přešel systém do bezpečného stavu, který minimalizuje nebezpečí a nevytvořil se tak prostor pro útok.

Samotná kontrola, kterou provádí bezpečnostní mechanismus před provedením operace může dopadnout v obecné rovině 3 různými způsoby:

- **Povolení operace**
- **Zamítnutí operace**
- **Vyhození výjimky**

Výpis kódu 1.1 Příklad nesprávně navržené bezpečnostní kontroly

```
has_access = None
try:
    has_access = check_access(current_user) # Kontrola pristupu
except Exception as e:
    logging.info(str(e))

if has_access == False:
    raise AccessDenied()
else:
    privileged_func()
```

Jak lze vidět ve výpisu kódu 1.1, při použití nesprávně navrženého mechanismu může dojít v případě selhání a následného vyhození výjimky k neoprávněnému povolení operace Je důležité zmínit, že výjimku nemusí nutně vyvolat pouze samotná kontrola, ale i jiné operace, které jí předchází. I v tomto případě, pokud je mechanismus nesprávně navržen, může být výsledkem neoprávněné povolení operace.

Bezpečnostní mechanismus by tedy měl být vždy navržen tak, aby bylo v případě selhání výchozím chováním zamítnutí operace.

Výpis kódu 1.2 Příklad správně navržené bezpečnostní kontroly

```
has_access = None
try:
    has_access = check_access(current_user) # Kontrola pristupu
except Exception as e:
    logging.info(str(e))
```

```
if has_access == True:
    privileged_func()
else:
    raise AccessDenied()
```

Princip nejnižších oprávnění

Princip nejnižších oprávnění říká, že by bezpečnostní architektura měla být navržena tak, aby každá entita obdržela minimální množství systémových prostředků a oprávnění, které potřebuje k vykonání své funkce. To znamená, že pokud při své práci uživatel/proces nepotřebuje přistupovat k určitému souboru, funkci, nebo části systému, měl by mu být tento přístup automaticky odepřen.

Dodržováním tohoto principu se výrazně zmenšuje prostor k útoku. Čím méně práv totiž entita má, tím méně jich může být zneužito. To může mít za přínos například:

- **Zvýšení ochrany dat**: Díky tomu, že přístup k citlivým datům mají pouze uživatelé, kteří k nim přistupovat potřebují, klesá riziko jejich úniku. Např. pokud by se k nim chtěl dostat útočník, využitím technik sociálního inženýrství, byl by okruh jeho potenciálních obětí omezen jen na uživatele s přiděleným přístupem.
- **Snížení dopadu škodlivého kódu**: Pokud by se útočníkovi podařilo získat pomocí škodlivého kódu kontrolu nad procesem, získá stejná práva, se kterými byl napadený proces spuštěn. Tedy s čím nižšími právy byl proces spuštěn, tím menší potenciální dopad na systém bude škodlivý kód mít.

S rostoucím počtem entit ale výrazně roste i obtížnost manuálního přiřazování a kontroly oprávnění každé jednotlivé entity v systému. Proto se tak často využívají nástroje a technologie, které tento proces zjednodušují. Mezi ně patří například IAM systémy, které umožňují definovat různé role, obsahující sadu vybraných oprávnění a ty následně přiřazovat uživatelům.

Security through obscurity

Security through obscurity je bezpečnostní koncept, spoléhající se na útočníkovu neznalost systému.

Příkladem může být použití nestandardních a obtížně uhodnutelných názvů souborů/adresářů, nebo například přiřazení nestandardních portů různým službám.

Ačkoliv je tento koncept efektivní ke zmatení a zpomalení útočníků, neposkytuje sám o sobě zabezpečení systému. Útočník může použít různé skenery a metody k získání potřebných informací a ve chvíli, kdy se mu to podaří, ztrácí použití tohoto konceptu význam. Nikdy by tedy neměl být používán jako hlavní bezpečnostní mechanismus, ale jen jako doplněk k jiným mechanismům.

1.2 Open Web Application Security Project (OWASP)

Open Web Application Security Project, zkráceně OWASP, je nezisková organizace, zabývající se bezpečností webových aplikací. Jejím hlavním cílem je šířit povědomí o bezpečnosti webových aplikací a vytvářet nástroje, které mají pomáhat vývojařům a odborníkům na bezpečnost vytvářet a udržovat tyto aplikace tak, aby byly co nejbezpečnější.

OWASP vznikl v roce 2001 a od té doby se stal největší neziskovou organizací svého druhu. V současnosti můžeme na jejich webových stránkách[5] nalézt přes 200 projektů, které zahrnují například:

- Nástroje pro testování bezpečnosti aplikací
- Nástroje pro modelování hrozeb
- Vzdělávací materiály
- Standardy pro vývoj bezpečných aplikací
- Záměrně zranitelné aplikace k trénování

1.2.1 OWASP Top 10

OWASP Top 10 je seznam deseti nejzávažnějších zranitelností webových aplikací. Vzhledem k tomu, že se postupem času objevují nové zranitelnosti a zároveň se závažnost starých zranitelností mění, je seznam pravidelně aktualizován. K aktualizaci dochází přibližně každé 3–4 roky.

Nejnovější verze tohoto seznamu vyšla roku 2021 a obsahuje 8 zranitelností zvolených na základě poskytnutých dat z více než 500 000 aplikací. Zbývající 2 zranitelnosti byly zvoleny na základě uživatelské ankety. Ta umožňuje zařadit na seznam i novější zranitelnosti, které jsou závažné, ale nejsou ještě dostatečně otestované, a tedy nejsou dostatečně hojně zastoupené v nasbíraných datech.

Zranitelnosti uvedené v OWASP Top 10 2021 jsou dle [6]:

1. **Broken Access Control** - Tato zranitelnost vzniká v důsledku chybného nastavení přístupových práv uživatelů. Pokud jsou tato práva nastavena nesprávně, může se uživatel dostat k datům/funkcím, ke kterým by neměl mít přístup. To může mít vážné následky, jako jsou například únik citlivých informací, krádež identity, smazání důležitých dat atd. Zranitelnost tohoto typu byla v nějaké podobě obsažena v 94% testovaných aplikací. [7]

2. **Cryptographic failures** - Tato zranitelnost zahrnuje chyby v kryptografické ochraně aplikace. Mezi tyto chyby patří například posílání citlivých dat v nešifrované podobě, používání slabých kryptografických algoritmů, nesprávné využití kryptografických funkcí a spoustu dalších. Vzhledem k tomu, že dnes téměř každá aplikace pracuje s nějakými citlivými daty (např. hesla, osobní údaje, finanční transakce), je správné použití kryptografických prvků nezbytné pro jejich ochranu. [8]

3. **Injection** - tato zranitelnost je zapříčiněna nesprávnou validací uživatelského vstupu a umožňuje útočníkovi vložit do aplikace škodlivý kód. Typickými příklady mohou být třeba LDAP injection, OS command injection, nebo asi nejznámnější SQL injection, při které útočník vkládá SQL příkazy do vstupních polí aplikace, což může vést k neoprávněnému přístupu k databázi a úniku, úpravě, nebo smazání dat.[9]

4. **Insecure design** - zahrnuje bezpečnostní nedostatky v návrhu aplikace. To mohou být například špatně definovaná rozhraní mezi jednotlivými komponentami aplikace, nebo chybně navržené funkce. Zde je důležité zdůraznit, že chyby vzniklé špatným návrhem nejsou napravitelné správnou implementací a oproti implementačním chybám je mnohem náročnější je opravit. Proto by měla být bezpečnosti věnována náležita pozornost již od fáze návrhu.[10]

5. **Security Misconfiguration** - zahrnuje chyby v konfiguraci aplikace. Těmi může být například používání služeb, které k běhu aplikace nejsou potřeba, ponechání výchozích uživatelských účtů a hesel beze změny, nebo povolené procházení adresářů na serveru. Pro zajištění bezpečnosti v oblasti konfigurace by tak aplikace měla obsahovat jen komponenty, které ke svému běhu potřebuje a ty by měly být konfigurovány ideálně tak, aby v žádném ohledu nerozšiřovaly prostor pro útok. [11]

6. **Vulnerable and outdated components** - Používání zastaralých a zranitelných komponent je problém, vyskytující se u více než poloviny testovaných aplikací. Nejenže může vést k problémům s kompatibilitou a výkonem aplikace, ale představuje i nemalé bezpečnostní riziko. Konkrétní verze komponent (knihovny, frameworky, softwary třetích stran) často obsahují známé zranitelnosti, které mohou být zneužity útočníkem. Použití takových komponent pak může ohrožovat celou aplikaci. Je tak důležité pravidelně kontrolovat bezpečnost a kompatiblilitu používaných komponent. [12]

7. **Identification and authentication failures** - zahrnuje chyby při autentizaci uživatele a správě relací. Ty mohou útočníkovi umožnit například provádět, za účelem získání uživatelských údajů, útok hrubou silou, nebo se pomocí kompromitovaného jednoznačného identifikátoru relace vydávat za jiného uživatele. Je tedy důležité zajistit bezpečnou správu relací a použití silného autentizačního mechanismu. Další účinnou obrannou metodou je použití vícefaktorové autentizace, která rozšiřuje autentizační proces o další faktor a zajišťuje tak ochranu i v případě úspěšného prolomení uživatelských údajů. [13]

8. **Software and Data Integrity Failures** - zahrnuje chyby při ověřování integrity dat. Téměř každá aplikace používá nějaké komponenty třetích stran (pluginy, knihovny, moduly…), které stahuje z externích zdrojů. Tyto zdroje mohou být napadeny a mohou být vydány nové verze komponent, obsahující škodlivý kód. Pokud zdroje a z nich stahované komponenty nejsou dostatečně ověřovány, může dojít ke stažení verze, obsahující škodlivý kód. Je tedy důležité ověřovat integritu stažených dat (například pomocí digitálního podpisu) a věrohodnost použitého zdroje. [14]

9. **Security Logging and Monitoring Failures** - zahrnuje bezpečnostní nedostatky spojené s nedostatečným logováním a monitorováním. Logování má za účel sběr a úkládání dat, která jsou následně monitoringem analyzovány k identifikaci potenciálních bezpečnostních hrozeb. Absence logování důležitých událostí (přihlášení uživatele, neúspěšný pokus o přihlášení, atd.) výrazně ztěžuje a zpomaluje detekci útoků. Kromě logování je stejně tak důležité efektivní a pravidelné monitorování. V opačném případě může dojít k přehlédnutí podezřelé události, což má z bezpečnostního hlediska stejný výsledek, jako by data nebyla logována vůbec. [15]

10. **Server Side Request Forgery** - Tato zranitelnost může být zapříčiněna špatnou konfigurací serveru, nebo chybějící validací URL adresy a vzniká ve chvíli, kdy webová aplikace získává data z externího zdroje, poskytnutého uživatelem. Umožňuje útočníkovi přimět server, aby provedl neočekávané požadavky na jiné servery, nebo služby dostupné pouze z interní sítě, což může mít za následek například únik citlivých dat, nebo spuštění škodlivého kódu na vzdáleném serveru. [16]

1.2.2 OWASP API Security Top 10

API (Application Programming Interface) je rozhraní, které umožňuje aplikacím komunikovat mezi sebou a poskytuje programátorům jednotný způsob, jak komunikovat s danou aplikací, aniž by potřebovali znát její vnitřní implementaci.

Většina webových aplikací používá API pro získávání dat z jiných služeb, nebo naopak pro poskytování svých dat ostatním aplikacím. Stává se tak logicky potenciálním cílem útočníků a jeho bezpečnosti by měla být věnována náležitá pozornost.

OWASP API Security Top 10 seznamuje s deseti nejběžnějšími zranitelnostmi, které lze v API nalézt a s doporučeními, jak jim předejít. Stejně jako u předchozího seznamu, i tento je průběžně aktualizován

Pro rok 2023 byl zveřejněn koncept nové verze seznamu, který dle [17] obsahuje následující zranitelnosti:

1. **Broken Object Level Authorization** - Zranitelnost zapříčiněná absencí autorizace na úrovni objektu. API endpointy často přijímají jako jeden z parametrů identifikátor objektu, se kterým poté provádí nějakou operaci. Pokud API nekontroluje přístup uživatele k danému objektu, může útočník upravit identifikátor objektu v odesílaném požadavku a získat/upravit/smazat citlivá data, ke kterým by správně neměl mít přístup. Je tedy důležité zajistit, aby autorizace na úrovni objektu vždy probíhala. [18]

2. **Broken Authentication** - Zranitelnosti spojené s procesem autentizace. Mezi ně patří například možnost útoku hrubou silou bez jakékoliv limitace, používání slabých hashovacích algoritmů, nebo odesílání citlivých autentizačních detailů (token, heslo, atd.) v URL adrese. Endpointy, které mají proces autentizace na starost (registrace, přihlášení, obnova hesla) jsou pro útočníka zároveň jednoduchým cílem, jelikož jsou z podstaty věci dostupné komukoliv a je tak potřeba věnovat jejich zabezpečení zvýšenou pozornost.[19]

3. **Broken Object Property Level Authorization** - Zranitelnost zapříčiněná absencí autorizace na úrovni vlastností objektu. Tento bod spojuje dohromady 2 kategorie z předchozí verze seznamu – Excessive Data Exposure a Mass Assignment. API totiž často v rámci objektu odesílají nadbytečné a mnohdy citlivé informace, které jsou pro uživatele v případě zachycení odpovědi viditelné (Excessive Data Exposure). Pokud navíc neprobíhá autorizace úpravy vlastností objektu, může uživatel upravit požadavek sloužící k úpravě objektu tak, aby odstranil, přidal, nebo změnil vlastnosti, které by správně měnit neměl (Mass Assignment). Kromě kontroly přístupu k samotnému objektu je tedy důležité odesílat jen ty vlastnosti objektu, které by měl uživatel vidět a zároveň zajistit, aby uživatel mohl upravovat jen vlastnosti, které by měl. [20]

4. **Unrestricted Resource Consumption** - Vykonávání požadavků na API spotřebovává zdroje, jako jsou operační paměť, paměť úložiště, atd. Často se stává, že jsou požadavky limitovány nedostatečně, nebo dokonce vůbec. To může být zneužito útočníkem a následně vést k zaplnění paměti, zvýšení finančních nákladů, nebo odepření služby celého API. Jedním z možných scénářů útoku může být například odeslání obrovského souboru na endpoint sloužící pro nahrání souboru. [21]

5. **Broken Function Level Authorization** - Takřka v každém API nalezneme endpointy, které by měly být dostupné jen pro uživatele s vyššími oprávněními. Pokud potřebná oprávnění nejsou kontrolována správně, může tyto endpointy volat i uživatel, který by k němu neměl mít přístup. Využití privilegovaných endpointů běžným uživatelem může mít za následek například manipulaci s citlivými daty, nebo eskalaci oprávnění. [22]

6. **Unrestricted Access to Sensitive Business Flows** - Zranitelnost způsobená nedostatečnou ochranou proti automatizovanému použití API. Zneužití této slabiny je ve většině případech těžko detekovatelné a má pro organizaci spíše ekonomický, než technický dopad, i přesto je potřeba věnovat jí náležitou pozornost. Ke zneužití dochází ve chvíli, kdy útočník chápe určitou funkcionalitu krok po kroku a dokáže její provádění zautomatizovat. Tím se může zvýhodnit, například při nákupu limitovaného zboží z e-shopu, nebo jiné akci, u které je očekáváno, že bude prováděna člověkem. [23]

7. **Server Side Request Forgery** - Tato zranitelnost je takřka identická se zranitelností Server Side Request Forgery ze seznamu OWASP TOP 10. I API často získávají data z adres poskytnutých uživateli. Pokud tyto adresy nejsou správně validovány, může útočník získat citlivá data z interních adres, na které má API přístup, případně přimět server k provedení nežádoucích požadavků. [24]

8. **Security Misconfiguration** - Zahrnuje chyby v konfiguraci API. Mezi ně patří například používání nepotřebných komponent, absence TLS, nebo zobrazování příliš detailních chybových hlášek, obsahujících citlivé informace, na straně uživatele. Těmto chybám lze většinou

snadno předejít a zbytečně zvětšují prostor pro útok, měl by na ně tedy být při konfigurování API brán ohled.[25]

9. **Improper Inventory Management** - Tato zranitelnost vzniká špatnou správou API a jeho nedostatečnou dokumentací. Aplikace velmi často provozují několik verzí API, určených pro různé prostředí (např. vývoj, test, produkce). Nedostatky v dokumentaci a chybná správa často způsobují, že jsou tyto verze dostupné i mimo prostředí, pro které jsou určené. Často se také stává, že jsou staré verze aktivní a dostupné i v případě nahrazení novou verzí. Jelikož jsou endpointy těchto verzí API mnohdy hůře zabezpečené, mohou být tyto nedostatky využity útočníkem.[26]
10. **Unsafe Consumption of APIs** - Tato zranitelnost je zapříčiněna přílišnou důvěrou k API třetích stran a nedostatečným zabezpečením při jejích používání. Vývojáři často sází na věrohodnost externího API a nevěnují tak dostatečnou pozornost zabezpečení komunikace s nimi. Mezi takto vzniklé bezpečnostní nedostatky patří například komunikace skrz nešifrovaný kanál, neověřování přesměrování, nebo nedostatečná validace obdržených dat. V případě kompromitace externího API může pak útočník využít těchto nedostatků k napadení jiných API, které ho používají. Při používání externích API by tedy vždy měla proběhnout důkladná analýza bezpečnostních rizik a zabezpečení komunikace v souladu s nejlepšími postupy.[27]

1.2.3 OWASP WSTG

OWASP WSTG (Web Security Testing Guide) [28] je rozsáhlý průvodce, poskytující ucelený přístup k testování bezpečnosti webových aplikací. Jeho hlavním cílem je pomoci bezpečnostním testerům a vývojářům vytvářet bezpečné webové aplikace a chránit je proti co nejširší škále kybernetických útoků.

Je rozdělen do několika částí, které pokrývají různé aspekty testování bezpečnosti webových aplikací. Každá z těchto částí poskytuje podrobné popisy jednotlivých zranitelností, včetně vysvětlení jak fungují a jak je mohou útočníci zneužít. Kromě toho můžeme v každé části najít podrobné postupy pro testování těchto zranitelností.

Je dobré zmínit, že narozdíl od OWASP TOP 10 a OWASP API Security TOP 10, zmiňovaných dříve, které obsahují pouze nejzávažnější zranitelnosti a mají tak sloužit spíše jako vzdělávací materiál, OWASP WSTG nabízí mnohem komplexnější pohled na problematiku bezpečnosti webových aplikací a je používán širokou škálou bezpečnostních testerů jako standard pro testování bezpečnosti. Navíc je pravidelně aktualizován a je tak v souladu s nejnovějšími hrozbami a zranitelnostmi webových aplikací.

1.3 Hodnocení závažnosti zranitelností

Hodnocení závažnosti zranitelností je proces, sloužící k identifikaci a hodnocení zranitelností, které mohou ohrozit bezpečnost informačního systému. Jeho hlavním cílem je poskytnout přehled o tom, jaké zranitelnosti jsou nejvíce závažné a které části systému by měly být při implementaci bezpečnostních opatření prioritizovány.

Celý proces je velmi komplexní a zahrnuje mnoho různých faktorů, jako jsou např. proveditelnost útoku, důležitost ohroženého aktiva, atd. Tyto faktory bývají často subjektivní a výsledné hodnocení se tak mezi jednotlivci může výrazně lišit. Proto je dobré používat nějakou metodiku, která zajistí konzistentní výsledky. Jednou z takových metodik je CVSS.

1.3.1 CVSS

CVSS (Common Vulnerability Scoring System) [29] je standardizovaná metodika pro hodnocení závažnosti zranitelností. Jejím cílem je poskytnout postup, jak zranitelnosti hodnotit

jednotně a nezávisle na cílové platformě.

Existuje několik verzí CVSS, přičemž tou nejnovější je CVSS v3.1. Hojně používány však zůstávají i starší verze, jako je například CVSS v2.0.

Metodika CVSS se dle [30] skládá ze tří druhů metrik, na základě kterých se závažnost zranitelnosti vyhodnocuje:

- **Základní metriky (Base metrics)**, popisující vlastnosti, které se pro danou zranitelnost postupem času, ani změnou prostředí nemění. Ty se dále dělí na metriky spojené se složitostí zneužití dané zranitelnosti (Exploitability metrics) a metriky spojené s následky v případě jejího úspěšného zneužití (Impact metrics).
- **Dočasné metriky (Temporal metrics)**, popisující vlastnosti, které se pro danou zranitelnost mění v průběhu času. Tyto vlastnosti mohou být závislé např. na existenci exploitů, nebo na množství informací, které jsou o zranitelnosti dostupné.
- **Metriky závislé na prostředí (Enviromental metrics)**, popisující vlastnosti, které se pro danou zranitelnost mění v závislosti na prostředí, pro které se zranitelnost hodnotí. Takové vlastnosti mohou být závislé např. na důležitosti ohrožených aktiv, nebo na bezpečnostních opatřeních, která jsou v daném prostředí zavedena.

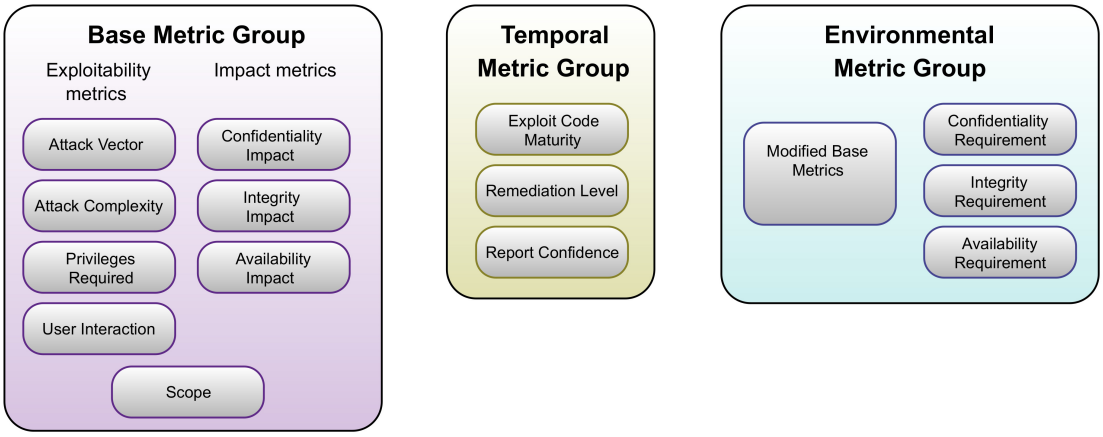
Každá z možných hodnot jednotlivých metrik má přiřazenou číselnou konstantu, která se dále dosazuje do vzorců pro výpočet závažnosti. Aby byl tento proces pro uživatele co nejjednodušší, je na stránkách organizace FIRST, která metodiku CVSS spravuje, dostupná kalkulačka [31], pomocí níž lze závažnost na základě zadaných hodnot metrik vypočítat.

Výsledkem je *CVSS skóre* – číselná hodnota mezi 0 a 10, kde 0 reprezentuje nejnižší závažnost a 10 nejvyšší. Tato hodnota může být alternativně převedena na slovní reprezentaci dle tabulky.

Závažnost	CVSS skóre
None	0.0
Low	0.1 – 3.9
Medium	4.0 – 6.9
High	7.0 – 8.9
Critical	9.0 – 10.0

Tabulka 1.1 Slovní reprezentace CVSS skóre dle CVSS v3.1

Je dobré zmínit, že k výpočtu CVSS skóre jsou povinné pouze základní metriky. Použití dočasných metrik a metrik závislých na prostředí je nepovinné, avšak umožňuje lépe zohlednit kontext zranitelnosti a získat tak přesnější výsledky vzhledem k danému prostředí a časovému období.



Obrázek 1.2 Metriky CVSS v3.1

[32]

Kapitola 2
Penetrační testování

Nedílnou součástí bezpečnostní analýzy je i penetrační testování – systematický proces, při kterém je zkoumána odolnost aplikace vůči potenciálním útokům. Penetrační testování simuluje reálný útok na aplikaci s cílem odhalit a identifikovat bezpečnostní slabiny a zranitelnosti v systému, dříve než by je odhalil skutečný útočník.

Tento proces poskytuje organizacím cenné informace o bezpečnostním stavu jejich aplikace. Získané poznatky pak umožňují zavést bezpečnostní opatření, nezbytná pro odstranění nalezených slabin a zamezit tak jejich zneužití.

2.1 Druhy penetračních testů

Existuje několik různých kritérií, podle kterých lze penetrační testy dělit (například podle rozsahu testování, úrovně automatizace, typu útoku atd.). Pro účely této práce bylo zvoleno dělení podle úrovně znalostí o testovaném systému.

Na základě množství poskytnutých informací o testovaném systému existují 3 způsoby, jak k penetračnímu testování přistupovat.

Black-box testování

Při black-box testování má tester stejné znalosti o testované aplikaci, jako běžný uživatel. Nejsou mu poskytnuty žádné informace o architektuře aplikace, ani přístup ke zdrojovým kódům. Celé testování tak probíhá externě, skrz funkce, které jsou veřejně dostupné uživatelům aplikace. Tento druh testování nejautentičtěji simuluje reálný útok neprivilegovaného externího útočníka.

Nevýhodou tohoto přístupu je, že odhaluje jen zranitelnosti zneužitelné z vnějšku a případné interní slabiny nechá skryty.

White-box testování

Při white-box testování má tester kompletní přístup ke zdrojovým kódům testované aplikace a k detailním informacím o její architektuře a implementaci. To umožňuje testovat aplikaci na výrazně hlubší úrovni, než při black-box testování.

Testování probíha interně a zahrnuje analýzu zdrojového kódu jednotlivých komponent aplikace. Tester oproti black-box přístupu netestuje bezpečnost aplikace z pozice uživatele, ale z pozice vývojáře. Kvůli tomu, je také od testera vyžadována vysoká úroveň technického know-how a znalost programování.

Grey-box testování

Grey-box testování je kompromisem mezi white-box a black-box testováním. Tester dostane informace o vnitřní implementaci a architektuře aplikace, které jsou ale oproti white-box přístupu výrazně limitovány (např. chybí přístup ke zdrojovým kódům). To dává testerovi větší povědomí o testované aplikaci než při black-box testování a umožňuje tak identifikovat zranitelnosti, které v běžném uživatelském rozhraní nejsou viditelné.

2.2 Fáze penetračního testování

Stejně jako v předchozí sekci, která se věnovala dělení penetračních testů, ani zde není uváděná varianta tou jedinou možnou. Existuje mnoho různých metodologií pro provádění penetračních testů, které se mnohdy mj. liší i v tom, na jaké fáze testování člení. Pro účely této práce bylo zvoleno členění podle PTES (Penetration Testing Execution Standard).

PTES [33] je standard, vyvinutý s cílem poskytnout strukturovaný přístup k provádění penetračních testů a zajistit jejich konzistenci. Tento standard dělí penetrační testování na 7 následujících fází:

1. **Pre-engagement interactions** - První fáze se zaměřuje na úvodní interakce mezi testerem a klientem, před začátkem samotného testování. Probíhají během ní diskuze o očekáváních klienta, za účelem stanovení cílů a rozsahu testování. Během této fáze by měly být také dohodnuty základní parametry testování, jako je například finanční ohodnocení, odhadovaná doba trvání, atd. Tato fáze je klíčová pro vzájemné porozumění obou stran a získání obrazu o tom, co se od testování očekává. [34]

2. **Information gathering** - Druhá fáze se zaměřuje na sběr informací o cílovém systému. V rámci této fáze jsou využívány techniky sběru informací, ať už pasivního (např. OSINT), či aktivního (např. skenování portů) s cílem získat co největší povědomí o testovaném prostředí. Shromážděná relevantní data, mohou testerovi pomoci lépe porozumět architektuře aplikace, odhalit potenciální zranitelnosti a lépe tak naplánovat následný útok. [35]

3. **Threat Modeling** - Náplní třetí fáze je modelování hrozeb. Během něho jsou ve spolupráci s klientem identifikována a analyzována ohrožená aktiva cílové organizace a s nimi spojené hrozby a jejich potenciální aktéři. Následně probíhá vyhodnocení jednotlivých rizik, beroucí v potaz faktory, jako jsou například hodnota ohrožených aktiv, technické možnosti útočníků, nebo motivace k útoku. Celý proces následně umožňuje prioritizovat jednotlivé části systému při dalších fázích penetračního testování. [36]

4. **Vulnerability analysis** - Čtvrtou fází je analýza zranitelností – proces, při kterém jsou v cílovém systému hledány slabiny, které by mohly být zneužity útočníkem. K tomu jsou často používány specializované nástroje, které jsou schopny značnou část práce zautomatizovat a ušetřit tak testerovi spoustu času. Součástí nálezu zranitelnosti je také zhodnocení z hlediska obtížnosti jejího zneužití. Nalezené zranitelnosti jsou dále použity, jako přímý vstup v následující fázi. [37]

5. **Exploitation** - Pátá fáze se zaměřuje na samotné zneužití nalezených zranitelností. Tester při ní provádí útoky mířené na zranitelné části aplikace, s cílem pronikout do systému, získat neautorizovaný přístup k privilegovaným funkcím, nebo provést jakékoliv jiné nežádoucí akce, které jsou v souladu s dohodnutými pravidly testování. Tyto útoky zahrnují využití různých technik a nástrojů, jako jsou například *exploity*, reverzní inženýrství, útok hrubou silou a spoustu dalších. [38]

6. **Post-exploitation** - V případě, že se během předchozí fáze podařilo proniknout do systému, je cílem šesté fáze prozkoumání a zhodnocení kompromitovaných komponent a zajištění udržení kontroly pro jejich pozdější použití. To zahrnuje například identifikaci a extrakci citlivých

dat, pokus o rozšíření získaných oprávnění, odhalení případných dalších systémů, analýzu konfigurace atd. [39]

7. **Reporting** - Během poslední fáze jsou shrnuty získané výsledky, na základě kterých tester vytvoří závěrečný report. Report by měl obsahovat netechnickou a technickou část. Netechnická část je mířena převážně pro management a měla by popisovat nalezené zranitelnosti především z hlediska jejich obchodního dopadu. Technická část je naopak mířena pro odborníky, kteří se budou podílet na opravě nalezených slabin. V této části by měly být detailně popsány nalezené zranitelnosti, jejich závažnost, návrhy na mitigaci a jednotlivé kroky provedené během testování, včetně použitých technik, nástrojů a metodologií. Výsledný report je nejdůležitějším výstupem celého testování a slouží klientovi jako důležitý nástroj pro pochopení rizik a přijetí vhodných opatření pro jejich odstranění. [40]

2.3 Nástroje pro penetrační testování

Specializované nástroje jsou nedílnou součástí takřka každého penetračního testu. Tyto nástroje umožňují testerovi značnou část testování zautomatizovat a provést ho tak efektivněji a komplexněji.

Existuje mnoho různých nástrojů, které pokrývají všechny možné aspekty penetračního testování, od sběru informací, přes analýzu zranitelností, až po exploitaci. V této sekci si představíme některé z nich.

2.3.1 Nmap

Nmap [41] je rozsáhlý síťový skener, který je nejčastěji používán k zjištění stavu síťových portů a identifikaci veřejně vystavených běžících služeb. Tento nástroj nabízí širokou škálu funkcí a přepínačů, které umožňují upravit parametry skenu (rychlost, protokol, rozsah portů, …) dle potřeby a získat detailnější informace, jako jsou například používané operační systémy, verze běžících služeb a spoustu dalších.

Jednou z největších výhod nástroje Nmap je jeho rozšiřitelnost pomocí skriptovacího enginu NSE. Ten umožňuje uživatelům psát a spouštět skripty pro automatizaci různých úkolů během skenování. Těmito úkoly může být například analýza zranitelností, nebo dokonce jejich exploitace.

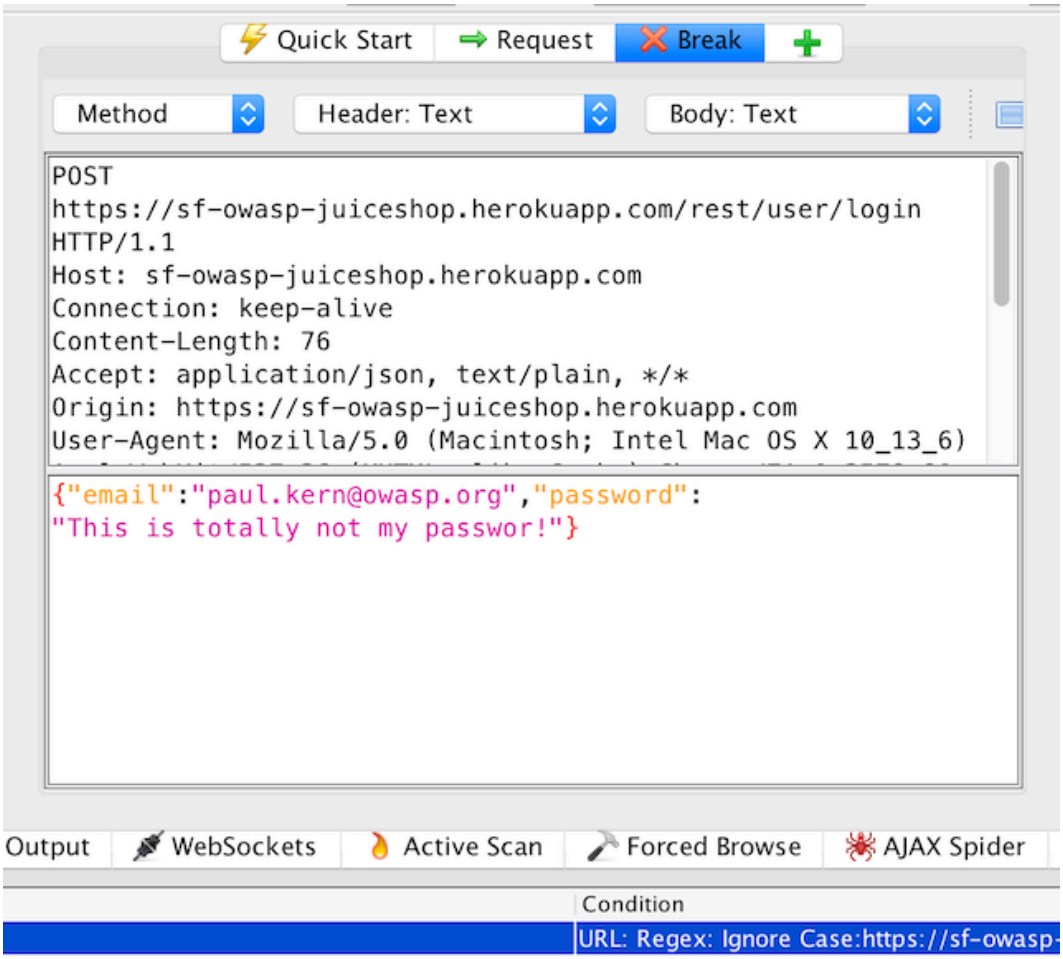
Níže je jako příklad použití uveden příkaz, který na stránce *www.adresacile.cz* provede pomocí nástroje Nmap sken 100 nejvyužívanějších portů, pokusí se detekovat verze běžících služeb a pomocí NSE spustí skripty zaměřené na analýzu zranitelností

```
nmap -F -sV --script vuln www.adresacile.cz
```

2.3.2 OWASP ZAP

OWASP Zed Attack Proxy (ZAP) [43] je komplexní program pro testování bezpečnosti webových aplikací. Díky širokému spektru nabízených funkcionalit, pokrývajících mnoho různých aspektů testování bezpečnosti, je OWASP ZAP nezbytným nástrojem pro spoustu penetračních testerů. Mezi nejužitečnější nabízené funkcionality patří například:

- **Automatizovaný skener**, který zahrnuje pasivní [44] a aktivní [45] skenování. Pasivní skenování skenuje veškerou komunikaci s testovanou aplikací a na základě ní identifikuje časté bezpečnostní chyby. Toto skenování probíhá na pozadí a nijak nemanipuluje s požadavky ani odpověďmi. Aktivní skenování hledá potenciální zranitelnosti tím, že s testovanou aplikací přímo komunikuje a provádí na ní známé útoky. Mělo by tak být používáno výrazně opatrněji.



Obrázek 2.1 Ukázka požadavku zachyceného pomocí Owasp ZAP Proxy.

[42]

- **Fuzzer**, který umožňuje generovat a následně odesílat náhodné, nebo cílené vstupy do testované aplikace, za účelem odhalení potenciálních chyb ve zpracování vstupních dat. Ty nejčastěji vedou k problémům jako jsou přetečení bufferu, nebo odepření služby (DoS). [46]

- **Spider**, který slouží k prozkoumávání webových aplikací a sběru informací o jejich struktuře. Spider typicky dostane výchozí seznam URL adres k prozkoumání, tzv. *seedy*. Každou z těchto adres navštíví, identifikuje odkazy, které se na nich nachází a tyto odkazy přidá do seznamu adres k prozkoumání. Tento proces je rekurzivně opakován, dokud se daří objevovat nové adresy. Použití tohoto nástroje tak zajišťuje důkladnější pokrytí, při testování bezpečnosti webových aplikací. [47]

- **Proxy**, umožňující sledovat a analyzovat veškerou komunikaci mezi prohlížečem a serverem, díky čemuž může tester detailněji porozumět, jak komunikace probíhá. Pomocí tzv. *breakpointů* lze odesílané požadavky navíc i zachytávat a modifikovat. Tester tedy může v určitém bodě komunikaci zastavit a provést úpravy na odesílaných datech. Tímto způsobem lze simulovat různé útoky a ověřit, zda aplikace reaguje správně na neočekávané vstupy.[48]

2.3.3 Metasploit framework

Metasploit framework [49] je open-source platforma, poskytující rozsáhlou a vysoce flexibilní sadu nástrojů pro penetrační testování.

Základním stavebním kamenem této platformy jsou tzv. moduly – části softwaru, které poskytují specifickou funkcionalitu. Takovou funkcionalitou může být například sken konkrétní služby, překonání bezpečnostních opatření, nebo spuštění škodlivého kódu.

Dle typu prováděné akce jsou moduly podle [50] děleny na následující základní typy:

- **Auxiliary modules**, které poskytují různé pomocné funkce, využitelné v průběhu penetračního testování. Mezi jejich typické úkoly patří například oskenování cíle, sběr informací, provedení DoS útoku atd. Tyto moduly neprovádí samotnou exploitaci cílového systému

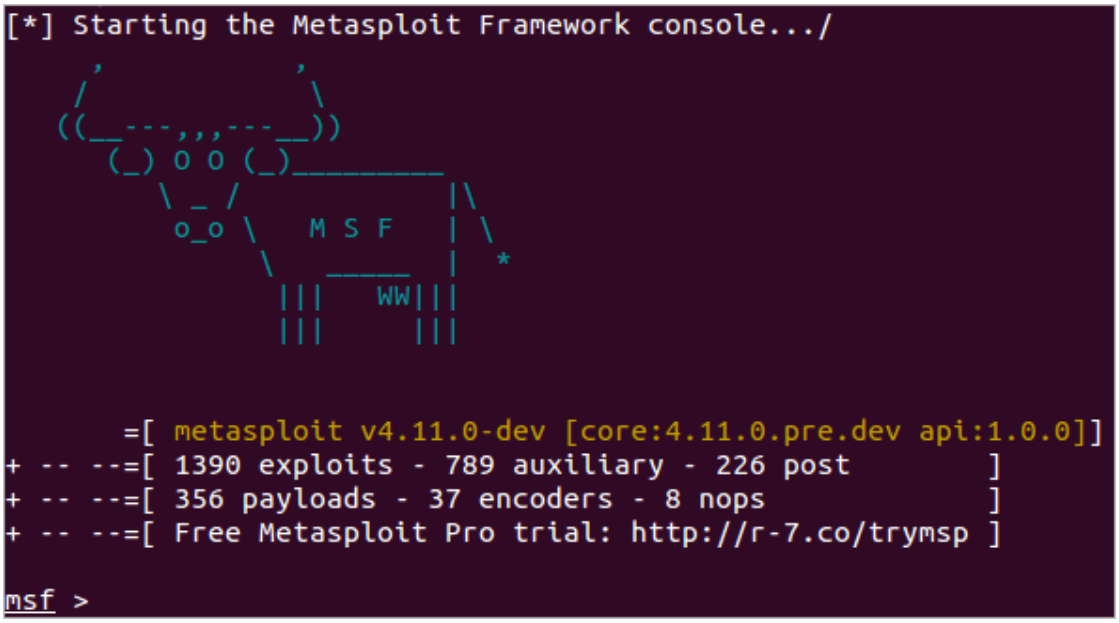
- **Exploit modules**, které slouží k zneužití zranitelností nalezených v cílovém systému. Toho docílí spuštěním tzv. *payloadu* – škodlivého kódu, který je mířen přímo na konkrétní zranitelnost. Výsledkem úspěšného běhu takových modulů je nejčastěji získání přístupu do cílového systému.

- **Encoder modules**, sloužící k zakódování exploitů a payloadů. Hlavním cílem těchto modulů je obfuskace škodlivého kódu a skrytí skutečné povahy útoku s cílem překonat použité bezpečnostní mechanismy.

- **Payload modules**, neboli moduly obsahující payloady. Tyto moduly zapouzdřují škodlivý kód, který je spuštěn při úspěšné exploitaci cílového systému a definuje akce, které mají být na cílovém systému provedeny. Takovou akcí bývá často například vytvoření tzv. *reverse shellu*, pomocí něhož může útočník napadený systém ovládat.

- **Nop modules**, generující sekvence prázdných instrukcí. Cílem těchto modulů je vytvořit tzv. *nop-sled* – sekvenci instrukcí „no-operation“, která je umístěna do paměti před škodlivým kódem. Pokud program během svého provádění skočí na adresu, kde se tato sekvence nachází, dostane se sekvenčně až ke škodlivému kódu. Tato metoda se používá při útoku na přetečení bufferu, tzv. *buffer overflow*.

- **Evasion modules**, umožňující generovat payloady, které mají za cíl vyhnout se detekci a obejít ochranné mechanismy, jako je například firewall, IDS/IPS systémy, antivirová ochrana atd.



Obrázek 2.2 Úvodní obrazovka interaktivní konzole Metasploit frameworku (Msfconsole)

- **Post modules**, zaměřující se na činnosti prováděné po úspěšné exploitaci cílového systému. Mezi typické úkoly těchto modulů patří například sběr informací o napadeném sytému, vytváření záloh, instalace malwaru, rozšiřování přístupu a spoustu dalších.

2.3.4 Nikto

Nikto [51] je skener, určený pro testování webových serverů a hledání potenciálních zranitelností. Zaměřuje se na detekci nebezpečných souborů, programů a problémů specifických pro konkrétní servery. Dále dokáže kontrolovat, zda server nepoužívá zastaralé komponenty a zda je správně nakonfigurovaný. Mezi často detekované chyby v konfiguraci patří například špatně nastavené HTTP hlavičky, nebo veřejně dostupné citlivé soubory/adresáře.

Je důležité upozornit, že nástroj byl navržen tak, aby provedl testování webového serveru co nejrychleji a bez použití dodatečných technik je tak lehce detekovatelný cílovým serverem.

Použití nástroje je jednoduché, stačí zadat cílovou adresu webové aplikace a spustit skenování. Základní příklad užití by tedy mohl vypadat takto:

```
nikto -h www.adresacile.cz
```

Kromě základního použití je dostupné nemalé množství přepínačů, které umožňují přizpůsobit sken potřebám testera a upravit tak například maximální délku skenu, formát výstupu, cílové porty a spoustu dalších parametrů skenu.

Kapitola 3
Aplikace Uniqway

Tato kapitola má za cíl stručně obeznámit čtenáře se systémem Uniqway a popsat důležité části jeho struktury.

3.1 Architektura systému

Uniqway se skládá z několika komponent, které mezi sebou navzájem komunikují a tvoří dohromady celý systém. Hlavní struktura systému jde nejlépe vidět na obrázku 3.1. Těmi nejzajímavějšími a nejdůležitějšími komponentami, pro účely této práce, jsou následující.

3.1.1 uniqway-server

uniqway-server je nejdůležitější částí systému, jelikož tvoří backend celé aplikace. Pro jeho vývoj byl použit open-source framework Play a programovací jazyk Java. Uniqway-server definuje a vykonává všechny funkce, které jsou uživatelům poskytovány. V rámci této komponenty je, mimo jiné, definováno i API, které server vystavuje a pomocí kterého komunikuje se všemi ostatními komponentami. Dalšími funkcemi serveru, zajímavými z hlediska bezpečnosti, jsou například zajišťování autentizace, nebo definice uživatelských rolí, využívaných pro autorizaci uživatelů.

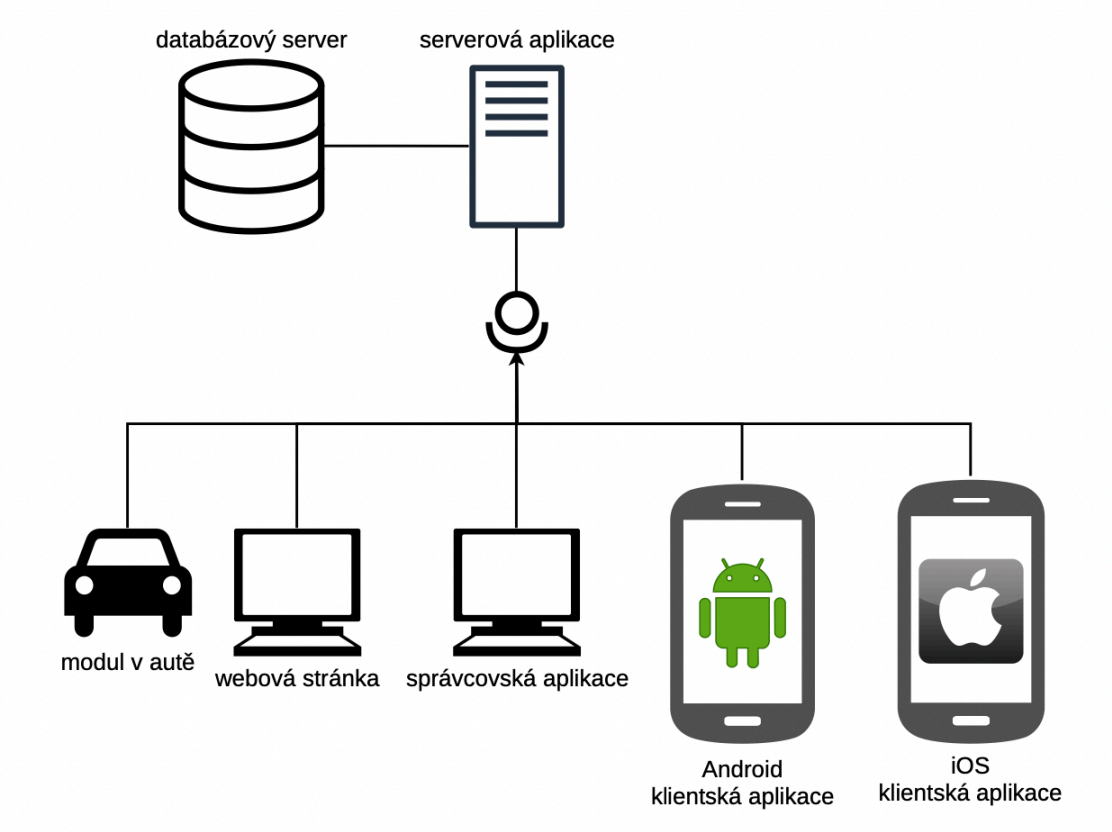
3.1.2 uniqway-client-app

uniqway-client-app je klientská mobilní aplikace, dostupná pro telefony s operačním systémem iOS a Android. Prostřednictvím této aplikace je přihlášeným uživatelům umožněno využívat veškeré klientské funkce, které Uniqway poskytuje. Mezi takové funkce patří převážně všechny činnosti spojené s procesem vypůjčení vozidla (prohlížení dostupných vozidel, rezervace, vypůjčení a následné vrácení vozidla), ale také například prohlížení a zakoupení produktů Uniqway v obchodu s odměnami.

Vzhledem k tomu, že jsou všechny funkce, poskytované klientskou mobilní aplikací prováděny čistě v podobě volání endpointů vystavovaných v Uniqway API, jsou tyto funkce otestovány v rámci kapitoly 5

3.1.3 uniqway-static-web

uniqway-static-web je webová aplikace, nacházející se na adrese *www.uniqway.cz*. Slouží k poskytování informací a dokumentů dostupných veřejnosti. Těmi jsou například VOP, ceník, zá-



Obrázek 3.1 Architektura systému Uniqway

[52]

kladní informace o projektu atd. Mimo to zde probíhá proces registrace a verifikace uživatele.

Aplikace využívá framework Nuxt.js, který umožňuje generovat statické stránky. Ty ze své podstaty zmenšují prostor pro útok, jelikož nekomunikují se serverem a zvyšují tak bezpečnost aplikace.

3.1.4 uniqway-admin-app

uniqway-admin-app je webová aplikace přístupná pouze správcům systému Uniqway. Pomocí této aplikace mohou administrátoři pracovat s daty uloženými na serveru a spravovat tak například objednávky v obchodě s odměnami, uživatelské účty, nebo vozidla, která Uniqway poskytuje.

3.2 Uživatelské role

Používání rolí je významnou součástí návrhu a provozu většiny aplikací. Uživatelské role přiřazují různá přístupová práva různým skupinám uživatelů v systému. Správně definované role jsou pak klíčové pro minimalizaci rizik spojených s neoprávněným přístupem a zároveň výrazně zjednodušují správu oprávnění uživatelů v systému.

Jak již bylo zmíněno, Uniqway definuje uživatelské role v rámci komponenty uniqway-server. Na základě těchto rolí se následně rozhoduje o udělení, či odepření přístupu k funkcím, které aplikace poskytuje. Jednotlivými rolemi, které systém Uniqway využívá jsou:

1. **user** – Tato role je určena pro běžné uživatele systému Uniqway a má z definovaných rolí nejnižší oprávnění. Uživatel tuto roli získá poté, co se do systému zaregistruje a verifikuje skrz statickou webovou aplikaci. Uživatelé s touto rolí mají přístup do klientské aplikace uniqway-client-app a ke klientským funkcím, které tato aplikace poskytuje.

2. **carAdmin** – Role carAdmin je určena pro administrátory automobilů. Uživatelé s touto rolí mají přístup do administrátorské aplikace, kde mají práva omezená na provádění akcí týkajících se správy automobilů. Mohou tak například upravovat nabídku dostupných automobilů, spravovat moduly, které se v nich nachází, nebo rezervovat automobily, které by měly projít údržbou.

3. **userAdmin** - Role userAdmin je určena pro administrátory uživatelských účtů. Uživatelé s touto rolí mají taktéž přístup do administrátorské aplikace, ve které mohou, narozdíl od role carAdmin, provádět akce spojené s administrací účtů. Mezi tyto akce patří například správa objednávek z obchodu s odměnami, schvalování registrovaných účtů, nebo přidělování slev uživatelům.

4. **admin** - Role admin je nejprivilegovanější rolí, kterou může mít uživatel přiřazen. Poskytuje nejvyšší množství oprávnění a uživatel díky ní může například spravovat podporu jednotlivých verzí aplikace, nebo přiřazovat a odebírat role jiným uživatelům. Kromě toho má přístup ke všem funkcím, ke kterým mají přístup ostatní, výše uvedené role.

Během průzkumu uživatelských rolí a jejich privilegií nebylo objeveno žádné pochybení. Jednotlivé role jsou nadefinovány rozumně a tak, aby měly přístup pouze k těm funkcím, které potřebují využívat. Během testování bylo také pro jednotlivé role vyzkoušeno volání všech endpointů, vyžadujících vyšší privilegia. Žádné toto volání nebylo úspěšné a nedošlo tak k neoprávněnému přístupu.

Kapitola 4

Black-box testování systému Uniqway

Penetrační testování systému je komplexní proces, zahrnující spoustu různých akcí a jeho detailní popis by byl pro účely této práce zbytečný. V této kapitole je tedy jen lehce nastíněn průběh black-box penetračního testování systému Uniqway. Následně, jsou uvedeny slabiny, které byly během testování nalezeny, přiřazeny k jednotlivým bodům již zmiňovaného seznamu OWASP TOP 10 a zhodnoceny dle metodiky CVSS.

4.1 Skenování portů

Skenování portů je typicky jednou z prvních akcí, která je při penetračním testování prováděna. Jak již bylo zmíněno, umožňuje získat informace o službách, které v systému běží, a podle nich následně určit postup útoku.

Ke skenování portů byl použit nástroj Nmap který nalezl 2 otevřené porty – 80/tcp a 443/tcp. Konkrétní použitý příkaz je uveden níže

```
nmap -p0-65535 -A -T4 -oA uniqwayscan-%D uniqway.cz
```

4.1.1 Port 80/tcp

Na tomto portu běží HTTP server Apache. Server nevystavuje zbytečně detailní informace a nepodařilo se tak zjistit jeho konkrétní verzi. To útočníkovi ztěžuje hledání zranitelností, jelikož se spousta z nich nachází jen v konkrétních verzích služby. V tomto ohledu je tedy server správně nakonfigurován.

Použitím nástroje Nikto byly ovšem nalezeny slabiny v nastavování bezpečnostních HTTP hlaviček. Konkrétně nejsou nastavovány následující hlavičky:

- **X-XSS-Protection-Header**, která umožňuje nastavit konfiguraci XSS filtrů zabudovaných v prohlížeči a slouží tak k prevenci útoků typu Cross-Site scripting. Správné nastavení říká prohlížeči, aby aktivně blokoval pokusy o takové útoky. Nastavování této hlavičky je ovšem vhodné pouze jako ochrana pro uživatele starších webových prohlížečů. Moderní webové prohlížeče ji ignorují a poskytují tuto ochranu v rámci nastavení hlavičky CSP. Absence této hlavičky tak nebude považována za zranitelnost. [53]
- **X-Content-Type-Options**, která zajišťuje, aby prohlížeč neodhadoval typ načítaného obsahu a řídil se tím, jak je nastavena tato hlavička. Její nastavení tak může předejít případům,

kdy by se útočníkovi podařilo do místa s nesprávným ošetřením vstupu vložit škodlivý skript, který by mohl být při neověřování typu obsahu nesprávně zpracován a následně serverem spuštěn. [54]

- **X-Frame-Options**, která informuje o tom, zda smí být stránka zobrazována v rámu jiné stránky pomocí odpovídajících HTML tagů (např. <frame>). Zobrazování může být zakázáno úplně, nebo jen omezeno na konkrétní definované domény. Tato hlavička slouží jako ochrana proti tzv. clickjacking útokům, které spočívají ve vložení obsahu důvěryhodné stránky na jiné, podvodné stránce. Stránka se následně na první pohled jeví jako důvěryhodná a oběť je tak jednodušeji přesvědčitelná například k vložení citlivých informací, ke kterým následně získá útočník přístup. [55]

4.1.2 Port 443/tcp

Na tomto portu běží taktéž Apache HTTP server, který ale narozdíl od předchozího serveru používá protokol SSL. Ten zajišťuje šifrovanou komunikaci s klientem a zvyšuje tak odolnost proti tzv. Man in the middle útokům, zaměřeným na odposlech komunikace.

I zde byly pomocí nástroje Nikto objeveny slabiny v nastavování bezpečnostních HTTP hlaviček. Kromě výše zmíněných k nim přibyly ještě následující dvě hlavičky:

- **Strict-transport-security header**, která informuje prohlížeč o tom, aby při návštěve stránky využíval po definovanou dobu pouze zabezpečené spojení pomocí protokolu HTTPS. Jakékoliv pokusy o přístup pomocí HTTP jsou automaticky přesměrovány na HTTPS. [56]
- **Expect CT header**, která umožňuje kontrolovat vyhodnocování SSL certifikátu v *Certificate Transparency* a v případě problémů s ověřením (například použití podvodného certifikátu) odesílat reporty. Tato hlavička už ovšem není v současnosti podporována a její absence tak nebude považována za zranitelnost. [57]

4.1.3 Alternativní domény

Během skenování portů byly dále objeveny následující domény, na kterých běží aplikace, spadající pod Uniqway:

- **admin.uniqway.cz**
- **preview.uniqway.cz**
- **test.uniqway.cz**
- **staging.uniqway.cz**
- **wiki.uniqway.cz**

Je důležité podotknout, že k jejich obsahu mají přístup pouze přihlášení uživatelé, kteří mají přiřazena potřebná přístupová práva. V opačném případě se není možné dostat dále, než na přihlašovací obrazovku.

4.2 Výsledky black-box testování

Výsledky black-box penetračního testu aplikace Uniqway byly rozděleny do jednotlivých bodů, již zmiňovaného seznamu OWASP TOP 10.

4.2.1 Broken Access Control

V aplikaci se nepodařilo najít žádnou zranitelnost tohoto typu. Co se týče statické webové aplikace Uniqway, slouží primárně k poskytování základních informací o projektu a k registraci a následnému ověření uživatele. Nenachází se na ní tedy stránky s přístupem omezeným pouze pro uživatele se specifickými oprávněními.

Co se týče ostatních nalezených domén, přístup k nim je podmíněn přihlášením, bez kterého není možné se na jakoukoliv stránku na dané doméně dostat.

4.2.2 Cryptographic Failures

Veškerá komunikace probíhá pomocí zabezpečeného protokolu HTTPS a je šifrována. K tomu aplikace využívá síťový protokol TLS v1.2. Podporuje ovšem několik sad šifer, které jsou dle [58] považovány za slabé. Těmito sadami šifer jsou:

- **TLS_DHE_RSA_WITH_AES_128_GCM_SHA256**
- **TLS_DHE_RSA_WITH_AES_256_GCM_SHA384**
- **TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256**
- **TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384**

Uvedené sady šifer jsou náchylné na různé typy útoků, které útočníkovi umožňují je v rozumném čase prolomit.

4.2.3 Injection

Použitý framework Vue.js poskytuje základní ochranu proti útokům typu XSS. Co se týče útoků typu SQL injection, může na ně být aplikace náchylná v případě nesprávně navrženého zdrojového kódu, pracujícího s SQL databází. Během testování uživatelských vstupů v aplikaci nebyla žádná taková chyba nalezena. Aplikace správně zpracovává speciální znaky a vstupy, které by mohly způsobit problémy a neprojevilo se, že by na tento typ útoků byla náchylná.

4.2.4 Insecure Design

Co se týče návrhu aplikace, nebyly zde z hlediska bezpečnosti nalezeny kritické nedostatky. I tak ovšem stojí za zmínku nesprávná kontrola formátu informací vyplněných v políčkách registračního formuláře. Formulář vůbec nekontroluje například formát a správnost zadaného telefonního čísla, nebo čísla průkazu. To v případě nesprávně nastavené limitace počtu dotazů výrazně zjednodušuje automatizované vytváření účtů a umožňuje tak útočníkovi zahltit databázi falešnými uživatelskými účty.

4.2.5 Security Misconfiguration

Kromě výše již zmíněného nesprávného nastavování HTTP hlaviček nebyla v aplikaci nalezena žádná další slabina z této kategorie. Server je nakonfigurován správně a nevystavuje zbytečně nepotřebné informace, nebo porty, které by útočníkovi akorát zvětšovaly prostor pro útok.

4.2.6 Vulnerable and Outdated Components

Pomocí nástroje Owasp ZAP bylo detekováno použití zastaralé verze knihovny JQuery ve verzi 1.12.1, které teoreticky přináší dle [59] možnosti útoků typu:

- Cross-site scripting (CVE-2020-11022, CVE-2020-11023)
- Prototype pollution (CVE-2019-11358)

. Během testování webové aplikace ovšem nebylo nalezeno místo, kde by se těchto zranitelností dalo reálně zneužít.

4.2.7 Identification and Authentication Failures

Co se týče statické aplikace, vzhledem k tomu, že nijak neumožňuje přihlášení uživatele a je kompletně dostupná každému, nebylo v ní nalezeno pochybení v oblasti autentizace uživatele.

Na ostatních zmiňovaných doménách ovšem nedochází k limitaci odesílání požadavků na přihlášení uživatele. Zároveň přihlašování není nijak omezené na počet pokusů s nesprávnými údaji, což útočníkovi umožňuje neomezený útok hrubou silou.

4.2.8 Software and Data Integrity Failures

Webová aplikace stahuje své závislosti pomocí nástroje npm. Data a soubory, které se na stránkách získávají jsou z drtivé většiny statické a během testování nebylo nalezeno žádné místo, kde by aplikace získávala data z externích, neověřených zdrojů, které by v případě kompromitace mohly způsobit bezpečnostní riziko pro aplikaci Uniqway.

4.2.9 Security Logging and Monitoring Failures

Dotazy, které jsou v rámci funkcí dostupných na statické webové aplikaci volány, jsou monitorovány a je možné je skrz dashboard sledovat. Všechny důležité prováděné akce jsou zároveň logovány a je tak možné, v případě potřeby, zpětně dohledat podstatná data, která jsou s nimi spojená. V tomto ohledu tedy aplikace není nijak zranitelná.

4.2.10 Server Side Request Forgery

Ve statické webové aplikaci nebylo během testování nalezeno žádné místo, kde by aplikace nesprávně validovala uživatelsky zadanou URL a není tedy vůči útoku typu Server Side Request Forgery zranitelná.

4.3 Zhodnocení nalezených slabin

Vzhledem k tomu, že webová aplikace Uniqway je z drtivé většiny informativní, obsahuje statická data a nezahrnuje akce, omezené na přihlášené uživatele, nemá na ní útočník nijak veliký prostor pro útok. Místa, kde tento prostor má, byla otestována, ale nebyla nalezena žádná kritická slabina, která by výrazně ohrožovala bezpečnost systému. I tak stojí za zmínku slabiny uvedené níže.

Podpora slabých šifer

CVSS Skóre: 3.1 (Low)

Podpora slabých šifer může vést k rozšifrování komunikace zachycené útočníkem a získání přeposílaných dat ve formě *plaintextu*. I přesto, že je zneužití jejich slabin mnohdy velmi náročné, doporučuje se, tyto šifry preventivně nepodporovat, pokud pro to není konkrétní důvod.

Chybějící hlavička X-Content-Type-Options

CVSS Skóre: 3.5 (Low)

V průběhu testování webové aplikace Uniqway nebylo objeveno žádné místo, kde by mohla být absence této HTTP hlavičky zneužita. I přesto by bylo vhodné hlavičku preventivně nastavovat a vyvarovat se tak potenciálním budoucím problémům. Úprava je jednoduchá a spočívá v nastavení hodnoty **X-Content-Type-Options** na hodnotu **nosniff** v konfiguraci serveru.

Chybějící hlavička X-Frame-Options

CVSS Skóre: 4.3 (Medium)

Nenastavování hlavičky X-Frame-Options na webové stránce Uniqway může zapříčinit primárně únik osobních informací. Na stránce se totiž nachází registrační formulář, kam uživatel mimo jiné nahrává i citlivá data, jako jsou například osobní doklady. V případě, že by byl tento formulář zobrazen v rámu nějaké podvodné stránky, získal by útočník, po odeslání formuláře uživatele, přístup k těmto citlivým údajům. Úprava je opět jednoduchá a stejně jako v předchozím případě spočívá v nastavení vhodné hodnoty hlavičky **X-Frame-Options** v konfiguraci serveru.

Chybějící hlavička Strict-transport-security

CVSS Skóre: 3.3 (Low)

Absence hlavičky Strict-transport-security může útočníkovi za specifických podmínek umožnit odposlech komunikace mezi uživatelem a serverem v nešifrované podobě a způsobit tak únik přeposílaných informací ve formě *plaintextu*. Úprava, stejně jako v předchozích případech, spočívá v nastavení vhodné hodnoty hlavičky **Strict-transport-security** v konfiguraci serveru.

Použití zranitelné verze knihovny JQuery

CVSS Skóre: 4.3 (Medium)

I přesto, že v aplikaci nebylo nalezeno místo, kde by se zranitelná část knihovny dala zneužít, bylo by v rámci prevence vhodné prozkoumat možnost aktualizace a případně aktualizovat používanou verzi knihovny JQuery na verzi **>= 3.5.0**, která již uvedené zranitelnosti neobsahuje.

Absence limitace požadavků na přihlášení uživatele

CVSS Skóre: 7.5 (High)

Jak již bylo zmíněno, na alternativních doménách zmíněných v sekci 4.1.3 nedochází k žádné limitaci uživatele na počet požadavků na přihlášení a zároveň není nijak omezen počet pokusů o přihlášení s nesprávnými údaji. To umožňuje útočníkovi útok hrubou silou na uživatelské údaje. Vzhledem k tomu, že se navíc na daných doménách nacházejí citlivé informace, ať už jsou to informace o architektuře systému (**wiki.uniqway.cz**), nebo administrátorské rozhraní (**admin.uniqway.cz**), představuje pro systém možnost útoku hrubou silou o to větší riziko.

Zde by bylo vhodné počet možných volání omezit pomocí tzv. *rate limitingu*, který uživatelům umožňuje na stejný endpoint poslat jen omezený počet požadavků za určitý časový úsek. Dalším případným řešením je, po určitém počtu pokusů o přihlášení s nesprávnými uživatelskými údaji účet dočasně zablokovat. To ovšem přináší další riziko v podobě znepřístupnění služby pro

legitimního uživatele, který by se do účtu chtěl během blokace přihlásit. Toto řešení tedy není ideální a nedoporučuje se.

Zranitelnost	CVSS skóre
Podpora slabých sad šifer	3.1 (Low)
Chybějící hlavička X-Content-Type-Options	3.5 (Low)
Chybějící hlavička X-Frame-Options	4.3 (Medium)
Chybějící hlavička Strict-transport-security	3.3 (Low)
Použití zranitelné verze knihovny JQuery	4.3 (Medium)
Absence limitace požadavků na přihlášení uživatele	7.5 (High)

Tabulka 4.1 Shrnutí zranitelností nalezených při black-box testování

Kapitola 5

Testování bezpečnosti Uniqway API

Tato kapitola se zaměřuje na představení a otestování bezpečnosti API, které systém Uniqway vystavuje. Výsledky testování budou rozděleny dle již dříve zmiňovaného seznamu OWASP API Security Top 10.

5.1 API cesty

Systém Uniqway poskytuje produkční API na adrese https://www.uniqway.cz/api a je rozděleno na následující části:

- **/client**, poskytující uživatelské endpointy pro klientskou mobilní aplikaci. Jsou zde tedy zahrnuty například endpointy pro přihlášení uživatele, rezervaci automobilů, nebo odemykání a zamykání zapůjčených aut.
- **/demo**, poskytující endpointy pro získávání informací zobrazovaných na statické webové aplikaci. Zde můžeme najít například endpointy pro získání novinek, recenzí, nebo výpočet ceny půjčení konkrétního vozidla.
- **/admin**, poskytující endpointy pro administrátory systému Uniqway. Zde můžeme najít například endpointy sloužící pro správu vozidel, uživatelů, nebo rezervací.
- **/v2**, poskytující novější verzi několika endpointů z částí /client a /demo.
- **/gp**, poskytující 1 endpoint pro správu globálních plateb.
- **/citymove**, poskytující API endpointy poskytující informace o vozidlech pro systémy třetích stran.

5.2 Výsledky testování API

Výsledky testování bezpečnosti API byly opět rozděleny do jednotlivých bodů seznamu OWASP API TOP 10.

5.2.1 Broken Object Level Authorization

V endpointech klientského Uniqway API se identifikátor objektu, jakožto parametr při volání používá velmi sporadicky. Byla zde ale nalezena chyba u endpointu, sloužícího pro získání informací ke konkrétnímu produktu v obchodě s odměnami – **/client/shop/products/:id**. Tento endpoint využívá jako parametr identifikátor produktu. Každý produkt v databázi si u sebe drží atribut *visible*, který značí, zda má být pro klienty produkt v obchodě viditelný, nebo ne. Zavolání tohoto endpointu s identifikátorem produktu, který by měl být v obchodě skrytý proběhne v pořádku a vrátí uživateli informace o daném produktu, i přesto, že by ho správně neměl moct vidět.

5.2.2 Broken Authentication

V Uniqway API existují 2 endpointy pro přihlášení uživatelů:

- **/client/login**, sloužící k přihlášení klientů
- **/admin/login**, sloužící k přihlášení administrátorů systému

Ani jeden z těchto endpointů nijak nelimituje uživatele ve frekvenci volání, ani v neúspěšných pokusech o přihlášení. Díky tomuto pochybení je útočníkovi umožněno provádět útok hrubou silou na uživatelské údaje.

5.2.3 Broken Object Property Level Authorization

Žádný z API endpointů nevrací v odpovědích nadbytečná data, která by byla posléze filtrována pryč. Co se týče endpointů, sloužících pro modifikaci objektů, probíhá při jejich volání dostatečná kontrola parametrů v dotazu a není tak umožněno modifikovat data, která by modifikována být neměla. V této kategorii zranitelností tedy nebylo objeveno žádné pochybení.

5.2.4 Unrestricted Resource Consumption

API nijak nelimituje uživatele v počtu dotazů, což může způsobovat problémy ve více ohledech. Jedním z nich je již výše zmiňovaná možnost útoku hrubou silou přes endpointy sloužící k autentizaci. Dalšími problémy může být omezená dostupnost samotného API, způsobená jeho nadměrným vytížením, nebo zahlcení databáze skrz endpointy, které do ní ukládají data. Příkladem takového scénáře může být vytvoření obrovského množství uživatelů pomocí endpointu **/users/register** a následné odeslání ověřovacích dokumentů pomocí **/users/verification**. Ten umožňuje pro každého zaregistrovaného uživatele nahrát až 10MB fotografie zakódované pomocí Base64 a při vysokém počtu volání tak při ukládání do databáze databázi poměrně rychle zaplnit. Ukládání těchto dat bylo postupem času přesunuto z databáze do cloudového úložiště a tak už tento konkrétní případ naštěstí nehrozí.

5.2.5 Broken Function Level Authorization

U privilegovaných endpointů je při jejich volání zajištěna autorizace na základě výše popisovaných uživatelských rolí. Během testování bylo ověřeno, že žádný z těchto endpointů neumožňuje volání uživatelům, vlastnícím roli s nižšími oprávněními, než je požadováno. V tomto ohledu tedy nebyly nalezeny žádné zranitelnosti.

5.2.6 Server Side Request Forgery

Jediný endpoint, který přijímá jako parametr adresu od uživatele je **/assets/*file** , který slouží k získání souborů, které Uniqway veřejně vystavuje. Endpoint je nastaven tak, aby předanou adresu validoval a bylo možné získávat pouze soubory z adresáře public, který k uchovávání těchto souborů slouží. Pokusy o zavolání, například s doplněním relativní cesty k souborům mimo adresář byly tedy neúspěšné. V této kategorii nebyla v aplikaci objevena žádná zranitelnost.

5.2.7 Security Misconfiguration

Zde bylo nalezeno pochybení v konfiguraci Json Web Tokenu, který je používán k autorizaci uživatele, volajícího API endpointy. Platnost tohoto tokenu je nastavena na 259200 minut, což je velmi vysoká hodnota. Při odhlášení uživatele navíc nedochází k jeho zneplatnění a je možné ho nadále využívat k úspěšné autorizaci. To ještě více zvyšuje bezpečnostní riziko způsobené jeho dlouhou trvanlivostí. Pokud by se tak útočníkovi podařilo JWT-Token ukradnout, mohl by ho téměř bez omezení využívat.

5.2.8 Lack of Protection from Automated Threats

Během testování nebyly nalezeny procesy, u kterých by jejich automatizované provádění útočníka zvýhodnilo a představovalo hrozbu, nebo mělo jakýkoliv negativní ekonomický dopad na systém Uniqway. V tomto ohledu nebylo v API nalezeno žádné pochybení.

5.2.9 Improper Inventory Management

API endpointy systému Uniqway slouží jen pro interní potřeby aplikace a jejich dokumentace tak není dostupná veřejně, ale jen uživatelům s přiděleným přístupem. API má téměř všechny endpointy aktuální a využívané až na následující výjimky, pro které existuje /v2 verze:

- **/demo/cars/detailed**
- **/demo/cars/:id/detailed**
- **/client/cars/detailed**
- **/client/cars/:id/detailed**
- **/client/cars/reserved**

I přesto, že staré verze endpointů nevrací detailnější informace, než ty nové, jsou stále dostupné k volání. Bylo by lepší jejich podporu odstranit a poskytovat pro každý endpoint jen jednu verzi.

5.2.10 Unsafe Consumption of APIs

Uniqway využívá hned několik API třetích stran. Mezi ně patří například API služby Mailchimp, nebo Škoda API. Veškerá komunikace mezi API systému Uniqway a API třetích stran probíhá skrz zabezpečený komunikační kanál. Data, která jsou z nich vracena, jsou navíc validována a zpracována bezpečným způsobem. V tomto ohledu tedy nebylo nalezeno žádné pochybení.

5.3 Zhodnocení nalezených slabin

5.3.1 Získání skrytých produktů

CVSS Skóre: 4.3 (Medium)

Zneužití této slabiny nemá potenciál být pro systém Uniqway nebezpečné z technického hlediska, nýbrž z ekonomického. Vzhledem k tomu, že každý uživatel může získat informace o neveřejných produktech, mohou být tyto informace zneužity například konkurenční firmou, což může následně vést ke ztrátě konkurenční výhody, nebo i finanční újmě. Bylo by tedy dobré přístup k takovým produktům kontrolovat a neumožňovat ho každému.

Tato slabina může být mitigována jednoduše autorizací na základě rolí, které Uniqway využívá, v případech, kdy takto uživatel volá endpoint **/client/shop/products/:id** a snaží se přistupovat k neveřejnému produktu.

5.3.2 Absence limitace požadavků na API

CVSS Skóre: 7.5 (High)

Jak již bylo řečeno, neomezování uživatelů v požadavcích na API představuje hrozbu hned v několika ohledech. Mezi ně patří zahlcení API velkým množstvím požadavků s cílem omezit jeho dostupnost, zahlcení databáze dále nijak nevyužitými daty, nebo prolamování uživatelských údajů pomocí útoku hrubou silou.

Tato slabina lze mitigovat zavedením rate limitingu – omezení počtu požadavků, které může uživatel provádět za daný časový úsek. U veřejných endpointů pak lze zavést alespoň limitaci na základě IP adresy. U některých endpointů sice není absence limitace kritická, ale například právě u autentizačních endpointů /client/login a /admin/login představuje značnou hrozbu a může vést k neoprávněnému přístupu a zneužití účtů jiných uživatelů. Bylo by tedy dobré u endpointů zhodnotit, jak často je uživatel může potřebovat volat a rozumně jim limitaci nastavit.

5.3.3 Nesprávná konfigurace JWT-Tokenu

CVSS Skóre: 6.5 (Medium)

JWT-Token je v aplikaci Uniqway využíván k autorizaci přihlášených uživatelů volajících API endpointy. Ve chvíli, kdy se uživatel odhlásí, nebo přihlásí a je mu vygenerován nový token, původní token se nezneplatňuje a je platný až do uplynutí jeho platnosti. Nastavená hodnota je tedy důležitým faktorem bezpečnosti tokenu.

Vzhledem k tomu, že Uniqway nastavuje platnost JWT-Tokenu na 259200 minut (180 dní), je výrazně zvýšeno riziko spojené s jeho únikem. I přesto, že veškerá komunikace probíhá šifrovaně přes zabezpečený protokol HTTPS a není jednoduché token získat pouhým odposlechem komunikace, existují další metody, pomocí kterých by se útočník k tokenu mohl dostat.

Mitigace této slabiny je triviální a spočívá v nastavení rozumné nižší hodnoty platnosti tokenu v konfiguraci aplikace.

Zranitelnost	CVSS skóre
Získání skrytých produktů	4.3 (Medium)
Absence limitace požadavků na API	7.5 (High)
Nesprávná konfigurace JWT-Tokenu	6.5 (Medium)

Tabulka 5.1 Shrnutí zranitelností nalezených při testování API

Závěr

Cílem této bakalářské práce bylo seznámit čtenáře s bezpečnostními riziky webových aplikací a základy penetračního testování obecně a provést bezpečnostní analýzu carsharingového systému Uniqway. Každému bodu ze zadání jsem se náležitě pověnoval a cíle práce považuji za splněné.

V prvních dvou kapitolách jsem vypracoval teoretický základ, který seznamuje čtenáře s problematikou bezpečnosti webových aplikací a uvádí ho do světa penetračního testování. Tento teoretický základ je zpracován tak, aby byl čitelný a pochopitelný i pro čtenáře, kteří s kybernetickou bezpečností nemají příliš mnoho zkušeností.

Ve třetí kapitole je uveden stručný výstup analýzy architektury systému Uniqway, kterou jsem provedl. Jsou zde uvedeny a popsány komponenty, které pro mě z hlediska bezpečnostní analýzy byly nejpodstatnější. Dále jsou zde popsány také uživatelské role, které jsou pro Uniqway klíčové, co se týče autorizace uživatelů

Ve čtvrté a páté kapitole je pak stěžejní výstup praktické části práce – penetračního testování. Bylo provedeno jak black-box testování webových aplikací, které Uniqway provozuje, z pozice útočníka, který o architektuře systému nemá předem žádné povědomí, tak testování API se základní znalostí o architektuře systému. Výsledky byly zhodnoceny dle standardizované metodiky CVSS a u nalezených slabin, byl navržen způsob, jak je lze mitigovat.

Systém Uniqway se ukázal jako relativně dobře navržený a zabezpečený a nebyla v něm nalezena žádná zranitelnost, která by ho kriticky ohrožovala. I přesto byly nalezeny slabiny, které pro systém a jeho uživatele představují nemalé riziko a bylo by vhodné je mitigovat.

Vzhledem k tomu, že byl provoz systému Uniqway v průběhu psaní této práce přerušen, není jejich mitigace v blízké době plánována. I tak ale výstup práce obsahuje cenné informace, jelikož popisuje bezpečnostní stav, v jakém systém během jeho běhu byl a v jakém je napsán. Výstup bude předán vývojářům systému a pokud by se provoz systému v budoucnu obnovoval, budou tyto slabiny opraveny.

Bibliografie

1. HOLZMAN, Ondřej. *Škoda Auto spustila nový studentský carsharing Uniqway, který vyvinuli sami studenti* [online]. [cit. 2023-02-28]. Dostupné z: https://cc.cz/skoda-auto-spustila-novy-studentsky-carsharing-uniqway-ktery-vyvinuli-sami-studenti/.
2. KEMP, Simon. *DIGITAL 2023: GLOBAL OVERVIEW REPORT* [online]. [cit. 2023-02-28]. Dostupné z: https://datareportal.com/reports/digital-2023-global-overview-report.
3. ANON. *Zákon č. 181/2014 Sb. Zákon o kybernetické bezpečnosti* [online]. [cit. 2023-02-28]. Dostupné z: https://www.zakonyprolidi.cz/cs/2014-181.
4. COLOHOUSE LLC. *Infographic: Defense-in-Depth* [online]. [cit. 2023-03-03]. Dostupné z: https://colohouse.com/infographic-defense-in-depth/.
5. *OWASP* [online]. [cit. 2023-02-28]. Dostupné z: https://owasp.org.
6. OWASP FOUNDATION. *OWASP TOP TEN* [online]. [cit. 2023-02-28]. Dostupné z: https://owasp.org/Top10/.
7. OWASP FOUNDATION. OWASP TOP TEN. *A01:2021 – Broken Access Control* [online]. 2023 [cit. 2023-02-28]. Dostupné z: https://owasp.org/Top10/A01_2021-Broken_Access_Control/.
8. OWASP FOUNDATION. OWASP TOP TEN. *A02:2021 – Cryptographic Failures* [online]. 2023 [cit. 2023-02-28]. Dostupné z: https://owasp.org/Top10/A02_2021-Cryptographic_Failures/.
9. OWASP FOUNDATION. OWASP TOP TEN. *A03:2021 – Injection* [online]. 2023 [cit. 2023-02-28]. Dostupné z: https://owasp.org/Top10/A03_2021-Injection/.
10. OWASP FOUNDATION. OWASP TOP TEN. *A04:2021 – Insecure Design* [online]. 2023 [cit. 2023-02-28]. Dostupné z: https://owasp.org/Top10/A04_2021-Insecure_Design/.
11. OWASP FOUNDATION. OWASP TOP TEN. *A05:2021 – Security Misconfiguration* [online]. 2023 [cit. 2023-02-28]. Dostupné z: https://owasp.org/Top10/A05_2021-Security_Misconfiguration/.
12. OWASP FOUNDATION. OWASP TOP TEN. *A06:2021 – Vulnerable and Outdated Components* [online]. 2023 [cit. 2023-02-28]. Dostupné z: https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/.
13. OWASP FOUNDATION. OWASP TOP TEN. *A07:2021 – Identification and Authentication Failures* [online]. 2023 [cit. 2023-02-28]. Dostupné z: https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/.

14. OWASP FOUNDATION. OWASP TOP TEN. *A08:2021 – Software and Data Integrity Failures* [online]. 2023 [cit. 2023-02-28]. Dostupné z: https://owasp.org/Top10/A08_2021-Software_and_Data_Integrity_Failures/.
15. OWASP FOUNDATION. OWASP TOP TEN. *A09:2021 – Security Logging and Monitoring Failures* [online]. 2023 [cit. 2023-02-28]. Dostupné z: https://owasp.org/Top10/A09_2021-Security_Logging_and_Monitoring_Failures/.
16. OWASP FOUNDATION. OWASP TOP TEN. *A10:2021 – Server-Side Request Forgery (SSRF)* [online]. 2023 [cit. 2023-02-28]. Dostupné z: https://owasp.org/Top10/A10_2021-Server-Side_Request_Forgery_%28SSRF%29/.
17. OWASP FOUNDATION. *OWASP API TOP TEN* [online]. [cit. 2023-03-02]. Dostupné z: https://owasp.org/API-Security/editions/2023/en/0x11-t10/.
18. OWASP FOUNDATION. OWASP API TOP TEN. *API1:2023 Broken Object Level Authorization* [online]. 2023 [cit. 2023-03-02]. Dostupné z: https://owasp.org/API-Security/editions/2023/en/0xa1-broken-object-level-authorization/.
19. OWASP FOUNDATION. OWASP API TOP TEN. *API2:2023 Broken Authentication* [online]. 2023 [cit. 2023-03-02]. Dostupné z: https://owasp.org/API-Security/editions/2023/en/0xa2-broken-authentication/.
20. OWASP FOUNDATION. OWASP API TOP TEN. *API3:2023 Broken Object Property Level Authorization* [online]. 2023 [cit. 2023-03-02]. Dostupné z: https://owasp.org/API-Security/editions/2023/en/0xa3-broken-object-property-level-authorization/.
21. OWASP FOUNDATION. OWASP API TOP TEN. *API4:2023 Unrestricted Resource Consumption* [online]. 2023 [cit. 2023-03-02]. Dostupné z: https://owasp.org/API-Security/editions/2023/en/0xa4-unrestricted-resource-consumption/.
22. OWASP FOUNDATION. OWASP API TOP TEN. *API5:2023 Broken Function Level Authorization* [online]. 2023 [cit. 2023-03-02]. Dostupné z: https://owasp.org/API-Security/editions/2023/en/0xa5-broken-function-level-authorization/.
23. OWASP FOUNDATION. OWASP API TOP TEN. *API6:2023 Unrestricted Access to Sensitive Business Flows* [online]. 2023 [cit. 2023-03-02]. Dostupné z: https://owasp.org/API-Security/editions/2023/en/0xa6-unrestricted-access-to-sensitive-business-flows/.
24. OWASP FOUNDATION. OWASP API TOP TEN. *API7:2023 Server Side Request Forgery* [online]. 2023 [cit. 2023-03-02]. Dostupné z: https://owasp.org/API-Security/editions/2023/en/0xa7-server-side-request-forgery/.
25. OWASP FOUNDATION. OWASP API TOP TEN. *API8:2023 Security Misconfiguration* [online]. 2023 [cit. 2023-03-02]. Dostupné z: https://owasp.org/API-Security/editions/2023/en/0xa8-security-misconfiguration/.
26. OWASP FOUNDATION. OWASP API TOP TEN. *API9:2023 Improper Inventory Management* [online]. 2023 [cit. 2023-03-02]. Dostupné z: https://owasp.org/API-Security/editions/2023/en/0xa9-improper-inventory-management/.
27. OWASP FOUNDATION. OWASP API TOP TEN. *API10:2023 Unsafe Consumption of APIs* [online]. 2023 [cit. 2023-03-02]. Dostupné z: https://owasp.org/API-Security/editions/2023/en/0xaa-unsafe-consumption-of-apis/.
28. OWASP FOUNDATION. *OWASP WSTG* [online]. [cit. 2023-03-03]. Dostupné z: https://owasp.org/www-project-web-security-testing-guide/stable/.
29. FIRST. *Common Vulnerability Scoring System* [online]. [cit. 2023-03-04]. Dostupné z: https://www.first.org/cvss/.

30. FIRST. Common Vulnerability Scoring System. *Common Vulnerability Scoring System v3.1: Specification Document* [online]. 2023 [cit. 2023-03-04]. Dostupné z: https://www.first.org/cvss/v3.1/specification-document.

31. FIRST. *Common Vulnerability Scoring System Version 3.1 Calculator* [online]. [cit. 2023-03-04]. Dostupné z: https://www.first.org/cvss/calculator/3.1.

32. FIRST. *Common Vulnerability Scoring System v3.1: Specification Document* [online]. [cit. 2023-03-03]. Dostupné z: https://www.first.org/cvss/v3-1/media/MetricGroups.svg.

33. PTES. *Penetration testing execution standard* [online]. [cit. 2023-03-05]. Dostupné z: http://www.pentest-standard.org/index.php/Main_Page.

34. PTES. Penetration testing execution standard. *Pre-engagement* [online]. 2023 [cit. 2023-03-05]. Dostupné z: http://www.pentest-standard.org/index.php/Pre-engagement.

35. PTES. Penetration testing execution standard. *Intelligence Gathering* [online]. 2023 [cit. 2023-03-05]. Dostupné z: http://www.pentest-standard.org/index.php/Intelligence_Gathering.

36. PTES. Penetration testing execution standard. *Threat Modeling* [online]. 2023 [cit. 2023-03-05]. Dostupné z: http://www.pentest-standard.org/index.php/Threat_Modeling.

37. PTES. Penetration testing execution standard. *Vulnerability Analysis* [online]. 2023 [cit. 2023-03-05]. Dostupné z: http://www.pentest-standard.org/index.php/Vulnerability_Analysis.

38. PTES. Penetration testing execution standard. *Exploitation* [online]. 2023 [cit. 2023-03-05]. Dostupné z: http://www.pentest-standard.org/index.php/Exploitation.

39. PTES. Penetration testing execution standard. *Post Exploitation* [online]. 2023 [cit. 2023-03-05]. Dostupné z: http://www.pentest-standard.org/index.php/Post_Exploitation.

40. PTES. Penetration testing execution standard. *Reporting* [online]. 2023 [cit. 2023-03-05]. Dostupné z: http://www.pentest-standard.org/index.php/Reporting.

41. LYON, Gordon. *Nmap security scanner. Version 7.93* [soft.]. 1997. [cit. 2023-03-06]. Dostupné z: https://nmap.org.

42. PABLO. *OWASP ZAP Tutorial - Part 1: Intercepting Traffic* [online]. [cit. 2023-03-03]. Dostupné z: https://3.bp.blogspot.com/-J4Lt29XbFJ8/XBHeDAepqFI/AAAAAAAAogg/KAz8l6NhdQMVQExegkZIEVfQ7CBIKDibACLcBGAs/s400/Screen%2BShot%2B2018-12-12%2Bat%2B10.19.16%2BPM.png.

43. OWASP. *Zed Attack Proxy. Version 2.12.0* [soft.]. 2014. [cit. 2023-03-06]. Dostupné z: https://www.zaproxy.org.

44. OWASP FOUNDATION. OWASP ZAP. *Passive scan* [online]. 2023 [cit. 2023-03-06]. Dostupné z: https://www.zaproxy.org/docs/desktop/start/features/pscan/.

45. OWASP FOUNDATION. OWASP ZAP. *Active scan* [online]. 2023 [cit. 2023-03-06]. Dostupné z: https://www.zaproxy.org/docs/desktop/start/features/ascan/.

46. OWASP FOUNDATION. OWASP ZAP. *Fuzzing* [online]. 2023 [cit. 2023-03-06]. Dostupné z: https://www.zaproxy.org/docs/desktop/addons/fuzzer/.

47. OWASP FOUNDATION. OWASP ZAP. *Spider* [online]. 2023 [cit. 2023-03-06]. Dostupné z: https://www.zaproxy.org/docs/desktop/addons/spider/.

48. OWASP FOUNDATION. OWASP ZAP. *Manipulator-in-the-middle Proxy* [online]. 2023 [cit. 2023-03-06]. Dostupné z: https://www.zaproxy.org/docs/desktop/start/features/intercept/.

49. RAPID7 LLC. *Metasploit Framework. Version 4.22.0* [soft.]. 2013. [cit. 2023-03-07]. Dostupné z: https://www.metasploit.com.

50. RAPID7 LLC. Metasploit framework. *Metasploit modules* [online]. 2023 [cit. 2023-03-06]. Dostupné z: https://docs.metasploit.com/docs/modules.html.
51. SULLO, Chris. *Nikto. Version 2.5* [soft.]. 2012. [cit. 2023-03-08]. Dostupné z: https://cirt.net/Nikto2.
52. PROUZA, Petr. *Transformace systému z monolitické architektury do architektury mikroslužeb*. Diplomová práce.Praha:České vysoké učení technické v Praze,Fakulta informačních technologií, 2021.
53. MDN WEB DOCS. *X-XSS-Protection* [online]. [cit. 2023-03-08]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-XSS-Protection.
54. MDN WEB DOCS. *X-Content-Type-Options* [online]. [cit. 2023-03-08]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Content-Type-Options.
55. MDN WEB DOCS. *X-Frame-Options* [online]. [cit. 2023-03-08]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/X-Frame-Options.
56. MDN WEB DOCS. *Strict-Transport-Security* [online]. [cit. 2023-03-08]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Strict-Transport-Security.
57. MDN WEB DOCS. *Expect-CT* [online]. [cit. 2023-03-08]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Expect-CT.
58. RUDOLPH, Hans Christian. *Ciphersuite* [online]. 2017. [cit. 2024-02-10]. Dostupné z: https://ciphersuite.info/cs/.
59. SNYK. *jquery@1.12.1 vulnerabilities* [online]. [cit. 2023-03-08]. Dostupné z: https://security.snyk.io/package/npm/jquery/1.12.1.

Příloha A
Seznam použitých zkratek

API Application Programming Interface

CVSS Common Vulnerability Scoring System

ČVUT České vysoké technické učení v Praze

ČZU Česká zemědělská univerzita v Praze

DoS Denial of Service

DDoS Distributed Denial of Service

FIRST Forum of Incident Response and Security Teams

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

IAM Identity and Access Management

IDS Intrusion Detection System

IPS Intrusion Prevention System

IT Information Technology

JWT Json Web Token

LDAP Lightweight Directory Access Protocol

NSE Nmap Scripting Engine

OS Operating System

OSINT Open Source Intelligence

OWASP Open Web Application Security Project

PIN Personal Identification Number

PTES Penetration Testing Execution Standard
SQL Structured Query Language

SSL Secure Sockets Layer

TLS Transport Layer Security

URL Uniform Resource Locator

USB Universal Serial Bus

VOP Všeobecné obchodní podmínky

VŠE Vysoká škola ekonomická v Praze

WSTG Web Security Testing Guide

XSS Cross-site scripting

ZAP Zed Attack Proxy

Obsah přiloženého archivu

```
readme.txt ..... stručný popis obsahu média
src
    thesis ..... zdrojová forma práce ve formátu LaTeX
text ..... text práce
    thesis.pdf ..... text práce ve formátu PDF
```