FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

Zadání diplomové práce

Název:	Analýza bezpečnosti webové aplikace Seznamovák
Student:	Bc. Jaroslav Kříž
Vedoucí:	Ing. Tomáš Nováček
Studijní program:	Informatika
Obor / specializace:	Počítačová bezpečnost
Katedra:	Katedra informační bezpečnosti
Platnost zadání:	do konce letního semestru 2022/2023

Pokyny pro vypracování

Webová aplikace Seznamovák již deset let slouží jako hlavní technologický prvek pro organizaci úvodního soustředění FIT ČVUT. Protože byla psána studenty pro studenty, je velmi pravděpodobné, že obsahuje bezpečnostní rizika, která mohou komukoliv umožnit systém poškodit či získat informace o účastnících.

1) Popište největší bezpečnostní hrozby webových aplikací.
2) Seznamte se s aplikací Seznamovák a popište ji, soustřeďte se na její zabezpečení. Zahrňte již známé problémy celé aplikace.
3) Analyzujte webovou aplikaci Seznamovák z pohledu její bezpečnosti. Zaměřte se převážně na bezpečnost hesel, osobních údajů a poskytovaného API.
4) Využijte vámi zvolený nástroj určený pro penetrační testování ke zjištění slabých míst aplikace.
5) Diskutujte nalezená zjištění a pokuste se navrhnout opravy, které vyřeší nalezené zranitelnosti.

Elektronicky schválil/a prof. Ing. Róbert Lórencz, CSc. dne 29. listopadu 2021 v Praze.

FAKULTA
INFORMAČNÍCH
TECHNOLOGIÍ
ČVUT V PRAZE

Diplomová práce

Analýza bezpečnosti webové aplikace Seznamovák

Bc. Jaroslav Kříž

Katedra informační bezpečnosti
Vedoucí práce: Ing. Tomáš Nováček

28. dubna 2022

Poděkování

V první řadě bych chtěl poděkovat vedoucímu práce Ing. Tomáši Nováčkovi za odborné a cenné rady, vedení a pomoc při tvorbě této práce. Dále bych chtěl poděkovat všem, kteří svými radami přispěli ke zkvalitnění mé práci. V neposlední řadě bych chtěl poděkovat své rodině a přátelům za podporu 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 principu 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 předpisu. V souladu s ust. § 2373 odst. 2 zákona č. 89/2012 Sb., občanský zákoník, ve znění pozdějších předpisu, tímto uděluji nevýhradní oprávnění (licenci) k užití této mojí práce, a to včetně všech počítačových programu, jež jsou její součástí či přílohou a veškeré jejich dokumentace (dále souhrnně jen „Dílo"), a to všem osobám, které si přejí Dílo užít. Tyto osoby jsou oprávněny Dílo užít jakýmkoli zpusobem, který nesnižuje hodnotu Díla a za jakýmkoli účelem (včetně užití k výdělečným účelum). Toto oprávnění je časově, teritoriálně i množstevně neomezené. Každá osoba, která využije výše uvedenou licenci, se však zavazuje udělit ke každému dílu, které vznikne (byť jen zčásti) na základě Díla, úpravou Díla, spojením Díla s jiným dílem, zařazením Díla do díla souborného či zpracováním Díla (včetně překladu) licenci alespoň ve výše uvedeném rozsahu a zároveň zpřístupnit zdrojový kód takového díla alespoň srovnatelným zpusobem a ve srovnatelném rozsahu, jako je zpřístupněn zdrojový kód Díla.

V Praze dne 28. dubna 2022 . . . . . .

Odkaz na tuto práci

Kříž, Jaroslav. *Analýza bezpečnosti webové aplikace Seznamovák*. Diplomová práce. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2022.

Abstrakt

Tato diplomová práce se zabývá analýzou bezpečnosti webové aplikace Seznamovák. Během analýzy bezpečnosti bylo objeveno několik bezpečnostních zranitelností, které mohou oslabit použitou kryptografii nebo ohrozit bezpečnost hesel a osobních informací účastníků. Pokud by byl útočníkem účastník Seznamováku, mohly by mu také jednotlivé informace zkazit zážitek z akce.

Klíčová slova bezpečnostní analýza, OWASP, Top 10 2021, API Security Top 10 2019, bezpečnostní zranitelnost, penetrační test, aplikace Seznamovák

Abstract

This diploma thesis addresses the security analysis of the web application Seznamovák. During the investigation, several security vulnerabilities were discovered that might weaken used cryptography or jeopardize the security of passwords. Or even leak personal information of participants, which is necessary for organizers of the Seznamovák. If the attacker participated in the event, the individual information could also ruin his experience.

Keywords security analysis, OWASP, Top 10 2021, API Security Top 10 2019, security vulnerabilities, penetration test, application Seznamovák

Obsah

Seznam obrázků

Seznam tabulek

Seznam výpisů kódu

Úvod

Tato diplomová práce se zabývá analýzou webové aplikace Seznamovák. Jedná se o aplikaci, kterou vytvořili organizátoři úvodního seznamovacího kurzu (tzv. Seznamováku FIT) pro zjednodušení organizace celé akce, která každoročně čítá zpravidla 180 účastníků a 20 organizátorů.

Organizátoři jsou zároveň studenty na Fakultě informačních technologií ČVUT v Praze. Jelikož se jedná o technologické nadšence, v relativně krátkém období vytvořili i několik dalších aplikací, které jsou provázané s původní webovou aplikací a usnadňují proces organizace.

Webová aplikace Seznamovák vznikala v průběhu několika let a podstoupila řadu oprav a rozšíření, které byly provedeny různými vývojáři s minimem koordinace. Tento způsob vývoje mohl zásadně narušit robustnost celé aplikace a její odolnost vůči bezpečnostním rizikům.

Protože aplikaci vyvíjeli studenti pro studenty, kteří mají ještě limitované znalosti bezpečnostních rizik, je velmi pravděpodobné, že do aplikace neúmyslně zanesli několik bezpečnostních zranitelností, které mohou být zneužity útočníky. Tento předpoklad se v průběhu let ukázal jako správný, opakovaně totiž došlo k narušení integrity aplikace.

První kapitola této diplomové práce tudíž popisuje zranitelnosti webových aplikací a představuje organizaci OWASP, která se zabývá zvyšováním povědomí o těchto bezpečnostních hrozbách. Tato organizace vytvořila projekt OWASP Top 10, který upozorňuje na nejčastější zranitelnosti webových aplikací. Mezi její další projekty patří API Security Top 10, který popisuje nejčastější bezpečnostní zranitelnosti API. Seznamy jsou vždy v horizontu několika let aktualizovány a doplňovány o nové a aktuální zranitelnosti.

Druhá kapitola seznamuje čtenáře s webovou aplikací Seznamovák, důvodem jejího vzniku a představuje technologie, které aplikaci tvoří. V této kapitole jsou popsány i aplikace, zejména pro mobilní telefony, pro které webová aplikace představuje nedílnou součást a pro něž slouží jako backend.

Třetí kapitola se zabývá analýzou samotné aplikace Seznamovák a způsobem, jakým je nakládáno s uživatelskými daty účastníků a organizátorů. Je zde zároveň provedena statická analýza kódu jednotlivých komponent a navržena oprava zranitelných míst a konkrétních částí, které neodpovídají současným nárokům na bezpečnost.

Čtvrtá kapitola se zaměřuje na penetrační testování webové aplikace Seznamovák. Jsou zde zároveň představeny některé běžně používané nástroje pro penetrační testování, které byly v průběhu práce použity.

Pátá a zároveň poslední kapitola obsahuje scénář, při němž se podaří útočníkovi získat hashe (otisky) hesel z databáze a poté se na ně pokusí zaútočit.

Kapitola 1

Zranitelnosti webových aplikací

Odhaduje se, že internet, potažmo webové stránky, používá asi 4,5 miliardy lidí, kteří souhrnně navštěvují zhruba 1,7 miliardy webových stránek [1].

Jelikož se jedná opravdu o velké číslo, stává se otázka bezpečnosti velmi aktuální. Bohužel se ale ukazuje, že tato představa je spíše utopická, jednotlivé webové aplikace obsahují více či méně závažné chyby, které mohou vést k napadení těchto aplikací a následně ke zneužití dat, která jsou v nich obsažena.

Vznik bezpečnostních chyb ve webových aplikacích má mnoho příčin, může se jednat o nedostatečné znalosti na straně programátora webových stránek, na straně autora použitých knihoven a frameworků nebo nedostatečným či nekvalitním otestováním celku (nebo dílčích částí) aplikace. Popřípadě může být vznik chyb zapříčiněn vysokým tlakem na rychlost vývoje aplikace, a proto bývá zabezpečení aplikací vynecháno, nebo potlačeno na absolutní minimum.

Zranitelností existuje takřka nekonečný výčet, nicméně některé zranitelnosti jsou častější, snáze proveditelné, potažmo mnohem nebezpečnější než jiné. Proto existují organizace, které se snaží o osvětu na poli zabezpečení webových aplikací. Jednou z nich je i OWASP.

1.1 Open Web Application Security Project

Open Web Application Security Project (OWASP) je mezinárodní nezisková organizace, která se zaměřuje na zvyšování bezpečnosti zejména webových aplikací. Za jejím vznikem stojí Mark Curphey a Dennis Groves, kteří založili OWASP Foundation v roce 2001 a následně ji v roce 2004 transformovali do neziskové organizace.

Za dobu své existence zaštítila organizace OWASP značné množství projektů, které mají za cíl zlepšit povědomí o bezpečnosti a celkovém zabezpečení webových aplikací. I proto na webových stránkách organizace [2] nalezne čtenář nepřeberné množství materiálů, mezi které patří například články, výuková videa, aplikace demonstrující zranitelnosti a útoky na ně.

Mezi projekty pod hlavičkou OWASP patří například:

- **OWASP Top 10** je žebříčkem 10 nejzávažnějších bezpečnostních hrozeb pro webové aplikace.
- **API**[1] **Security Top 10** je žebříčkem 10 nejzávažnějších bezpečnostních hrozeb pro API.
- **OWASP ASVS**[2] je norma určující standard pro testování bezpečnostních prvků aplikací.
- **WebScarab** je aplikací pro testování zranitelností webových aplikací.
- **The Guide** je směrnice obsahující pokyny pro dosažení správného zabezpečení webových aplikací během vývoje.

1.2 OWASP Top 10

Pravděpodobně nejznámější projekt této organizace je OWASP Top 10, tedy seznam deseti nejkritičtějších zranitelností webových aplikací. Jelikož se zranitelnosti v průběhu let vyvíjí, dochází každých pár let, tradičně se jedná o tři nebo čtyři roky, k vydávání nového, aktualizovaného seznamu.

Vzhledem ke struktuře a samotné podstatě tohoto seznamu je nutné si uvědomit, že se jedná o zranitelnosti, které jsou buď velmi rozšířené nebo mají potenciálně velký dopad na bezpečnost celé aplikace. Určitě ale nelze očekávat, že po odstranění těchto zranitelností už není aplikace napadnutelná. Pořád existuje nepřeberné množství dalších technik a způsobů, jak na aplikaci zaútočit.

Zároveň může tento seznam vypadat jako dobrý odrazový můstek, kde lze začít při provádění bezpečnostní analýzy aplikace. Od tohoto postupu ale autoři žebříčku odrazují, jelikož tento seznam má spíše informativní charakter (a může mít poměrně omezený rozsah). Některé položky zároveň nemusí být vůbec snadno testovatelné. Například statickou analýzou kódu lze rozhodnout, jestli dochází k logování a monitoringu dat, ale je nemožné určit, jestli je toto logování efektivní nebo dostatečné. [3]

Pro potřeby testování (a ideálně zařazení do celého procesu vývoje) je doporučené porozhlédnout se po vhodnějších alternativách. Jednou z těchto alternativ je i výše zmíněný OWASP Application Security Verification Standard, který vznikl pod hlavičkou organizace OWASP a který se na testování zaměřuje.

[1] Application Programming Interface
[2] Application Security Verification Standard

Mezi jednotlivé položky OWASP Top 10 pro rok 2021 patří dle [4]:

1. **Broken Access Control**, kdy útočník může využít chyb v kontrole přístupu, aby získal přístup k citlivým datům nebo určitým funkcím systému.

2. **Cryptographic Failures**, kdy není zabezpečen přenos dat nebo citlivých informací. Data je poté možné během přenosu změnit, popřípadě využít k dalším útokům.

3. **Injection** neboli zranitelnost vsunutím škodlivého kódu. Zranitelnost vede k úniku a ztrátě dat nebo spuštění dalšího škodlivého kódu.

4. **Insecure Design**, kdy samotný návrh systému neobsahuje bezpečnostní prvky vůbec, nebo jsou tyto prvky považované za zastaralé či nevhodné. Do této kategorie patří i nedostatečně otestovaný produkční kód.

5. **Security Misconfiguration**, při kterém dochází k používání výchozích, neúplných nebo špatných konfigurací systémů. Kvůli tomu pak může docházet k detailnímu výpisu chyb uživatelům, případně může špatná konfigurace výrazně usnadnit kompromitaci systému nebo služeb.

6. **Vulnerable and Outdated Components**, při kterém jsou využívány zranitelnosti v komponentách a frameworcích třetích stran. Tyto komponenty mohou být neaktualizované a mohou obsahovat známé zranitelnosti.

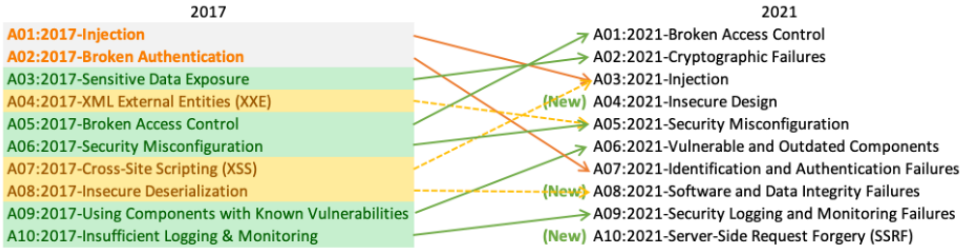
7. **Identification and Authentication Failures**, kdy je nedostatečným nebo chybným způsobem implementovaná autentizace. Zranitelnost tak může vést ke kompromitaci uživatelských účtů nebo celého systému.

8. **Software and Data Integrity Failures**, při kterém dochází ke stahování a spouštění kódu a aplikací, aniž by byla ověřena jejich správnost a integrita. Do této kategorie patří i nezabezpečená serializace dat.

9. **Security Logging and Monitoring Failures**, kdy dochází k nedostatečnému logování a monitorování. Časté jsou chybějící automatické notifikace, což vede k pomalé, nebo chybějící odpovědi na útoky.

10. **Server-Side Request Forgery**, při kterém dochází k modifikaci URL, což může vést ke stahování dat, ke kterým by útočník neměl mít přístup.

Žebříček byl sestavený na základě dat z více než půl milionu aplikací, což je zatím největší dataset, který se podařilo získat. Data do tohoto seznamu byla poskytnuta několika desítkami společností (od některých z nich anonymně), například se jednalo o společnosti AppSec Labs, WhiteHat a GitLab. [5]



Obrázek 1.1: Změny oproti OWASP Top 10 2017 [6]

Prvních osm položek na seznamu odpovídá zranitelnostem, které se podařilo nasbírat z reálných dat od mnoha různých společností a aplikací. Poslední dvě místa v žebříčku jsou obsazena na základě výsledků ankety, které se účastní odborná veřejnost. Seznam je proto směsicí mezi dnes poměrně běžnými zranitelnostmi, a položkami, které sice nemusí být v současné době tak rozšířené, ale mají potenciál být velkou hrozbou v budoucnosti.

Samotní autoři přiznávají, že v žebříčku občas dochází ke stírání rozdílů mezi *příčinou* a *následkem*. Tedy že například kryptografická selhání (bod 2) mohou vést k odhalení citlivých informací (obsaženo v bodu 1), nebo třeba špatně nastavená konfigurace (bod 5) povede k zahlcení a nedostupnosti služby (obsaženo v bodu 10). Pro vyřešení tohoto problému je proto lepší snažit se vyřešit příčinu, což ve svém důsledku povede k vyřešení symptomů.

V průběhu let se postupně mění závažnosti jednotlivých hrozeb, nebo dochází k jejich slučování, popřípadě k objevování nových zranitelností. Oproti předchozí verzi seznamu (OWASP Top 10 2017) došlo ke sloučení tří existujících kategorií a zároveň došlo k přejmenování a dílčím změnám v dalších čtyřech kategoriích. Obrázek 1.1 zobrazuje všechny tyto změny graficky a poskytuje tak pohled na dynamiku zranitelností na webu.

1.2.1 Broken Access Control

Broken access control, neboli nefunkční kontrola přístupu a přístupových práv. Přístupová práva rozhodují o tom, jestli daný uživatel může provést tu či onu akci.

Špatně nastavená práva tedy mohou vést k tomu, že uživatel může získat přístup k datům, ke kterým by neměl mít přístup, popřípadě data může modifikovat nebo i smazat.

Podle dat vedoucích k sestavení žebříčku OWASP Top 10, a datasetu testovaného na tuto zranitelnost, který čítá přes 318 tisíc aplikací, bylo více než 94 % z těchto aplikací náchylných k nějaké formě narušení přístupových práv. Není proto divu, že se oproti předchozímu Top 10 posunula nefunkční kontrola přístupových práv z páté pozice na první příčku.

Do této kategorie je zahrnuto několik velmi častých prohřešků proti bezpečnosti, mezi něž patří například:

- Porušení tzv. *principu nejnižších oprávnění*, tedy implicitního zamítnutí všech přístupů a jejich povolování na jednotlivé úkony, které by měly být povoleny uživatelům, popřípadě skupinám uživatelů, kteří na ně mají nárok při provádění jednotlivých operací.

- Přemostění kontroly přístupových práv na základě modifikace URL, vnitřního stavu aplikace nebo modifikace API požadavků.

- Pokud se útočníkovi podaří získat jedinečný identifikátor (token) skutečného uživatele, může si zobrazit všechny informace, ke kterým má napadený uživatel přístup. V krajním případě poté může z role administrátora provádět úpravy mnohem většího a závažnějšího charakteru.

- Eskalace privilegií, popřípadě přístup k datům i jakožto nepřihlášený uživatel, nebo vystupovat jako administrátor, zatímco je přihlášený jako běžný uživatel. [7]

1.2.2 Cryptographic Failures

Cryptographic Failures, neboli pochybení při použití kryptografických prvků. Takřka každá aplikace potřebuje uchovávat data a nemalé procento z nich obsahuje i citlivé údaje, mezi které patří například osobní údaje, zdravotní záznamy, čísla kreditních karet a mnoho dalších.

Druhý bod z OWASP Top 10 žebříčku popisuje případy, kdy těmto informacím není věnována speciální péče, například v podobě šifrování při přenosu dat, při jejich perzistentním uložení a nebo pouhém zobrazení.

Podle dat z žebříčku jsou pochybení z této kategorie přítomny u téměř 80 % z 233 tisíc testovaných aplikací, a proto se posunují na druhou příčku v současném seznamu.

Pro tuto kategorii jsou velmi časté prohřešky, při nichž:

- Jsou data přenášena ve formě tzv. *plaintextu*, tedy bez jakéhokoliv šifrování.

- Jsou používány zastaralé nebo slabé kryptografické algoritmy, popřípadě protokoly. Těch může být velká řada, nejčastěji se ale jedná o (částečně nebo úplně) prolomené hashovací funkce MD5 a SHA1 nebo zastaralý síťový protokol TLS[3] ve verzi 1.0 a 1.1.

- Dochází ke generování slabých, případně již jednou použitých klíčů.

- Dochází ke špatnému ověření serverových certifikátů, popřípadě celého řetězce certifikátů. [8]

[3]Transport Layer Security

1.2.3 Injection

Injection, neboli zranitelnost vsunutím škodlivého kódu. Aplikace většinou pracují se vstupem, který je závislý na tom, co zadává uživatel. Typickým reprezentantem je SQL[4] injection a tvorba SQL dotazů na základě uživatele (vyhledávání, filtry, a jiné). V případě, že tento vstup není nijak ošetřen a je následně použit pro dotazy do databáze, může se stát, že se podaří útočníkovi zobrazit data, ke kterým by normálně neměl mít přístup. V jiném, ale stále stejně závažném, případě může tyto informace modifikovat nebo úplně smazat. Alternativně, např. PHP[5] injection, může spustit jiný (a pravděpodobně škodlivý) kód.

Zranitelnost vsunutím škodlivého kódu byla, od samotného počátku sestavování, na přední příčce Top 10. Dřívější zranitelnost číslo jedna se ale po více než deseti letech přesunula na třetí pozici. Ačkoliv se předchozí dva body v seznamu ukázaly jako mnohem závažnější, zranitelnost vsunutím škodlivého kódu je v nějaké formě stále přítomna u více než 94 % testovaných aplikací.

Do této kategorie patří několik všeobecně známých a závažných útoků, mezi které patří například SQL Injection, LDAP[6] Injection nebo zranitelnost typu Cross-site scripting (XSS).

Aplikace jsou proti těmto útokům zranitelné tehdy, když data zadávaná uživateli nejsou žádným způsobem ověřena nebo filtrována a jsou následně, bez správného escapování, použita v dynamicky generovaných dotazech. [9]

1.2.4 Insecure Design

Insecure Design, neboli návrh aplikace, který nevyhovuje bezpečnostním standardům. Jedná se o nově vytvořenou kategorii, která má poměrně široký záběr a potenciálně velké dopady na bezpečnost celé aplikace.

Pro plné pochopení zranitelností v této kategorii je třeba uvědomit si rozdíl mezi špatným návrhem a špatnou implementací návrhu. Zatímco při nalezení chyby v implementaci je relativně snadné a přímočaré ji opravit, při nalezení chyby v návrhu aplikace nedokáže sebelepší implementace bezpečnostních prvků zlepšit zabezpečení celé aplikace. Vývojáři aplikací by proto měli věnovat čas správnému a bezpečnému návrhu aplikace ještě před začátkem vývoje.

Mezi chyby, které se často objevují tak může patřit například:

- Nedostatečné nebo zcela chybějící modulární a integrační testování produkčního kódu.
- Generování chybových zpráv s příliš detailními informacemi, které obsahují citlivé informace.

[4] Structured Query Language
[5] PHP: Hypertext Preprocessor, původní zkratka vznikla ale z „Personal Home Page“.
[6] Lightweight Directory Access Protocol

- Přidávání bezpečnostních prvků na poslední chvíli, v nejhorším případě až po dopsání hlavních komponent aplikace. [10]

1.2.5 Security Misconfiguration

Chybně nastavená konfigurace je dalším z velmi častých problémů. Z dat použitých pro žebříček vyplývá, že s chybně nastavenou konfigurací se dnes potýká takřka každá aplikace, a zranitelnost se tak posouvá v současném žebříčku o jednu pozici výše, než byla v předchozí inkarnaci OWASP Top 10 žebříčku.

Doporučením v této kategorii je i existence vývojových a testovacích prostředí, která jsou z pohledu konfigurace identická, ale obsahují jiné přihlašovací údaje pro každé z těchto prostředí.

Mezi typické zranitelnosti, které spadají do této kategorie, patří například:

- Povolení služeb, které nejsou aplikací nijak využívány. Může se jednat o samotné služby, porty, účty a mnoho dalších.
- Používání výchozích konfigurací, které nejsou v základu nijak zabezpečené. Popřípadě ponechání výchozích účtů a hesel v produkčních verzích aplikací, které mají mnohdy administrátorská oprávnění.
- Špatně nakonfigurované HTTP hlavičky, které odhalují informace důležité pro útočníka, nebo naopak neodesílají bezpečnostní hlavičky. Útočník tak může svůj útok lépe zacílit a vynaložit na něj menší úsilí.
- Dochází k detailnímu výpisu chybových zpráv, které na straně klienta obsahují citlivé informace. [11]

1.2.6 Vulnerable and Outdated Components

Zranitelné a neaktualizované komponenty jsou problémem, kterým trpí téměř 52 % aplikací z datasetu. Kromě toho se tato zranitelnost stala i číslem dva ve výsledcích ankety určené pro odbornou veřejnost.

Do této kategorie ale patří i spouštění komponent – knihoven, frameworků a modulů – se stejnými právy jako aplikace, ze které byly tyto komponenty volány. Pokud navíc obsahuje tato komponenta dříve objevenou zranitelnost (nebo k ní existuje i *exploit*[7]), vede použití takovéto komponenty k výraznému ulehčení úspěšného útoku.

Aplikace je pravděpodobně zranitelná, pokud:

- Nedochází k hledání nových zranitelností v pravidelných intervalech, popřípadě k odběru novinek jednotlivých komponent, které jsou v aplikacích využívány.

[7] Jedná se o kód, který využívá chybu, která vede k původně nezamýšlené činnosti a umožňuje útočníkovi získat nějaký prospěch.

- Nejsou známé přesné verze všech komponent, které jsou použité v aplikaci. Může se tak používat zranitelný, nepodporovaný nebo zastaralý software.
- Dochází k aktualizaci komponent, aniž by bylo předem zjištěno, zda jsou tyto aktualizace kompatibilní se současným nastavením systému. [12]

1.2.7 Identification and Authentication Failures

Druhým sestupem z předních příček seznamu, tentokrát o poznání větším, je přesun z druhého místa na místo sedmé, pro zranitelnost, která se týká pochybení u identifikace a autentizace uživatele.

V této zranitelnosti se útočník zaměřuje na chyby v procesu autentizace a správě *session* (relace). Může se mu tak podařit získat identitu skutečných uživatelů, za které se může vydávat buď dočasně, nebo trvale. Pro útočníka je ideální získat tímto způsobem účet s vyššími právy, například účet administrátora. Díky tomu pak může způsobit narušení aplikace.

Ačkoliv aplikace nemusí mít (a mnohdy také nemají) správně implementovaný proces autentizace nebo umožňují používat svým uživatelům slabá hesla, už se více dbá na vícefaktorovou autentizaci (MFA) a proces změny hesla je zabezpečenější. V dalších letech by proto snad už tato zranitelnost nemusela být mezi deseti nejčastějšími. Přesto jsou velmi častá pochybení, kdy:

- V aplikaci je možné provádět automatické útoky, mezi které patří třeba útok hrubou silou.
- Používají se slabé nebo neefektivní způsoby obnovy zapomenutých hesel.
- Dochází ke špatnému (nebo vůbec žádnému) způsobu zneplatňování Session ID, například při odhlašování uživatelů, přihlášení uživatelů na více zařízeních současně nebo po vypršení času, po kterém je uživatelova relace validní. [13]

1.2.8 Software and Data Integrity Failures

Chyby při ověřování integrity dat jsou další kategorií, která je v Top 10 nová. Při této chybě nedochází k ověřování, jestli jsou pluginy, moduly a jiná rozšíření aplikace staženy korektně a jestli nedošlo během přenosu k jejich narušení.

Toto se týká hlavně aktualizací, které mohou být staženy z nedůvěryhodných zdrojů, popřípadě aplikací, které samy poskytují možnost automatického stažení aktualizace. V případě kompromitace tohoto updatu dochází poté ke spuštění škodlivého kódu v rámci stejných práv, jaká měla aplikace.

Součástí této kategorie je i nebezpečná deserializace[8], která byla do tohoto bodu sloučena. Deserializace se vyskytuje tehdy, kdy jsou uživatelem

[8]Proces, při kterém jsou data v podobě řetězce převedena na nějakou formu interní reprezentace.

kontrolovaná data deserializována webovou aplikací. Potenciálně to poté dovoluje útočníkovi, aby manipuloval se serializovanými daty a do aplikace přidal škodlivý kód. [14]

1.2.9 Security Logging and Monitoring Failures

Nesprávné logování a monitoring dat jsou problém, který byl přítomný už v předchozím žebříčku z roku 2017. Oproti předchozímu seznamu si tato zranitelnost ale pohoršila a posouvá se o jednu pozici na žebříčku výše. Zároveň se ve výsledcích hlasování odborné veřejnosti dostalo nesprávné monitorování a logování na třetí pozici.

Nedostatečné logování umožňuje útočníkům, aby prováděli svou činnost delší dobu, aniž by vůbec byli detekování. Dle dat plynoucích pro sestavení předchozí verze Top 10 je průměrná doba, než dojde ke zjištění, že byl systém narušen, 191 dní [15]. Efektivně tak vede vytváření záznamů, které poté nikdo nekontroluje ke stejné chybě jako nevytváření záznamů vůbec.

Součástí správného logování by mělo být i vhodné kódování uložených dat takovým způsobem, aby při bezpečnostním auditu těchto logů nedocházelo ke vzdálenému spuštění škodlivého kódu. K dalším pochybením dochází, když:

- Nejsou logovány důležité události, jako je například přihlášení (ať už úspěšné, nebo i neúspěšné) a provádění drahých transakcí.
- Dochází ke generování záznamů, které jsou nedostatečně podrobné, nesrozumitelné a nebo neumožňují zjistit, k čemu vlastně došlo.
- Penetrační testování během celého procesu nespouštějí vůbec žádná varování, nedochází tedy k detekci podezřelé aktivity. [16]

1.2.10 Server-Side Request Forgery

Tato položka byla do Top 10 přidána na základě hlasování odborné komunity, ve kterém obsadila první příčku. Ačkoliv se zranitelnost objevuje u poměrně velkého procenta testovaných aplikací (67 %), v absolutním počtu se jedná jen o necelých 10 tisíc záznamů. Je to zejména z důvodu malého datasetu, na kterém byla tato zranitelnost testována.

K této zranitelnosti dochází tehdy, když webová aplikace načítá data ze vzdáleného zdroje, aniž by prováděla validaci URL, kterou zadal uživatel.

Útočník tak může přinutit aplikaci, aby odeslala požadavek na adresu, se kterou nebylo při návrhu počítáno. Může se jednat o přesměrování na úplně jinou webovou stránku nebo například adresu v rámci své vnitřní sítě. To může v konečném důsledku vést k odhalení citlivých dat, ke kterým by normálně uživatel neměl přístup. K datům by totiž nepřistupoval z vnější sítě, ale ze sítě vnitřní, která má být zabezpečena. [17]

1.3 OWASP API Security Top 10

API je dnes součástí mnohých webů a umožňuje (ze své podstaty) přístup k citlivým datům, která jsou uložená v databázi webů a aplikací obecně. Z tohoto důvodu se často stávají terčem útoků. Proto je dalším z projektů organizace OWASP i API Security Top 10. Jedná se o seznam deseti nejzávažnějších zranitelností a bezpečnostních rizik se zaměřením na API. Nejnovější verzí tohoto seznamu je podoba z roku 2019.

Mezi jednotlivé položky OWASP API Security Top 10 pro rok 2019 patří dle [18]:

1. **Broken Object Level Authorization**, kdy nedochází k autorizaci uživatelských vstupů. Útočník tak může získat přistup k datům jiných uživatelů a eventuálně je i modifikovat.

2. **Broken User Authentication**, kdy je nedostatečným způsobem implementována autentizace.

3. **Excessive Data Exposure**, při které jsou uživatelům zpřístupněny objekty (včetně jejich vlastností), které mohou obsahovat citlivá data.

4. **Lack of Resources & Rate Limiting**, při které nejsou nijak omezovány parametry a množství odeslaných dotazů. Útočník tak může způsobit nedostupnost serveru nebo API.

5. **Broken Function Level Authorization**, kdy může docházet k chybám v autorizaci z důvodu nejasných rozdílů mezi právy obyčejného uživatele a uživatele s vyššími oprávněními, například administrátora.

6. **Mass Assignment**, při které nedochází k omezení toho, co může daný uživatel modifikovat v rámci svého dotazu.

7. **Security Misconfiguration**, kdy dochází ke špatnému nastavení systému, což v konečném důsledku může vést k odhalení informací o systému nebo jeho naprosté kompromitaci.

8. **Injection**, při které dochází k vložení kusu kódu, který mění zamýšlenou logiku prováděných dotazů. Zranitelnost vede k úniku a ztrátě dat nebo spuštění dalšího škodlivého kódu.

9. **Improper Assets Management**, kdy jsou v produkční verzi aplikace přístupné koncové body určené pro vývoj a ladění. Tyto endpointy nemusí být chráněny a mohou útočníkovi poskytovat větší manévrovací prostor pro provedení útoku.

10. **Insufficient Logging & Monitoring**, při které dochází k nedostatečnému logování a monitorování dat.

1.3.1 Broken Object Level Authorization

Autorizace na úrovni objektu je přístupový mechanismus, který ověřuje, že uživatel přistupuje pouze k těm datům a akcím, ke kterým by měl mít přístup, respektive oprávnění.

Útočník se během tohoto útoku snaží manipulovat s parametry, které jsou odesílány současně s požadavkem. To může vést nejen ke zpřístupnění potenciálně citlivých dat, ale i k jejich úpravě, potažmo smazání. [19]

1.3.2 Broken User Authentication

Při snaze zabránit útoku, který se zaměřuje na chyby v procesu autentizace, by měly API endpointy, které jsou zodpovědné za proces autentizace, mít implementovanou zvláštní úroveň ochrany. Vzhledem k faktu, že jsou volně přístupné, dá se očekávat, že budou napadány ve větší míře, než systémy přístupné po úspěšné autentizaci.

Zranitelnost odpovídá již dříve zmíněné zranitelnosti Identification and Authentication Failures. K výše zmíněnému textu je ještě vhodné doplnit, že (nejen) při procesu autentizace by v žádném případě neměly být v URL přítomny autentizační tokeny nebo hesla. [20]

1.3.3 Excessive Data Exposure

API zasílají citlivá data, která v mnoha případech obsahují více informací, než je ve skutečnosti nutné. Toto je následně odfiltrováno na straně uživatele, který tak tyto informace nevidí. Útočník ale může přenášené informace zachytit a zobrazit si nefiltrovaný obsah dat. Tedy obsah, který obsahuje ona citlivá data. [21]

1.3.4 Lack of Resources & Rate Limiting

Požadavky na API vytěžují obsluhující stroj z hlediska síťového provozu, využití procesoru nebo například paměti. V konečném důsledku mohou limity, které jsou nastavené nesprávným způsobem, vést k nedostupnosti API nebo služby samotné (DoS[9]).

API je zranitelné, pokud alespoň jeden z těchto prvků má nesprávným způsobem nastavené limity:

- časový limit provádění dotazu,
- maximální přidělenou paměť,
- počet požadavků na klienta a jiné. [22]

[9] Denial of Service

1.3.5 Broken Function Level Authorization

Zneužití této zranitelnosti spočívá v odeslání validního požadavku na API endpoint, ke kterému by ale neprivilegovaný uživatel neměl mít přístup. Tedy jak nepřihlášený uživatel, tak uživatel s neodpovídajícími právy. [23]

Zranitelnost opět odpovídá již dříve uvedenému bodu. Konkrétně se jedná o Broken Access Control.

1.3.6 Mass Assignment

Pro tento útok potřebuje útočník provést dotaz, například metodou GET, kdy si zobrazí data. Tím může být třeba stav účtu, který nějakým způsobem modifikuje. Takto upravená data poté odešle zpět, tentokrát ale metodou POST. Ačkoliv by k takovéto úpravě neměl mít oprávnění, přesto se mu podařilo změnit stav objektu (a zvýšit stav svého bankovního konta).

Obranou proti tomuto útoku může být vytvoření seznamu povolených akcí (tzv. *whitelist*), které mohou být na straně klienta prováděny. [24]

1.3.7 Security Misconfiguration

Ke zneužití této zranitelnosti dochází, pokud se využívají zastaralé systémy, systémy bez nejnovějších bezpečnostních záplat nebo jsou například na straně klienta přístupné informace pro ladění. [25]

Chybně nastavená konfigurace je problematikou, která je prakticky totožná s již dříve uvedenou sekcí Security Misconfiguration.

1.3.8 Injection

Pokud nedochází ke správnému ověřování dat zadávaných uživatelem, může se stát, že se útočníkovi podaří spustit kód, se kterým nebylo v původním návrhu počítáno. Pro minimalizaci rizik je vhodné data nejen ošetřovat, ale zároveň i omezit počet záznamů, které jsou uživateli vráceny. V krajním případě tak nedochází k získání všech dat, ale jen jejich části. [26]

Zranitelnost je totožná s bodem Injection, který byl uvedený v předchozí části textu.

1.3.9 Improper Assets Management

V této zranitelnosti využívá útočník funkcionality aplikací, které nejsou zdokumentované a nebo jsou nezabezpečené. Těmi mohou být například zastaralé endpointy, které jsou stále aktivní a připojené k produkční databázi. Nebo se naopak může jednat o nové endpointy, které jsou určené pro vývoj a neobsahují tak limity, které omezují počet dotazů, které může uživatel provádět. [27]

1.3.10 Insufficient Logging & Monitoring

Stejně jako u bodu Security Logging and Monitoring Failures, který byl uvedený v předchozí části textu, i v případě API je nutné provádět logování a jejich následný monitoring.

Bez správného logování a monitoringu má totiž útočník velké množství času, aby mohl svůj útok úspěšně dokončit a následně zneužít získané informace. [28]

Kapitola 2

Aplikace Seznamovák

Cílem této kapitoly je seznámit čtenáře s aplikací Seznamovák a důvody, proč je tato aplikace pro organizátory akcí Seznamovák a Magistrovák potřebná. Jsou zde představeny technologie, které tvoří webovou aplikaci Seznamovák, a ostatní aplikace, pro které tato webová aplikace představuje páteřní systém.

2.1 Seznamovací kurz Seznamovák

Seznamovák FIT [29] je akce, která se od roku 2010 koná každoročně na přelomu srpna a září. Slouží zejména k seznámení nových studentů, kteří nastoupili do bakalářského studijního programu na Fakultě informačních technologií ČVUT v Praze. Cílem je seznámit studenty mezi sebou a předat jim cenné rady o studiu od jejich starších a zkušenějších kolegů. Akce se koná vždy ve dvou turnusech, každý s kapacitou 90 účastníků.

V rámci celé akce jsou účastníci ubytováni v rekreačním areálu a každý den je pro ně připraven bohatý program, který obsahuje přednášky o studiu, volnočasové aktivity a hry. Ty je nejen sbližují, ale účastníci za ně i získávají body, na základě kterých jsou na konci celé akce ti nejúspěšnější odměněni a obdrží věcné ceny na památku.

Během celé akce se občas stane, že zasáhne nepřízeň osudu, a organizátoři jsou nuceni sáhnout ke změně programu. Například díky deštivému počasí bývá vyměněn dopolední a odpolední program. Nejedná se o nijak výjimečnou situaci, téměř každý rok probíhá pár takovýchto změn. Nicméně do té doby, než existovala mobilní aplikace Seznamovák, bylo třeba účastníky obejít a informace o změně jim předat ústně, přičemž se stávalo, že se k některým účastníkům informace ani nedonesla.

Pro potřeby organizace si dokonce organizátoři vytvořili vlastní webovou aplikaci, která umožnila podstatnou část věcí provádět automaticky a s vynaložením menšího úsilí. Aplikace vznikla v roce 2013 jako bakalářská práce Ing. Markéty Janochové [30], v průběhu let ale procházela postupným vývojem

a zapracováváním dílčích vylepšení a požadavků. Nevýhoda tohoto přístupu je neodhalení všech chyb a potenciálních zranitelností, které mohly v průběhu vývoje vzniknout.

Celkově je seznamovací kurz hodnocen každý rok velmi kladně, účastníci si odvážejí spoustu zážitků, mnoho nových informací nejen o studiu, ale zejména spoustu nových přátel [31]. Akce je díky velkému množství účastníků (ale také organizátorů) velmi náročná na organizaci, je potřeba připravit jednotlivé hry, vymyslet program, dohlížet na účastníky a mnoho dalšího.

Důležitou poznámkou je, že v roce 2019 byl vytvořen i úvodní kurz Magistrovák FIT [32]. Ten je určen pro nové studenty magisterského studijního programu. Magistrovák pokračuje ve stejném duchu jako Seznamovák, akce se koná pouze v omezenějším počtu 40 účastníků během jednoho turnusu, ale hry a jejich následné bodování a vyhodnocování probíhají obdobným způsobem.

2.2 Základní popis aplikace Seznamovák

Aplikace Seznamovák je poměrně komplexní webovou aplikací, která v sobě kombinuje několik více či méně populárních technologií. Základ aplikace ale tvoří webový PHP framework Nette [33, 34] a jako databáze je použita MariaDB [35]. Na vývoji této aplikace se podílelo několik vývojářů, hlavními vývojáři současné verze ale jsou Ing. Tomáš Nováček a Ing. Jan Horáček.

Uživatelé aplikace Seznamovák ji naleznou na webové stránce `https://seznamovak.fit.cvut.cz/aplikace/` odkud jsou hned přesměrováni na přihlášení `https://seznamovak.fit.cvut.cz/aplikace/prihlaseni`, protože nepřihlášený uživatel – tedy velmi pravděpodobně účastník – by neměl mít přístup k citlivým datům. Jedná se zejména o osobní informace o účastnících, popřípadě seznamy a popisy jednotlivých her, které by účastníkům mohly zkazit celkový zážitek z akce.

Pro účely testování existuje i testovací verze aplikace, která je dostupná na adrese `https://seznamovak.fit.cvut.cz/aplikace-test/`. Tato verze je velmi podobná verzi produkční, oproti produkční verzi ale například obsahuje data o účastnících, která jsou nějakým způsobem upravená. Je například provedena „anonymizace“, kdy jsou všechny e-maily nastaveny na testovací schránku, nebo zobrazen ladící nástroj Tracy [36].

Prakticky totožná aplikace, v mírně upravené podobě, je použita i pro akci Magistrovák. Ta vznikla tzv. *forkem*[10] ze současné webové aplikace Seznamovák. Díky tomuto kroku je náchylná ke stejným zranitelnostem jako aplikace původní. Tento *fork* je dostupný na `https://magistrovak.fit.cvut.cz/aplikace/`. Testovací verze Magistrovácké aplikace se analogicky nachází na adrese `https://magistrovak.fit.cvut.cz/aplikace-test/`.

[10] Jedná se o alternativní větev projektu, která bývá vyvíjena nezávisle na projektu původním.

2.3 Architektura aplikace

Použití frameworku Nette vedlo tvůrce původní aplikace Seznamovák k implementaci třívrstvého architektonického vzoru MVP. Model-View-Presenter je softwarová architektura vzniklá z architektury MVC[11]. Vrstvami MVP jsou:

- **Model** slouží k práci s daty a tvoří funkční základ a logiku celé aplikace. Kromě manipulace s databází zároveň komunikuje se síťovou vrstvou.
- **View** je vrstvou, která slouží k zobrazení výsledků jednotlivých požadavků. Slouží k interakci s uživatelem a následně předává požadavky Presenteru, který ovládá Model.
- **Presenter** je prostředníkem mezi Modelem a View. Načítá data z Modelu a následně je po zformátování předá pro zobrazení. Zároveň zpracovává reakci uživatele a udržuje stav perzistentních proměnných. [37]

2.4 Použité technologie

V této sekci jsou stručně shrnuty nejdůležitější technologie, které tvoří aplikaci a které se používají pro správné fungování celého projektu. Celkový výčet všech knihoven a závislostí je uveden v příloze B.

2.4.1 PHP

PHP je interpretovaný skriptovací programovací jazyk, který je svou syntaxí silně ovlivněn jazykem C. Využívá se zejména pro tvorbu dynamických webových stránek a aplikací. PHP je jazykem vyvinutým na konci 90. let, stále se ale jedná o populární jazyk, který i dnes využívá zhruba 80 % webových stránek [38]. Přesto nebývá širokou veřejností přijímán jako bezpečný, naopak bývá považován za *insecure by default* [39].

Většinou se po dotazu klienta spustí PHP skript, který zobrazí výsledek skriptu například ve formě HTML[12]. Použitím PHP nedochází k vytváření žádných perzistentních proměnných, po vykonání požadavku si server neukládá žádné informace o proběhlé akci.

2.4.2 Nette

Nette framework je open source framework určený pro tvorbu webových aplikací pro PHP 5 a novější. Nette bylo vytvořené českým autorem Davidem Grudlem v roce 2008 a od té doby se těší velké oblíbenosti ze strany českých vývojářů, pravděpodobně i díky obsáhlé dokumentaci psané v českém jazyce.

[11] Model-View-Controller
[12] Hypertext Markup Language

Framework se zaměřuje na eliminaci bezpečnostních chyb a rizik, úspěšně prošel mnoha bezpečnostními audity a jedná se (dle vlastních slov) o „nejbezpečnější PHP framework“ [40]. Pro rok 2015 se také jednalo o třetí nejpopulárnější framework na světě [41].

2.4.3 MariaDB

MariaDB [35] je multiplatformní databázový systém pro dotazovací jazyk SQL. Jedná se o relačně databázový systém typu DBMS[13], který klade důraz na rychlost a výkon, mnohdy za cenu zjednodušení určitých částí, například způsobu zálohování.

Projekt vznikl z původního projektu MySQL[14] [42], který byl vyvinut švédskou firmou MySQLAB v roce 1995. Po odkoupení původního projektu společností Oracle došlo k *forku* zejména z důvodu zachování svobodné licence GNU GPL[15] [43].

2.4.4 Composer

Composer [44] je nástroj, který sice není esenciálně nutný pro vývoj, ale umožňuje jednoduchou správu závislostí a knihoven pro programovací jazyk PHP. Umožňuje pomocí příkazové řádky a předem definovaného JSON[16] souboru nainstalovat rozšíření třetích stran do vlastního projektu. Při instalaci dojde ke stažení kódu z centrálního repozitáře do projektu. Díky tomuto nástroji je možné velmi snadno udržovat správné závislosti a mít projekt aktualizovaný na nejnovější verze.

2.4.5 Apache

Apache [45] je multiplatformní open source webový server, který vznikl v roce 1993 na Illinoiské univerzitě. I přes své stáří je dnes stále jedním z nejrozšířenějších webových serverů (hned po Nginx), i když se sestupnou tendencí [46].

Server podporuje – díky různým modulům rozšiřujícím jádro – nemalé množství programovacích jazyků, autentizačních schémat, podpory TLS, komprese posílaných webových stránek a mnoho dalších funkcí.

[13] Database Management System
[14] My Structured Query Language
[15] GNU General Public License
[16] JavaScript Object Notation

2.5 Přidružené aplikace

Za dobu své existence se aplikace Seznamovák stala páteří pro několik dalších aplikací, které se ukázaly jako více či méně zásadní pro správné fungování celé akce. Díky své existenci vedla aplikace k ušetření času organizátorům, kteří se následně mohli (a nadále mohou) věnovat účastníkům s jejich dotazy a stmelováním.

Ke konci února 2022 existuje šest aplikací, pro které webová aplikace Seznamovák představuje páteřní aplikaci a poskytuje jim technické zázemí. Těmito aplikacemi jsou:

- **Seznamovák** [47] je mobilní aplikací pro účastníky akce, která je určena pro mobilní telefony se systémem Android. Aplikace byla vytvořena v rámci bakalářské práce Bc. Michaely Kučerové [48].
- **Seznamovák** [49] je mobilní aplikací pro účastníky akce, která je určena pro mobilní telefony se systémem iOS. V současné době ale není tato mobilní aplikace udržována.
- **Cerberus** je mobilní aplikací pro organizátory, která slouží pro úvodní příběhovou seznamovací hru Cerberus.
- **Kolíčkovaná** je mobilní aplikací pro organizátory, kteří mají na starosti hru Kolíčkovaná[17].
- **Zpěvník** [50] je mobilní aplikací pro účastníky i organizátory. Jedná se o databázi písniček, zejména táborových.
- **Web** jsou webové stránky celé akce.

Účastnické aplikace umožňují (po přihlášení) jednotlivým registrovaným účastníkům zobrazit si aktuální počet bodů ze seznamovacích her, harmonogram akce a hlavně notifikace, které umožní reagovat na nestandardní situace. Zároveň aplikace slouží jako kontrola pro hru Kolíčkovaná.

Cerberus umožňuje stáhnout si vygenerované seznamy účastníků, respektive seznamy týmů, do kterých jsou studenti přiřazeni, a následně spolu účastníci a organizátoři prochází příběhovou hrou.

Aplikace Zpěvník slouží k pasivnímu zobrazování dříve zadaných skladeb, které se následně hrají u táborového ohně. Aktuální seznam skladeb je spravovaný webovou aplikací, která záznamy následně hromadně odesílá do zpěvníku. Zpěvník je v plánu v budoucnu sloučit s účastnickou aplikací.

[17] Hra je známější pod obecným názvem Mafia. Účastníci i organizátoři dostanou na začátku akce kolíček se jménem svého cíle. Vrah musí (předem definovaným nenásilným způsobem) „sprovodit ze světa“ svůj cíl, který mu následně předá svůj kolíček a po ověření v aplikaci jsou vrahovi přiděleny body a hra může pokračovat dál.

Webové stránky Seznamováku slouží k prezentaci celé akce, web zobrazuje informace o aktuálním ročníku, informace o organizátorech a předchozích ročnících. Všechny tyto informace jsou dynamicky generované na základě dat z aplikace. Na webových stránkách se zároveň mohou účastníci (po vyplnění osobních údajů) zaregistrovat k účasti na aktuální ročník akce.

2.6 Endpointy

Backend aplikace Seznamovák zároveň poskytuje REST[18] API pro získávání potřebných dat pro mobilní aplikace. Data jsou dostupná na jednotlivých endpointech ve formátu JSON. Jedná se o tyto endpointy:

- **/get-mobile-user-data-json/{e-mail}** je endpointem, který poskytuje informace o uživateli s danou emailovou adresou. Informace zahrnují unikátní identifikátor, turnus a identifikátory pro hru Kolíčkovaná.
- **/news-json/{turnus}** poskytuje seznam všech novinek, které přidávají organizátoři pro zadaný turnus. V případě, že není turnus specifikován, jedná se o novinky, které jsou určeny pro celý ročník akce.
- **/harmonogram-json/{turnus}** nabízí jednotlivé položky harmonogramu, které jsou rozděleny podle dnů. Každá z položek na harmonogramu obsahuje kromě názvu i informace o začátku a konci události.
- **/organizer-harmonogram-json/** endpoint se podobá harmonogramu pro účastníky, nabízí ale informace relevantní pro organizátory.
- **/points-json/{ID}** obsahuje seznam her a počet bodů, které v každé z her získal účastník se zadaným *ID*.
- **/mafia-info/{turnus}/{ID}/{isOrganizer}** je stěžejní endpoint pro hru Kolíčkovaná. Nabízí informace o hráči se zadaným *ID*. Parametr *isOrganizer* je bool hodnotou, která určuje, jestli je uživatel organizátor nebo účastník. Detailní informace nejsou pro tuto práci důležité, více informací lze najít v [48].
- **/mafia-kill/{turnus}** na tento endpoint je dotazováno při zabíjení ve hře Kolíčkovaná. Opět se jedná o nepodstatné informace vzhledem k zaměření práce.
- **/teams/{číslo týmu}/{turnus}** nabízí informace podstatné zejména pro hru Cerberus a mobilní aplikaci téhož jména.
- **/songs-json/** obsahuje seznam písniček, které jsou využívány aplikací Zpěvník.

[18] Representational State Transfer

Kapitola 3

Analýza aplikace

V následující kapitole je popsána aplikace Seznamovák z pohledu účastníka a organizátora. Zároveň jsou zde popsány jednotlivé role pro organizátory (a jejich následné přístupy k různým druhům informací) a další způsoby, jakým je implementováno zabezpečení dat aplikace.

3.1 Aplikace z pohledu účastníka

Účastník Seznamováku s aplikací přijde do kontaktu jen minimálně. Registrace účastníka probíhá přes formulář uvedený přímo na hlavní stránce webu akce [29]. Pokud si ale chce zobrazit své vyplněné údaje nebo například provést nějaké změny, musí se už do aplikace přihlásit přes svůj unikátní klíč. I přes tyto informace ale nemá standardně běžný účastník žádné povědomí o rozsahu aplikace a ani jejím určení.

3.1.1 Registrace účastníka

Registrace účastníků probíhá přes formulář v pevně daném období roku. Toto období začíná tradičně řádnými zápisy do studia na fakultě a končí naplněním kapacit obou turnusů a kapacit náhradníků, což se liší rok od roku. Účastníci zadávají své osobní údaje nutné pro organizaci. Mezi tyto údaje patří:

- jméno,
- příjmení,
- datum narození,
- adresa trvalého bydliště,
- telefonní spojení a
- zdravotní informace (alergie, popřípadě pohybová či jiná omezení).

Po úspěšné registraci je účastník přesměrován na webovou stránku, kde jsou mu zobrazeny údaje o platbě a další důležité informace. Tyto informace jsou mu současně odeslány na e-mail.

Heslo pro přihlášení se generuje s pomocí metody `Random::generate`. Tato metoda je součástí statické třídy určené pro generování pseudonáhodných řetězců z frameworku Nette. Metoda je volána pouze s určením počtu znaků, uplatní se tedy výchozí chování a mezi vygenerovanými znaky jsou pouze čísla a znaky malé abecedy [51].

Proces generování účastnických klíčů v sobě ukrývá několik bezpečnostních chyb, mezi které patří:

1. Malý prostor možných klíčů. Obranou je rozšířit prostor možných klíčů. Toho lze docílit přidáním druhého argumentu do funkce, která se stará o generování klíčů. Například `Random::generate(10, '0-9a-zA-Z')` umožní vygenerovat i velká písmena z abecedy.

2. Účastnická hesla jsou přímo porovnávána s klíčem uloženým v databázi, nejsou nijak hashována a proto mohou být zranitelná proti časové analýze neboli tzv. *timing attack*[19]. Správným řešením by mělo být hesla hashovat.

3. Malý počet znaků pro účastnický klíč. Ten je vhodné rozšířit, alternativně je možné stávající počet ponechat, ale je nutné potenciálního útočníka zpomalit, například použitím pomalé hashovací funkce.

3.1.2 Přihlášení účastníka

Účastník se může přihlásit na stránce `/ucastnici/ucet` skrze desetiznakové heslo (klíč), které mu bylo odesláno současně s registrací na uvedený e-mail. Po přihlášení účastník vidí své osobní informace, které zadal při registraci. Dále vidí, jestli už za Seznamovák zaplatil, či nikoliv.

Útočník může ze svého hesla (například `aer0x36ubc`) odhadnout strukturu ostatních hesel. Této znalosti lze následně využít pro modifikaci metody POST a upravit odesílaný JSON uvedený na výpisu kódu 1. Formulář zároveň neumožňuje zadávat delší hesla, než je 10 znaků. Lze tedy předpokládat, že klíč bude mít přesně danou délku.

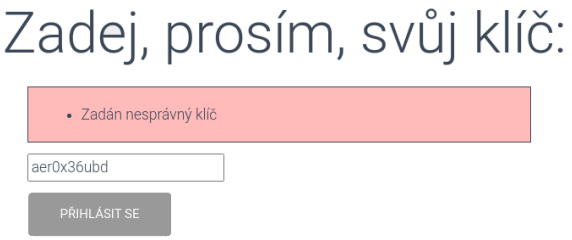
Pro útok může útočník kontrolovat návratový HTTP kód. Nevalidní uživatelský klíč je zobrazen v rámci stejné stránky s varovným textem „Zadán nesprávný klíč“ (obrázek 3.1) a HTTP požadavkem `200 OK`. Správný klíč uživatele přesměruje na vlastní stránku (obrázek 3.2) a návratovým kódem je tentokrát `303 See Other`.

Základem pro skript určený k útoku může být například kód uvedený ve výpisu kódu 2. Útočníka zajímá jen HTTP status kód, výstup jako takový

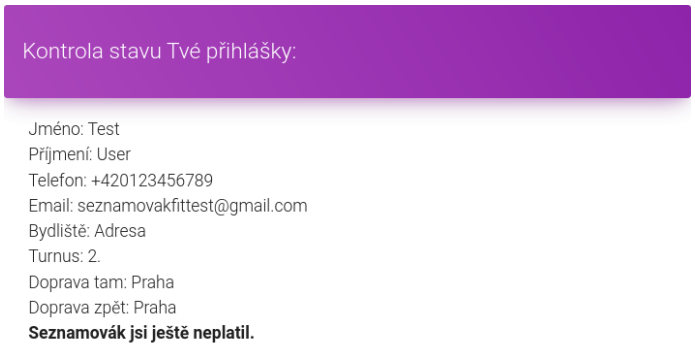
[19] Jedná se o útok postranním kanálem, při němž útočník měří, jak dlouho trvají odpovědi na různé vstupy.

```
{
    "key": "aer0x36ubc",
    "send" : "Přihlásit se",
    "_do" : "participantKey-participantKey-submit"
}
```

Výpis kódu 1: Odesílaný JSON pro přihlášení účastníka



Obrázek 3.1: Neplatné uživatelské heslo



Obrázek 3.2: Uživatelský profil po zadání platného hesla

```
for i in <iterace skrz možné klíče>; do
    status_code=$(curl -s -o /dev/null -w "%{http_code}" \
      '<adresa>/ucastnici/ucet' -H 'Cookie: _nss=1;' \
      --data-raw <upravený zakódovaný JSON>)
    if [ $status_code -eq 303 ]; then
        echo "$i"
    fi
done
```

Výpis kódu 2: Příklad skriptu pro zjištění účastnických klíčů

je potlačený (parametr `-o`). Důležitým parametrem je doplnění požadavku o detekci CSRF[20] útoku (`_nss=1`). Bez této kontroly je požadavek zahozen.

Obranou proti tomuto scénáři útoku je implementace oprav, které byly navrhnuty v předchozí sekci.

3.2 Aplikace z pohledu organizátora

Obrázek 3.3: Úvodní stránka aplikace Seznamovák

Jelikož aplikace obsahuje citlivé informace o účastnících, je přístup k nim podmíněn uživatelským účtem a následným přihlášením. Po úspěšném přihlášení je organizátor přesměrován na úvodní stránku webové aplikace Seznamovák viditelnou na obrázku 3.3.

Jak je možné si povšimnout, aplikace je v současné době poměrně robustní a umožňuje rozsáhlé množství akcí. Zároveň ale díky dělení zodpovědnosti má každý organizátor na starost jen určitou podmnožinu akcí, ačkoliv může přistupovat (v rámci své role) k libovolné z nich. Mezi typické operace, které organizátoři v aplikací provádějí, patří například:

- uchovávat a editovat informace u účastnících,
- zasílat hromadné e-maily účastníkům a organizátorům,
- generovat různé účastnické týmy pro jednotlivé hry (například podle studijních kruhů v prvním semestru),
- obodovat různé hry a generovat z nich výsledkové listiny,
- zadávat a spravovat úkoly pro organizátory,

[20] Cross-Site Request Forgery

- vytvářet akce pro organizátory (například organizační schůze) a odeslat o nich upozornění,
- zasílat notifikace do mobilní aplikace,
- udržovat seznam písní ve zpěvníku.

Každý organizátor si navíc může upravovat svůj uživatelský profil, aktualizovat fotografii či záliby. Tyto informace jsou poté vytištěny a jsou přístupné jednotlivým účastníkům na samotné akci. Zároveň jsou změny díky dynamicky generovanému webu ihned propsány na hlavní webovou stránku akce.

3.2.1 Registrace uživatele

Registrace nového uživatele – na rozdíl od registrace účastníka – není omezena žádným časovým obdobím. Na stránce `/organizatori/registrace/` vyplní nový uživatel základní informace jako jsou:

- uživatelské jméno,
- heslo,
- e-mail a
- kontrolní heslo.

Kontrolní heslo je sděleno novým organizátorům před samotnou registrací do aplikace. Ačkoliv se nejedná o velmi důmyslnou formu ochrany, je přesto tento postup funkční a použitelnou ochranou před uživateli a roboty, kteří nemají mít do aplikace přístup.

Všechny uživatelské účty jsou zároveň opatřeny příznakem, jestli je uživatelský účet povolený, tedy jestli se jedná o aktivního organizátora, který by měl mít k určitým informacím přístup. V případě, že uživatelský účet není povolený, proces přihlašování selže a uživateli se zobrazí varování s textem „Uživatel není povolen“.

Už od procesu registrace vystavuje ale aplikace Seznamovák své uživatele potenciální možnosti útoku. Přestože nenutí své uživatele následovat zastaralá pravidla pro tvorbu „bezpečného“ hesla, neřídí se ani doporučením NIST[21] [52], podle kterého by mělo být heslo silné a dostatečně dlouhé. Uživatel by měl upřednostnit tzv. *passphrase* – heslovitou frázi o několika náhodně zvolených slovech.

Aplikace nijak neupravuje požadavek na minimální délku hesla. Teoreticky je tedy možné zvolit pouze jednoznakové heslo. Pro opravu stačí přidat pouze jedno pravidlo pro kontrolu délky zadaného hesla. Správná validace proto může být například jako na výpisu kódu 3. Konstanta `MIN_PASS_LEN` je nastavena na 8 znaků, což je v souladu s doporučením NIST.

[21] National Institute of Standards and Technology

```
$form->addPassword('password', 'Heslo')
     ->addRule(Form::FILLED, 'Zadejte heslo.')
     ->addRule(Form::MIN_LENGTH,
               'Minimální délka hesla je %d znaků',
               self::MIN_PASS_LEN);
```

Výpis kódu 3: Opravená část formuláře pro registraci uživatele

```
public function authenticate(array $credentials): IIdentity {
    list($username, $pass) = $credentials;
    $r = $this->database->table('user')
              ->where('username', $username)->fetch();

    if (!$r)
        throw new Security\AuthenticationException(
            'Zadáno špatné uživatelské jméno nebo heslo.',
            self::IDENTITY_NOT_FOUND);

    if ($r->pass !== $this->calculateHash($pass, $r->pass))
        throw new Security\AuthenticationException(
            'Zadáno špatné uživatelské jméno nebo heslo.',
            self::INVALID_CREDENTIAL);

     if (!$r->allow)
        throw new Security\AuthenticationException(
            'Uživatel není povolen.',
            self::INVALID_CREDENTIAL);

    return new Identity($r->id, $r->role, $r->toArray());
}
```

Výpis kódu 4: Funkce pro autentizaci uživatele

3.2.2 Přihlášení uživatele

Proces přihlášení uživatele je zajištěn implementací funkce `authenticate`, která je uvedena ve výpisu kódu 4. V implementaci této funkce je možné zpozorovat několik prohřešků a bezpečnostních chyb:

1. Přímé porovnání dvou hashů může být potenciálně nebezpečné a zranitelné proti *timing attack*. Pro účely porovnání dvou hashů existuje v PHP například funkce `password_verify` nebo `hash_equals`. Obě tyto funkce jsou proti útoku časováním imunní [53, 54].

2. Použití zastaralé třídy `Identity`. Tato třída byla v Nette 3.1 přejmenována na `SimpleIdentity` a ačkoliv zatím funguje jako alias, v dalších verzích Nette bude úplně odebrána.

3. Rozhraní třídy `authenticate` se změnilo. Proměnná `$credentials` je nyní nahrazena proměnnou `$username` a `$password`.

3.2.3 Změna hesla

Zatímco přihlášenému účastníkovi Seznamováku není změna hesla (respektive přihlašovacího klíče) dovolena, organizátor si své heslo změnit může. Pokud se pro změnu hesla rozhodne, musí vyplnit tři údaje:

- staré heslo,
- nové heslo a
- nové kontrolní heslo.

Toto je standardní proces, který je implementovaný i v aplikaci. Bohužel zde opět chybí kontrola na délku hesla. Pro opravu můžeme opět využít přidání pravidla na minimální délku hesla, stejně jako v případě registrace nového uživatele, uvedenému na výpisu kódu 3.

Dalším nesprávným chováním je redundance kódu na ověření, jestli se nové heslo shoduje s kontrolním novým heslem. Kontrola je zajištěna už na straně formuláře díky pravidlu `Form::EQUAL` z frameworku Nette.

Naopak správně se aplikace chová v případě, že uživatel zadá špatně staré heslo. Tehdy ke změně hesla nedochází.

Uživatel může požádat o obnovu zapomenutého hesla skrz možnost „Obnova hesla“, kde vyplní své uživatelské jméno a na e-mailovou adresu, uvedenou v registraci, mu dorazí nově vygenerované heslo o délce 22 znaků. E-mail je šifrovaný, nedochází tedy k posílání nového hesla v čitelné podobě. Přesto se nejedná o osvědčený postup. Tím by bylo odeslat na e-mail odkaz, ze kterého dojde k samotné obnově hesla [55].

3.3 Role a přístupy

Omezení přístupu na základě rolí je obvyklým postupem, jak delegovat zodpovědnost a zamezit uživateli bez oprávnění provést potenciálně nebezpečnou akci. Aplikace Seznamovák tento postup dodržuje a implementuje několik rolí, kdy každá z nich má jiná oprávnění. Mezi definované role patří:

- Výpomoc,
- Uživatel,

- Správce (bankovního) účtu,
- Administrátor,
- Vývojář.

Výpomoc je role, která má minimální přístup k informacím. Uživatelé s touto rolí bývají většinou přizvání mezi organizátory na poslední chvíli, zejména z důvodu neúčasti některého z hlavních organizátorů. V rámci své role vidí Výpomoc informace o všech ostatních organizátorech, konaných událostech, zadaných úkolech, ale nemá přístup k informacím o účastnících. Výpomoci není umožněno vyhodnocovat hry (tzv. „obodovávat“ hry), ani posílat hromadné e-maily z aplikace.

Uživatel je standardní rolí pro organizátora. Má přístup ke všem informacím o studentech, které může i editovat. S rolí uživatele lze provádět veškeré operace, které jsou zmíněné v sekci 3.2.

Správce bankovního účtu je uživatel, který kromě role „obyčejného“ organizátora dohlíží na správu financí celé akce. Zatímco normální uživatelé mohou požádat o proplacení nákupů na Seznamovák, Správce tyto platby může autorizovat. Po úspěšné autorizaci žádosti pak odešle zadaný obnos peněz na účet organizátora, který tuto platbu zadal.

Administrátor má stejná práva jako správce účtu, navíc ale může:

- měnit role pro ostatní organizátory,
- upravovat proměnné použité v aplikaci (například kapacity turnusů),
- provádět zálohy databáze,
- obnovit stav aplikace ze záloh a
- připravit aplikaci pro další ročník Seznamováku (a současně provést zálohu databáze).

Role vývojáře je přiřazována bývalým organizátorům, kteří se podíleli na vývoji celé aplikace a slouží jako jakási technická podpora. Jejich přístup je proto ospravedlněn zejména opravou chyb v aplikaci. U této role není na webu viditelný profil s medailonkem, vývojář se již aktivně nepodílí na organizování akce. Jinak je role ekvivalentní s rolí Administrátor.

Existence role Výpomoc je z hlediska návrhu aplikace značně kontraproduktivní, organizátor-výpomoc se totiž nemůže plně věnovat samotnému organizování a tak deleguje svou práci na ostatní organizátory. Role je tak značně redundantní.

Ačkoliv role Správce bankovního účtu neumožňuje v základu měnit proměnné aplikace – například částku, kterou zaplatí účastník za akci, potažmo za tričko – je mu umožněno přes rozhraní `Finance` měnit nastavení právě

(a) Nastavení administrátora (b) Nastavení pro roli správce účtu

Obrázek 3.4: Nastavení proměnných v aplikaci

```
public static function calculateHash($pass, $salt = null) {
    if ($pass === Strings::upper($pass))
        $pass = Strings::lower($pass);
    return crypt($pass, $salt ?: '$2a$07$'.Random::generate(22));
}
```

Výpis kódu 5: Funkce pro výpočet hashe hesla

těchto atributů. Dochází tak k duplikaci nastavení proměnné a de facto k porušení oprávnění, která měla zabezpečit role. Duplikace nastavení proměnných je vidět na obrázku 3.4.

3.4 Hashování hesel

Hesla v databázi by nikdy neměla být uložena v čitelné podobě například ve formě tzv. *plaintextu*, nebo nějak obfuskována. Z toho důvodu je v databázi uchováván otisk, neboli *hash* hesla.

Pro tento účel je v PHP možné využít například hashovací algoritmus Bcrypt, který je implementovaný ve funkci `crypt`. Ačkoliv je ve webové aplikaci Seznamovák použit právě tento algoritmus, díky kombinaci několika faktorů je účinnost algoritmu podstatně snížena.

Jak je vidět z výpisu kódu 5, funkce `calculateHash` se stará o výpočet hashe ze zadaného hesla. Implementace této funkce je ale problematická hned z několika důvodů:

1. Podmínka testuje, zda je uživatelem zadané heslo psané verzálkami[22], a tím potenciálně snižuje entropii hesla. Tedy činí jisté metody útoku o něco snadnějšími.

[22] jinak řečeno jen velký písmeny

Cost	7	8	9	10	11	12
Čas [ms]	6.5	12.9	25.7	51.3	102.9	203.5

Tabulka 3.1: Rychlost výpočtu hashe v závislosti na parametru cost

2. Používání funkce `crypt` není dle manuálových stránek této funkce [56] ideální. Vhodnější alternativou je použití funkce `password_hash`. Tato funkce slouží jako tzv. *wrapper* funkce `crypt`, a tím odstiňuje programátora od implementačních detailů.

3. Používání čísel v kódu namísto definovaných konstant.

4. Použití funkce `Random::generate` bez explicitního zadání parametru pro znaky. Výchozí chování je omezeno na znaky `'0-9a-z'`, což opět snižuje entropii hesla, respektive soli.

5. Použití varianty `$2a$` algoritmu Blowfish. V roce 2011 byla nalezena zranitelnost týkající se hesel, která obsahovala znaky mimo standardní ASCII znaky. Po opravě je přesto pro nová hesla doporučováno používat variantu `$2y$` algoritmu [57].

6. Nedostatečný počet rund algoritmu Blowfish. Počet rundovních iterací (tzv. *cost*) je možné poznat z parametru `$07$`, jedná se tedy o $2^7 = 128$ rundovních operací. Dle dokumentace funkce `passwordHash` je doporučovaný *cost* závislý na cílové platformě, ale funkce by měla trvat méně než 100 ms [58]. Na serveru aplikace Seznamovák funkce s parametrem `$07$` trvá zhruba 6 ms. V tabulce 3.1 jsou uvedeny naměřené časy pro výpočet rychlosti v závislosti na zvyšujícím se čísle *cost*.

7. Ačkoliv to není na první pohled viditelné, tato funkce existuje v projektu dvakrát. Jednou v souboru `UserRepository.php` a podruhé v souboru `Authenticator.php`. Duplikace kódu je téměř vždy nežádoucí a je nejen z pohledu bezpečnosti špatná.

Pokud bychom používali pro hashování funkcí `password_hash`, už nemusíme (a ani bychom neměli) používat parametr `salt`. Ve verzi PHP 7.0 vede použití tohoto parametru na varování. V novějších verzích PHP navíc může být tento parametr odebrán úplně.

Opravená funkce by mohla vypadat například jako ve výpisu kódu 6. Alternativně lze využít samotného frameworku Nette a z balíčku `nette/security` použít třídu `Passwords`, respektive její metodu `hash`. Toto můžeme vidět na výpisu kódu 7. Samozřejmě proměnnou `$passwords` by bylo vhodnější mít uloženou například ve třídě `Authenticator`.

```
public static function calculateHash($pass) {
    return password_hash($pass, PASSWORD_BCRYPT, ['cost' => 11]);
}
```

Výpis kódu 6: Opravená funkce pro výpočet hashe hesla

```
public static function calculateHash($pass) {
    $passwords = new Passwords(PASSWORD_BCRYPT, ['cost' => 11]);
    return $passwords->hash($pass);
```

Výpis kódu 7: Alternativní opravená funkce pro výpočet hashe hesla

3.5 Zabezpečení API

Jak již bylo popsáno v sekci 2.6, aplikace poskytuje REST API. Zabezpečení přístupu je realizované přes heslo, které se vkládá do url. Bez hesla je uživateli vrácen chybový kód se zamítnutím přístupu. Skutečná adresa tak vypadá například `/api/songs-json/<heslo>`.

Data nejsou zabezpečena proti přístupu takřka vůbec (nebo jen velmi slabě), spoléhá se na neznalost účastníků, popřípadě útočníků. Jedná se proto o *bezpečnost skrz neznalost*[23].

Ke zjištění, že komunikace s endpointy není nijak zabezpečena, došlo během roku 2018. Seznamováku konaném v tomto roce se účastnil student, který si nainstaloval mobilní aplikaci pro Android, kterou následně analyzoval z pohledu komunikace s vnějším světem. Díky pevně zvolenému heslu, které je uloženo ve zdrojových kódech mobilní aplikace, se mu podařilo poměrně snadno odhalit mechanizmus přístupu k endpointu.

Ačkoliv útočník neprováděl žádné úpravy, už pouhý fakt, že to lze udělat, je dostatečným důkazem, že současné zabezpečení je nedostatečné.

[23] Známější spíše pod anglickým názvem Security through obscurity.

Kapitola 4

Penetrační testování

V této kapitole jsou popsány některé nástroje určené k penetračnímu testování, které jsou následně použity pro skenování portů a nalezení dalších zranitelností v aplikaci Seznamovák. Testování aplikace a API je provedeno dle OWASP Top 10, resp. OWASP API Security Top 10.

4.1 Použité nástroje

Penetrační testování se dnes už takřka neobejde bez specializovaných nástrojů, které umí mnoho útoků provést automatizovaně. Nástroje ve svých databázích také obsahují velké množství skriptů, postupů a payloadů, které lze využít pro zneužití nalezených zranitelností. Útočník tak může použít určitý payload a na napadeném stroji získat administrátorská (root) práva, instalovat různé programy pro logování nebo třeba pořizovat screenshoty.

4.1.1 Nmap

Nmap [59], neboli Network Mapper je nástroj, který slouží k bezpečnostnímu skenování síťových portů. Program dokáže odhalit i další podrobnosti, například operační systém cílového počítače, jména a verze služeb, které naslouchají, jak dlouho je systém online apod. Od verze 4.5 vydané roku 2007 obsahuje nástroj i NSE[24], což je řada skriptů, které lze využít k detailnějšímu zkoumání tradičních zranitelností. Tyto skripty jsou rozděleny do několika kategorií, kde každá z nich se zaměřuje na jinou část analýzy. Kategoriemi jsou dle [60] například:

- **brute** je kategorií skriptů, která se zaměřuje na zjištění přístupových údajů na vzdálené servery.

[24] Nmap Scripting Engine

- **version** je speciální kategorií, která se snaží zjistit přesné verze služeb a nástrojů, které jsou na webech použity.
- **vuln** tato kategorie se zaměřuje na kontrolu známých zranitelností, ale omezuje se na pouhé zjišťování, jestli jimi dané služby trpí.
- **exploit** kategorie jde ještě dále a aktivně se snaží objevené zranitelnosti zneužít.

4.1.2 Nikto

Nikto [61] je nástroj, který slouží k hledání známých zranitelností na webových stránkách. Zaměřuje se na rychlé zkoumání existujících souborů, nastavení serverů, HTTP hlaviček a použitých nástrojů.

Program umožňuje provádět analýzu jak v obecné rovině (tj. hledat obecné zranitelnosti nezávisle na použitém serveru), tak útoky, které cílí na konkrétní použitou instanci serveru, na které běží daná webová aplikace. [62]

4.1.3 Dirb

Nástroj Dirb [63] se zaměřuje na hledání existujících souborů a adresářů na zadaných webových stránkách. Pro hledání se využívá slovníkový útok, popřípadě slovníkový útok s pravidly. V obou případech se dá využít výchozí slovník, který obsahuje klíčová slova, která se dnes používají na webových stránkách. Díky tomuto přístupu se tak může podařit odhalit skryté a nebo zapomenuté soubory, které mohou útočníkovi odhalit mnoho informací, které zvyšují šanci na úspěšnou kompromitaci systému. [64]

4.1.4 Metasploit Framework

Metasploit Framework [65] je nástroj určený pro skenování systému a následnou přímočarou konfiguraci systému pro použití *exploitů* v závislosti na nalezených zranitelnostech. Program nabízí širokou škálu *payloadů*, které lze libovolně měnit a kombinovat s *exploity*. [66]

4.1.5 OWASP ZAP

OWASP ZAP [67] je dalším z nástrojů, které se využívají pro skenování webových aplikací. Program je zaštiťován a vyvíjen pod hlavičkou organizace OWASP, pro kterou představuje jeden z nejvíce aktivních projektů, které spravuje.

Program umožňuje zachytit (a v případě potřeby i pozměnit) zachycenou komunikaci pořízenou během automatického nebo manuálního penetračního testu.

```
$ sudo nmap -sS -sV 147.32.232.27 -O
  # zkrácený výstup
  PORT    STATE SERVICE  VERSION
  22/tcp  open  ssh      OpenSSH 7.9p1 Debian 10+deb10u1
  80/tcp  open  http     Apache httpd 2.4.38
  443/tcp open  ssl/http Apache httpd 2.4.38 ((Debian))
```

Obrázek 4.1: Zjištěné verze služeb pro otevřené porty (TCP SYN scan)

4.2 Skenování portů

Skenování otevřených portů je jednou z nejčastějších metod, které se využívají pro hledání zranitelností, a je zároveň první, která se testuje a podle které se určuje následný postup útoku.

Tradičně se pro zjišťování portů využívá nástroj Nmap, který umožňuje několik různých technik skenování. Nalezené porty jsou následně rozděleny do několika kategorií. Nejběžněji se jedná o dělení na otevřené, zavřené a filtrované, ačkoliv Nmap umožňuje ještě jemnější granularitu dělení. Útočníka zajímají zejména porty označené jako otevřené. Jak je možné vidět na obrázku 4.1, na serveru Seznamováku se jedná o tyto tři porty:

- 22/tcp
- 80/tcp
- 443/tcp

4.2.1 Port 22/tcp

Nástroj Nmap nalezl několik zranitelností, které byly ale většinou cílené na klienta, což při útoku na server není bohužel použitelné. Podařilo se alespoň zjistit informace o serveru a jeho službách. Je zde použito OpenSSH verze 7.9p1 a operačním systémem je Debian ve verzi 10.1. Dále se na tomto portu nepodařilo nalézt žádnou zranitelnost, kterou by bylo možné využít k přístupu na server.

Součástí analýzy byl i útok hrubou silou, který cílil na přihlášení se na server skrz SSH. Útok se nevydařil, přístup na server se nepodařilo získat.

4.2.2 Port 80/tcp

Na portu 80 běží služba HTTP, respektive server Apache. Kvůli nastavení serveru, který odesílá svou přesnou verzi v HTTP hlavičce v položce „Server“, se podařilo zjistit přesnou verzi Apache serveru. Jedná se o verzi 2.4.38.

Kvůli zjištění konkrétní verze serveru se útočník může omezit na hledání zranitelností v konkrétní verzi. Ačkoliv bylo několik zranitelností nalezeno, většina z nich se spoléhá na používání konkrétních modulů na serveru, které se ale na serveru aplikace Seznamovák nepoužívají. Popřípadě zranitelnosti vyžadují, aby měl uživatel přístup na server a následně mohlo dojít k eskalaci práv obyčejného uživatele.

Pomocí nástroje Nikto se podařilo objevit i několik dalších nedostatků, mezi které patří nenastavování určitých bezpečnostních HTTP hlaviček. Těmito hlavičkami jsou:

- **X-Frame-Options**, kdy přítomnost této hlavičky informuje prohlížeč o tom, jestli má povolit vykreslování stránky například pomocí `<frame>`. Webové stránky používají toto nastavení, aby zabránily tzv. *click-jacking* útokům, tedy vložení útočníkova obsahu (například škodlivého skriptu) do průhledné vrstvy nad vloženou stránkou.

- **X-Content-Type-Options**, kdy nastavení této hlavičky serverem informuje prohlížeč, aby neodhadoval typ souboru, nebo média, a řídil se pouze předanými informacemi.

4.2.3 Port 443/tcp

I na portu 443 je přítomný webový server Apache. Jeho přesnou verzi se podařilo určit opět díky špatně nastavené hlavičce, která v sobě obsahuje přesnou verzi služby a informaci o použitém operačním systému.

Stejně jako v předchozím případě, server nastavuje bezpečnostní hlavičky, které jsou uvedené výše u portu 80/tcp. Špatně nastavenými a nebo chybějícími hlavičkami jsou v tomto případě ale i další dvě hlavičky, opět nalezené nástrojem Nikto. Tyto hlavičky jsou:

- **Strict-Transport-Security**, jejíž přítomnost informuje prohlížeč, aby po stanovenou dobu probíhala veškerá komunikace výhradně přes zabezpečený HTTPS protokol.

- **Expect-CT**, která slouží jako reportující hlavička. Hlavička zajišťuje provádění kontroly webových certifikátů s *Certificate Transparency*. Případně upozorňuje, pokud se vyskytne problém s ověřením certifikátu, například pokud dojde k použití podvrženého certifikátu.

4.3 Test aplikace dle OWASP Top 10

Pro testování zranitelností na webu akce Seznamovák, respektive v aplikaci Seznamovák, byl využit seznam OWASP Top 10 2021. Detailní přehled uvedených zranitelností je k dohledání v části 1.2 této práce.

4.3.1 Broken Access Control

Špatně natavená přístupová práva jsou v aplikaci Seznamovák hned na několika místech. Prvním (a méně kritickým) místem je přístup k přehledu plateb, které monitorují pohyby na účtu Seznamováku.

Ke stránce `/platby/` by neměl mít přístup standardní uživatel, ale jen správce bankovního účtu, administrátor a vývojář. Kromě přehledu plateb lze na této stránce provádět i tzv. párování plateb. Tedy přiřadit konkrétního registrovaného účastníka a připojit k němu platbu celé akce. K této úpravě už ale uživatelé nemají oprávnění.

Druhou chybou je přístup k jakékoliv stránce, pokud její URL začíná na `/ucastnici/`. Vyjma této stránky samotné se může neregistrovaný uživatel registrovat na akci. Nebo na stránce `/participant/note/{id}` editovat informace o účastnících, aniž by bylo vyžadováno přihlášení. Kvůli této chybě dochází také ke zjištění jmen všech účastníků.

4.3.2 Cryptographic Failures

Jak bylo rozebíráno v dřívějších kapitolách této práce, aplikace Seznamovák obsahuje několik selhání, která se týkají správného nakládání s účastnickými nebo uživatelskými hesly. Účastnické klíče nejsou nijak hashovány a uživatelská hesla jsou hashována nedostatečným způsobem.

Dalším ze zjištěných kryptografických selhání je využívání zastaralých síťových protokolů TLS ve verzi 1.0 a 1.1. V TLS verze 1.2 jsou využívány šifry, které jsou dle [68] označené jako slabé a mohou být náchylné k útokům typu BEAST[25]. Mezi tyto šifry patří například:

- `TLS_RSA_WITH_AES_128_CBC_SHA`,
- `TLS_DHE_RSA_WITH_AES_256_CBC_SHA` nebo
- `TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256`.

4.3.3 Injection

Ačkoliv framework Nette implementuje ochranu proti SQL injection, stále může být aplikace tímto způsobem útoku ohrožena, zejména kvůli chybám programátorů při nesprávnému skládání dotazů do databáze. Žádné takovéto pochybení ale nebylo v průběhu testování nalezeno.

Nepodařil se ani útok typu XSS, tedy Cross-site scripting. Program Nmap sice odhalil podezřelé místo na webu akce, které by bylo možné použít k útoku typu *DOM based XSS*[26], ale při bližším zkoumání se nepodařilo najít žádný

[25] Browser Exploit Against SSL/TLS

[26] Document Object Model based XSS, také známé jako XSS typu 0

způsob, jak zranitelnost zneužít. Přesto je vhodné tento kus kódu ze stránek odstranit, protože dochází k míchání obsahu přenášeného přes HTTP a HTTPS. Kód obsahuje počítadlo návštěvnosti, to ale pozbývá významu, nikde s těmito daty není jinak nakládáno.

4.3.4 Insecure Design

Z pohledu návrhu se nepodařilo v aplikaci nalézt nic, co by ohrožovalo její bezpečnost. Jedinou výjimkou je API, na detailní zkoumání této oblasti se ale zaměřuje sekce 4.5.

Autoři webové aplikace vytvořili značné množství testů, které pokrývají všechny důležité funkcionality. Pro testy se využívá Selenium server, který spolu s nástrojem ChromeDriver zajišťují akceptační testy. Testy je možné spouštět jak jednotlivě, tak celé najednou. Výsledky testů jsou poté přístupné ve zvláštní složce spolu se snímkem obrazovky testu, který selhal.

Veškeré chybové hlášky, které v aplikaci nastávají, obsahují jen velmi zevrubný a generický popis toho, že nastala chyba. Tato chyba je doplněna o kontakt na administrátory aplikace. Nedochází tak k žádnému vyzrazení informací.

4.3.5 Security Misconfiguration

Při kontrole nastavení serveru bylo zjištěno, že informace, které server omylem sděluje v hlavičkách odpovědí, jsou pravdivé a odpovídají konfiguraci severu.

S pomocí nástroje Dirb se podařilo objevit několik informací, které mohou výrazným způsobem zvýšit útočníkovy šance o úspěšnou kompromitaci systému. Kvůli špatně zabezpečeným souborům tak může útočník zjistit mnoho informací, které jsou uvedené níže:

- Na serveru je nainstalovaný phpMyAdmin ve verzi 4.9.1. Přesnou verzi tohoto nástroje lze zjistit z nezabezpečených souborů, které jsou umístěné na adrese `/phpmyadmin/ChangeLog` a nebo `/phpmyadmin/README`.
- Na webu akce je použitý plugin Supersized, který umožňuje zobrazit fotogalerii. Plugin je už několik let neaktualizovaný a pro své fungování využívá jQuery v zastaralé verzi 2.2.1. Soubor je možné nalézt na adrese `/js/jquery.min.js`.
- Kvůli špatné konfiguraci je přístupný i README soubor z repozitáře webu, který obsahuje citlivé informace o přesné konfiguraci serveru. Nejdůležitější informace ze souboru jsou uvedeny na obrázku 4.2.
- Vzorový soubor, který je popsaný na obrázku 4.2, je také přístupný. Tentokrát pod jménem souboru `config-web.template.ini`. Ačkoliv v něm chybí uživatelské jméno a heslo, přesto lze zjistit název produkční databáze. Jedná se o název `seznamovak_app`.

```
- Naklonování repozitáře přes
  git clone git@gitlab.fit.cvut.cz:seznamovak/web.git
- Stránky jsou na serveru ve složce /var/www/html.
- Nejdůležitější je soubor index.php, kde je celá stránka.
- V /etc/seznamovak/$env/ musí být soubor config-web.ini,
  kde jsou přístupy k seznamovácké databázi.
```

Obrázek 4.2: README repozitáře webu

Jak už bylo objeveno během skenování portů, server v hlavičce vystavuje svou přesnou verzi a operační systém. Nastavení zobrazování těchto informací je vhodné vypnout v konfigurační souboru serveru umístěném na `/etc/apache2/conf-enabled/security.conf` a upravit položky:

- `ServerTokens` ze stavu `OS` na stav `Prod` a
- `ServerSignature` ze stavu `On` na stav `Off`.

Další hlavičkou, která spadá do kategorie špatné konfigurace, je přítomnost hlavičky `X-Powered-By`, kterou v odpovědi klientovi nastavuje Nette Framework. Zpráva obsahuje text „Nette Framework 3“. Útočník tak získává informaci, že pro vývoj aplikace byl použit právě tento framework a může se zaměřit na jeho zranitelnosti. V konfiguraci serveru je proto vhodné tento příznak z odpovědí smazat, například použitím `Header unset X-Powered-By`. Dle nástroje OWASP ZAP spadá nastavení této hlavičky do kategorie špatně nastavených přístupových práv [69]. Přesto spíše odpovídá zařazením do této kategorie.

Server také nenastavuje HTTP hlavičku takovým způsobem, aby obsahovala `X-Content-Type-Options`. V konfiguračním souboru serveru to lze udělat příkazem `Header add X-Content-Type-Options: "nosniff"`.

Další nenastavenou bezpečnostní hlavičkou je `X-Frame-Options`. Obranou může být doplnění `Header set X-Frame-Options: "sameorigin"` do konfiguračního souboru na serveru.

Poslední z nalezených a serverem nenastavovaných bezpečnostních hlaviček je `Content-Security-Policy`. Tato hlavička chrání proti XSS útokům a nahráváním škodlivého kódu, který útočí na uživatele. Nastavení není ale vůbec jednoduché, vyžaduje detailní znalost cílového systému. Přesto je doporučované tuto hlavičku nastavit.

4.3.6 Vulnerable and Outdated Components

Aplikace Seznamovák obsahuje několik neaktualizovaných verzí různých komponent. Toto je zřejmé už při spuštění nástroje Composer, který uživatele upozorní na chyby u dvou balíčků:

- Balíček *phpunit/php-token-stream* [70] již není aktivně vyvíjený [71]. Ačkoliv aplikace Seznamovák nemá tento balíček explicitně zmíněný v závislostech, závislost pochází z balíčku *Codeception*, který je v aplikaci Seznamovák využívaný v zastaralé verzi.
- Balíček *symfony/inflector* [72] byl ve verzi 5.1 označen jako zastaralý a následně přejmenovaný na *symfony/EnglishInflector* [73]. I tato chyba závislosti je získaná z jiných balíčků, konkrétně se jedná o *ublaboo/datagrid*.

Pomocí nástroje OWASP ZAP bylo detekováno i použití několika zastaralých verzí JavaScriptových knihoven. Na webu aplikace Seznamovák jsou tyto knihovny použity v rámci galerie fotek a v aplikaci Seznamovák v rámci rozšíření pro nahrávání souborů a filtry v databázi. Jedná se o tyto knihovny:

- Bootstrap verze 3.3.7,
- JQuery-ui verze 1.10.4 a 1.12.1,
- JQuery verze 1.10.4 a 3.2.1.

Nalezené zranitelnosti tak teoreticky umožňují Cross-site Scripting (CVE-2015-9251, CVE-2018-14041, CVE-2019-8331) nebo spuštění kódu, který nepochází z důvěryhodného zdroje (CVE-2021-41182, CVE-2021-41184).

K hledání nových zranitelností a aktualizaci komponent dochází jen velmi sporadicky. Ačkoliv je webová aplikace aktualizována častěji než samotný web, pořád se jedná nejméně o interval několika měsíců.

Díky testům, které jsou součástí webové aplikace a existenci testovací aplikace, dochází při aktualizaci verzí a jiných komponent k otestování všech důležitých funkcionalit aplikace ještě před tím, než je aktualizace provedena na produkční verzi aplikace. V tomto bodě je nastavená politika v souladu s doporučením pro zranitelnosti v této kategorii.

4.3.7 Identification and Authentication Failures

Pochybení v této oblasti byla už také objevena a diskutována v předchozí části textu. Kvůli špatně nastaveným politikám a využíváním knihovních funkcí, které útočníka nijak nezpomalují, je možné provádět útoky hrubou silou, časováním a mnoho jiných.

Aplikace umožňuje být přihlášený na více zařízeních současně. Přesněji řečeno, session ID na obou přihlášených zařízeních zůstává validní a pro oba přihlášené účty je možné vykonávat akce. Session se stane nevalidní pouze po vypršení časového limitu `Max-Age`, nebo po manuálním odhlášení z aplikace. Časový limit je nastavený na 1209600 sekund, což odpovídá 14 dnům. Tato hodnota je zbytečně vysoká, správně by měla být mnohem nižší.

Možnost být přihlášen na více zařízeních současně nemusí ale znamenat, že se jedná o bezpečnostní pochybení, záleží na vývojářích a celkové politice aplikace. Přesto je důležité tento bod zmínit a zvážit, zda současné nastavení odpovídá původnímu záměru.

4.3.8 Software and Data Integrity Failures

Balíčky a závislosti použité ve webové aplikaci jsou stahovány přes nástroj Composer. V této části nebylo objeveno žádné pochybení.

Na webových stránkách ale přesto byly nalezeny pochybení, které se týkají ověřování integrity stahovaných dat. Ve fotogalerii na adrese `https://seznamovak.fit.cvut.cz/galerie/index.html` se nachází skript

```
<script type="text/javascript" src="{odkaz}"></script>
```

který při kompromitaci stránky `{odkaz}`, respektive souboru na ní uložené, může vést k XSS. Obdobná chyba se vyskytuje na všech stránkách, ve kterých se používá TinyMCE WYSIWYG[27] editor.

Serializace ani deserializace není žádným způsobem v aplikaci využívána. Z těchto důvodů není tato zranitelnost pro webovou aplikaci využitelná.

4.3.9 Security Logging and Monitoring Failures

Aplikace Seznamovák provádí logování několika důležitých akcí, mezi něž patří například:

- úspěšné přihlášení organizátora,
- odeslání e-mailu účastníkům,
- odeslání push notifikací.

Zároveň je ale dlouhodobým problémem absence logování jiných důležitých akcí. Těmi může být například editace informací o účastnících nebo jejich úplné smazání z databáze. Chybějící ukládání záznamů o provedených akcích ztěžuje (a takřka znemožňuje) zpětnou rekonstrukci provedených úprav.

Logy je možné po přihlášení zobrazit na adrese `/logs/`. Přístupové údaje jsou umístěny v dokumentaci v GitLabovém repozitáři. Zobrazit si je tak teoreticky může jen oprávněná osoba.

V průběhu provádění penetračních testů byly některé pokusy o přístup zaznamenány a byl o nich pořízen záznam. Aplikace Seznamovák ale nedisponuje aktivním sledováním záznamů, proto by byly pokusy o napadení systému zjištěny až s odstupem času.

[27] What You See Is What You Get

4.3.10 Server-Side Request Forgery

Útok na tuto zranitelnost vyžaduje, aby se neprováděla validace URL, kterou zadává uživatel. Na straně aplikace Seznamovák to provádí framework Nette. Během testování se nepodařilo najít žádnou stránku, kde by tato validace byla opomenuta, nebo kde by byl úspěšným způsobem proveden útok.

4.4 Výsledek testu OWASP Top 10

Nalezené zranitelnosti jsou ohodnoceny podle metodiky CVSS (Common Vulnerability Scoring System) v aktuální verzi 3.1. Použití tohoto přístupu umožňuje číselné ohodnocení nalezených zranitelností, kde nejnižší míra závažnosti nabývá hodnoty 0 a nejvyšší závažnost je ohodnocena číslem 10. K číselné hodnotě existuje i slovní alternativa, na kterou je možné výsledné číslo převést podle tabulky 4.1.

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

Tabulka 4.1: Převedení CVSS skóre na slovní hodnocení

Kromě číselného (resp. slovního) ohodnocení je možné popsat zranitelnost i s využitím tzv. *vector stringu*. Tedy řetězce, který umožňuje nalezenou zranitelnost klasifikovat s využitím krátkého textu. Jednotlivé atributy, které umožňují provést klasifikaci zranitelností, jsou například složitost útoku, jeho rozsah nebo dopady na dostupnost dat. Přesná specifikace je dostupná na [74].

Pro výpočet CVSS skóre byla použita interaktivní kalkulačka, která je dostupná na webu FIRST[28], neboli organizace, která spravuje CVSS [75]. Nalezené zranitelnosti jsou uvedeny v tabulce 4.2. Jednotlivé vygenerované *vector stringy* k nalezeným zranitelnostem je možné nalézt v příloze C.

4.5 Test API dle OWASP API Security Top 10

Pro testování zranitelností API a jejích endpointů, které nabízí webová aplikace Seznamovák, byl využit seznam OWASP API Security Top 10 2019. Detailní popis jednotlivých zranitelností je opět uveden v sekci 1.3 této práce.

[28] Forum of Incident Response and Security Teams

Název zranitelnosti	Závažnost
Použití zastaralých síťových protokolů	Medium (6.5)
Slabé parametry pro hashovací algoritmus	Medium (5.9)
Hashování s nedostatečnou entropií	Medium (5.9)
Slabé požadavky na heslo	Medium (5.9)
Stahování souborů bez kontroly integrity	Medium (5.9)
Používání zranitelných JS knihoven	Medium (5.9)
Používání neudržovaných komponent	Medium (5.9)
Citlivé informace v přístupném souboru	Medium (5.3)
Slabé šifry při síťové komunikaci	Low (3.7)
Zveřejnění informací v HTTP hlavičce	Low (3.1)
Chybějící hlavička X-Content-Type-Options	Low (3.1)
Chybějící hlavička X-Frame-Option	Low (3.1)
Chybějící hlavička Content-Security-Policy	Low (3.1)
Chybějící hlavička Expect-CT	Low (3.1)

Tabulka 4.2: Shrnutí nalezených zranitelností ve webové aplikaci

4.5.1 Broken Object Level Authorization

Proces autorizace není v API aplikace Seznamovák nijak implementován. Útočník si tak může na endpointu **points-json**, v případě zadání správného ID, zobrazit počet bodů, které získal jiný účastník. Stejnou informaci může ale získat i v offline podobě na nástěnce, kde jsou k vidění body všech účastníků, včetně všech jmen a příjmení, což API nenabízí. Nejedná se proto o hrozbu úniku soukromých údajů.

4.5.2 Broken User Authentication

Pro přístup ke všem endpointům je použit stejný způsob autentizace, kterým je zadání speciálního přístupového tokenu do URL požadavku. Pokud s API komunikují například mobilní aplikace, vede nutně tento způsob autentizace k uložení přístupového hesla (tokenu) do zdrojových kódů aplikace. Pro útočníka poté nemusí být náročné toto heslo získat s využitím například zpětné analýzy aplikace. Proto je tento způsob ochrany naprosto nedostatečný.

4.5.3 Excessive Data Exposure

Všechny informace, které si uživatel ve svém dotazu na endpoint vyžádá, obsahují data, které poté nejsou žádným způsobem na straně uživatele filtrovány. Žádné pochybení v této kategorii nebylo nalezeno.

4.5.4 Lack of Resources & Rate Limiting

V API aplikace Seznamovák nebyly nalezeny špatně nastavené limity, které by způsobovaly nedostupnost služby.

Jediné, potenciálně zranitelné místo, je endpoint **songs-json**, který odesílá poměrně velké množství informací. Zatímco ostatní endpointy odesílají informace v jednotkách bajtů, tento koncový bod odesílá v testovací verzi aplikace 130 kB na jeden požadavek a v produkční verzi tohoto endpointu se jedná dokonce o 308 kB na jeden požadavek. Proto by mohl být tento endpoint prvním, na který se útočník zaměří, pokud by chtěl službu zahltit a způsobit tak její nedostupnost.

4.5.5 Broken Function Level Authorization

Přístup k API není omezený na základě rolí. Pokud zná uživatel endpoint a přístupový tokem, má přístup k informacím, které jsou na daném endpointu dostupné. V této kategorii nebylo nalezeno žádné pochybení.

4.5.6 Mass Assignment

Ani v této kategorii nebylo nalezeno žádné pochybení. Většina endpointů slouží pouze k zobrazení a jakákoliv vyzkoušená modifikace HTTP požadavku nevedla k narušení zamýšleného chování.

4.5.7 Security Misconfiguration

Pochybení, které bylo v této kategorii nalezeno, se týká dvojitého nastavování hlavičky `Set-Cookie`. Poprvé je hlavička nastavována standardními mechanismy, podruhé je přidávána s pomocí frameworku Nette. Dochází nejen k duplikaci odesílaných informací, ale i pravidel, kdy každé z nich slouží k trochu jiným účelům.

V případě API endpointu **songs-json** se dokonce jedná o nastavování tří různých hlaviček. Framework Nette dokonce nastavuje v rámci odpovědi proměnnou `PHPSESSID` na dvě různé hodnoty. Tento případ je zachycený i na obrázku 4.3.

Samotné nastavování hlavičky `Set-Cookie` je z hlediska současného používání endpointů značně redundantní, k nastavování hlaviček není žádný důvod.

4.5.8 Injection

Všechny endpointy v aplikaci slouží k zobrazení určitých dat. Tyto informace jsou generovány z databáze za pomoci frameworku Nette, které provádí validaci dotazů. Dotazy jsou navíc konstruovány způsobem, který se snaží zamezit jakémukoliv vsunutí škodlivého kódu. Pochybení v této kategorii se nepodařilo nalézt.

```
Set-Cookie: _nss=1; path=/; secure; HttpOnly; SameSite=Strict
Set-Cookie: PHPSESSID={SESSION}; expires={DATE};
    Max-Age=1209600; path=/; secure; HttpOnly; SameSite=Lax
Set-Cookie: PHPSESSID={SESSION2}; expires={DATE};
    Max-Age=1209600; path=/; secure; HttpOnly; SameSite=Lax
```

Obrázek 4.3: Vícenásobné nastavování hlaviček

4.5.9 Improper Assets Management

Endpointy v aplikaci Seznamovák slouží pouze pro potřeby interních mobilních aplikací. Proto je seznam těchto koncových bodů, včetně dokumentace, umístěn v repozitáři projektu, kam mají přístup pouze organizátoři. Téměř všechny endpointy jsou zdokumentované a v současné době využívané. Jedinou výjimku tvoří endpoint **get-participant-id-for-email-json**, který byl nahrazen novou verzí, která navíc obsahuje i relevantní informace pro hru Kolíčkovaná. Ačkoliv zastaralá verze endpointu obsahuje méně informací, než jeho nová verze, přesto by měla být využívána jen jedna z nich.

4.5.10 Insufficient Logging & Monitoring

Ačkoliv dochází k monitorování jednotlivých přístupů k API, záznamy jsou uloženy ve stejném souboru, ve kterém jsou uloženy přístupy i do ostatních částí aplikace. Dochází tak k míchání logovaného obsahu, což může ztěžovat následnou analýzu. Jak je vidět na obrázku 4.4, jednotlivé záznamy obsahují velmi povrchní popis chyby, ke které došlo. Obsah samotných logů tak může být nedostatečně detailní pro následné zkoumání.

```
[2022-03-09 22-01-16] HTTP code 403: Unauthorized access in
    APIPresenter.php:162 @ /aplikace/api/
[2022-04-11 16-15-47] HTTP code 400: Unknown game in
    APIPresenter.php:180 @ /aplikace/api/teams/1/1
```

Obrázek 4.4: Fragment záznamů o přístupu

4.6 Výsledek testu OWASP API Security Top 10

API aplikace Seznamovák se spoléhá na autentizaci pomocí fixního hesla, které je uloženo v mobilních aplikacích. Heslo je využito pro veškerou komunikaci, kterou odesílá uživatel a umožňuje provádět komunikaci s jakýmkoliv endpointem. Z tohoto důvodu má podle metriky CVSS hodnocení 7.5,

což odpovídá klasifikaci High. Přiřazený *vector string* poté odpovídá řetězci CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N.

Dalšími nalezenými zranitelnostmi je vícenásobné nastavování hlavičky `Set-Cookie` a nedostatečné logování. Tyto zranitelnosti jsou zde uvedeny hlavně z informativních důvodů, nepodařilo se objevit žádný způsob, jak je využít pro útok.

Kapitola 5

Útok na hesla

Tato kapitola se zaměřuje na scénář, při kterém by se útočníkovi podařilo získat hashe hesel z databáze akce Seznamovák nebo Magistrovák. Mohl by se tak tímto způsobem dostat k informacím o účastnících.

Většina organizátorů Seznamováku později organizuje i akci Magistrovák, proto jsou jejich uživatelské účty zkopírovány do databáze Magistrováku. Tím dochází k přenosu starého hesla do dalšího systému. Zhruba polovina organizátorů si toto heslo později však už nikdy nezmění. Dochází tak k opakování již použitého hesla, a v případě jeho prolomení i k úniku osobních informací účastníků z další akce.

5.1 Slovníkový útok

Ve slovníkovém útoku zkouší útočník hesla z předem připraveného seznamu (tj. slovníku). Většinou se jedná o efektivnější metodu, než zkoušet všechny kombinace, které je možné vytvořit. Metoda je účinnější zejména z důvodu nezodpovědného přístupu uživatelů, kteří za hesla volí slova, která nejsou ničím unikátní.

Mnoho již dříve uniklých hesel bylo později zkombinováno do slovníků, které čítají miliony hesel uložené v čitelné podobě. Jedním z populárních slovníků je například `rockyou.txt`, který obsahuje přes 14 milionů záznamů. Ačkoliv vyzkoušení těchto hesel nezaručuje, že dojde k prolomení hesel, podstatně zvyšují šanci na úspěch, zejména právě díky přístupu uživatelů.

Pro útok na hashe hesel byl používán nástroj Hashcat [76]. Ten umožňuje pro výpočet hashů využít procesor nebo grafickou kartu. Grafická karta umožňuje paralelizovat výpočty, což přispívá k zásadnímu snížení času, který je potřebný pro samotný útok. Jak je vidět na obrázku 5.1, pro vyzkoušení všech hesel z výše zmíněného slovníku `rockyou.txt` bylo na grafické kartě `NVIDIA GeForce RTX 2070` potřeba něco přes 26 hodin čistého času. Za tento čas se podařilo prolomit šest uživatelských hesel.

```
Session.........: hashcat
Status..........: Exhausted
Hash.Mode.......: 3200 (bcrypt $2*$, Blowfish (Unix))
Hash.Target.....: passwords.txt
Time.Started.....: Thu Mar 03 09:24:17 2022 (1 day, 3 hours)
Time.Estimated...: Fri Mar 04 12:52:43 2022 (0 secs)
Guess.Base.......: File (rockyou.txt)
Speed.#1.........: 6388 H/s (2.25ms)
Recovered........: 6/47 (12.77%) Digests
Progress.........: 674186048/674186048 (100.00%)
Candidates.#1....: pepe -> babygurl69
Hardware.Mon.#1..: Temp: 59c Fan: 68% Util: 83% Bus:16

Started: Thu Mar 03 09:24:10 2022
Stopped: Fri Mar 04 12:52:44 2022
```

Obrázek 5.1: Průběh programu Hashcat

5.2 Slovníkový útok s pravidly

Program Hashcat umožňuje kromě obyčejného slovníkového útoku provádět i slovníkový útok s pravidly. Ten spočívá v aplikaci pravidel, která rozšiřují standardní slovníkový útok. Jelikož bývají uživatelé prakticky ze všech stran napomínáni, co vše má jejich heslo splňovat, může se stát, že na základě tohoto přístupu pak tvoří hesla stejným způsobem i v dalších službách.

Tradiční pravidla pro tvorbu hesla, ačkoliv odporují současným doporučením pro tvorbu hesel (například dle [52]), bývají:

- alespoň 8 znaků,
- velké písmeno,
- malé písmeno,
- speciální znak a
- číslo.

Této znalosti může útočník využít a například slovník `rockyou.txt` obohatit o častá pravidla u hesel. Lze využít i předem připravené seznamy pravidel, program Hashcat například obsahuje i soubor `best64.rule` [77]. Ten obsahuje 64 nejčastějších pravidel u tvoření hesla, mezi něž patří například:

- `$[0-9]`, které přidá na konec hesla číslici,
- `so0`, které nahradí písmeno `o` číslicí `0`,
- `r`, které heslo napíše pozpátku, nebo
- `c`, které zvětší první písmeno v heslu.

Po více než 30 hodinách čistého času a vyzkoušení všech možných modifikací tohoto slovníku bohužel nedošlo k prolomení žádného dalšího hesla.

Může to být způsobeno tím, že uživatelé tímto způsobem hesla netvoří. Samozřejmě je tu i možnost, že použitá pravidla nebyla dostatečně robustní, aby byla schopna odhalit pravidla použitá uživateli.

5.3 Útok hrubou silou

Posledním pokusem pro prolomení hesel je použít hrubou sílu a vyzkoušet tak všechny možné kombinace pro hesla různých délek. Nejdříve pro 1 znak, poté pro 2 znaky a tak dále.

S každým dalším znakem v heslu roste složitost exponenciálně. Pokud uživatel nevolí jako své heslo pouze malá písmena abecedy, ale své heslo obohatí o alfanumerické znaky, popřípadě o speciální znaky (například o závorku nebo zavináč), roste prostor možných klíčů stále exponenciálně, ale ještě mnohem rychleji.

Vyzkoušet všechna možná hesla pro víceznakové heslo poté trvá dny, roky, nebo ještě delší dobu. Samozřejmě je možné útok urychlit s použitím výkonnější grafické karty nebo použít karet více, tedy prolamování ještě více paralelizovat. V tabulce 5.1 si lze všimnout, že s každou další generací grafických karet se obvykle znásobí počet hashů, které můžeme během jedné sekundy vypočítat.

V útoku hrubou silou byly vyzkoušeny všechny možné kombinace pro hesla do délky pěti znaků včetně. Všechna hesla obsahují alespoň šest znaků, protože při tomto útoku nebylo prolomeno ani jedno heslo. Pokračovat ve zkoušení dalších znaků je v domácím prostředí takřka nemožné.

Grafická karta	Počet H/s
NVIDIA RTX 2070	6 915
NVIDIA RTX 2080 Ti	13 315
NVIDIA RTX 3090	26 450
NVIDIA A100 SXM4	34 975

Tabulka 5.1: Počet vyzkoušených hashů v závislosti na grafické kartě [78]

5.4 Zhodnocení útoku na hashe

Během všech útoků se podařilo celkem prolomit šest uživatelských hesel, což tvoří téměř 13 % všech hesel. Prolomenými hesly jsou:

- marketka
- linkin
- testuser
- tajneheslo
- hibernia
- lokomotiva

Většina uživatelských účtů, ke kterým byly zjištěny přihlašovací údaje, již naštěstí není aktivních, tedy při procesu přihlašování už není uživatel vyhodnocený jako povolený a přihlášení selže.

Uživatel s heslem „lokomotiva“ je aktivním organizátorem Seznamováku a zároveň i akce Magistrovák. Současně v obou aplikacích používá stejné heslo.

Umělý testovací uživatel „testuser“ používá stejné heslo jako má uživatelské jméno. Ačkoliv se jedná pouze o účet, oproti kterému se spouštějí testy, přesto by měl mít složitější heslo. Tento uživatelský účet má totiž přístup ke všem uloženým informacím o účastnících akce. Uživatelský účet zároveň ale není povolený v produkční verzi aplikace.

V testovací verzi aplikace ale existuje i testovací uživatel, který vlastní administrátorská oprávnění a využívá stejné heslo jako „testuser“. A protože může administrátor provádět zálohu databáze (včetně jejího plného stažení) a nedochází k modifikaci hesel ostatních organizátorů, může útočník směřovat energii a výpočetní čas k prolomení hesel některého z aktivních organizátorů.

Jak již bylo uvedeno v tabulce 5.1, pokud jsou k dispozici výkonnější grafické karty, daří se útočníkovi podstatným způsobem zkrátit dobu potřebnou pro prolomení hesel.

Alternativně si útočník může pronajmout výpočetní čas na některém z nejvýkonnějších cloudů. Například Amazon [79] (v rámci své služby AWS[29] [80]) nabízí osm grafických karet, které za hodinu vyzkouší až 4 miliardy hashů pro algoritmus Bcrypt s parametrem *cost* `$07$` [81, 82].

[29] Amazon Web Services

Závěr

Cílem této diplomové práce bylo provést analýzu bezpečnosti webové aplikace Seznamovák a seznámit čtenáře se zranitelnostmi webových aplikací.

První kapitola diplomové práce se proto zaměřovala na organizaci OWASP, která se zabývá zvyšováním povědomí o těchto hrozbách. Jsou zde také představeny seznamy OWASP Top 10 a OWASP API Security Top 10, které popisují nejčastější zranitelnosti webových aplikací a API.

Ve druhé kapitole je čtenář seznámen s akcí Seznamovák a zároveň s komplexní webovou aplikací Seznamovák, kterou organizátoři akce využívají jako hlavní technologický prvek při organizaci, na který je navázáno i několik dalších aplikací určené pro mobilní telefony.

Třetí kapitola práce představovala statickou analýzu kódu aplikace, při které byl kladen důraz na bezpečnost použitých hesel, uložených osobních údajů účastníků a organizátorů. V této kapitole se také zaměřovalo na API, které webová aplikace Seznamovák poskytuje. Všechna kritická místa jsou opatřena komentářem a návrhem oprav, které vedou k vyřešení nalezených zranitelností.

Mezi největší nalezená pochybení patří například nesprávné použití algoritmu Bcrypt, který se používá pro hashování uživatelských hesel. Kvůli tomuto pochybení je podstatně snížena bezpečnost uložených hesel. Dalším nalezeným pochybením je absence hashování účastnických klíčů, které jsou proto náchylné vůči útoku hrubou silou.

Čtvrtá kapitola této práce obsahuje už samotné penetrační testování. Poté je provedeno testování aplikace a poskytovaného API podle OWASP Top 10.

Mezi největší nalezená pochybení v této kapitole patří použití zastaralých sítových protokolů a šifer, dále také špatně nastavené bezpečnostní hlavičky, které například prozrazují informace o konfiguraci serveru a umožňují provádět různé typy útoků. Dalším objeveným pochybením je také přítomnost souborů, které jsou volně přístupné a obsahují citlivé informace.

V páté kapitole je kladen důraz na prolomení otisků hesel, které jsou uloženy v databázi aplikace. Během útoku na tyto otisky bylo vyzkoušeno několik různých přístupů a v případě slovníkové metody bylo prolomeno šest hesel k uživatelským účtům.

Všechny objevené chyby byly diskutovány s autory aplikace. V době odevzdání této práce byly dokončeny opravy nejvíce kritických chyb, které byly v průběhu analýzy nalezeny. V brzké době by mělo dojít ke zveřejnění opravené verze aplikace Seznamovák. Zranitelnosti nalezené ve špatné konfiguraci serveru již byly opraveny.

Bibliografie

1. WEBSITESETUP. *How Many Websites Are There?* [Online]. 2021 [cit. 2022-04-19]. Dostupné z: https://websitesetup.org/news/how-many-websites-are-there/.
2. OWASP FOUNDATION. About the OWASP Foundation. *OWASP* [online]. 2021 [cit. 2022-01-31]. Dostupné z: https://owasp.org/about/.
3. OWASP FOUNDATION. How to use the OWASP Top 10 as a standard. *OWASP* [online]. 2021 [cit. 2022-01-31]. Dostupné z: https://owasp.org/Top10/A00_2021_How_to_use_the_OWASP_Top_10_as_a_standard/.
4. OWASP FOUNDATION. *OWASP Top Ten* [online]. 2021 [cit. 2022-04-23]. Dostupné z: https://owasp.org/www-project-top-ten/.
5. OWASP FOUNDATION. *Introduction: Thank you to our data contributors* [online]. 2021 [cit. 2022-03-02]. Dostupné z: https://owasp.org/Top10/A00_2021_Introduction/.
6. OWASP FOUNDATION. *Top 10 Web Application Security Risks* [online]. 2021 [cit. 2022-04-20]. Dostupné z: https://owasp.org/www-project-top-ten/assets/images/mapping.png.
7. OWASP FOUNDATION. *Top10: A01:2021: Broken Access Control* [online]. 2021 [cit. 2022-03-02]. Dostupné z: https://owasp.org/Top10/A01_2021-Broken_Access_Control/.
8. OWASP FOUNDATION. *Top10: A02:2021: Cryptographic Failures* [online]. 2021 [cit. 2022-03-02]. Dostupné z: https://owasp.org/Top10/A02_2021-Cryptographic_Failures/.
9. OWASP FOUNDATION. *Top10: A03:2021: Injection* [online]. 2021 [cit. 2022-03-02]. Dostupné z: https://owasp.org/Top10/A03_2021-Injection/.

10. OWASP FOUNDATION. *Top10: A04:2021: Insecure Design* [online]. 2021 [cit. 2022-03-02]. Dostupné z: https://owasp.org/Top10/A04_2021-Insecure_Design/.

11. OWASP FOUNDATION. *Top10: A05:2021: Security Misconfiguration* [online]. 2021 [cit. 2022-03-02]. Dostupné z: https://owasp.org/Top10/A05_2021-Security_Misconfiguration/.

12. OWASP FOUNDATION. *Top10: A06:2021: Vulnerable and Outdated Components: Description* [online]. 2021 [cit. 2022-03-02]. Dostupné z: https://owasp.org/Top10/A06_2021-Vulnerable_and_Outdated_Components/.

13. OWASP FOUNDATION. *Top10: A07:2021: Identification and Authentication Failures* [online]. 2021 [cit. 2022-03-02]. Dostupné z: https://owasp.org/Top10/A07_2021-Identification_and_Authentication_Failures/.

14. OWASP FOUNDATION. *Top10: A08:2021: Software and Data Integrity Failures* [online]. 2021 [cit. 2022-03-02]. Dostupné z: https://owasp.org/Top10/A08_2021-Software_and_Data_Integrity_Failures/.

15. OWASP FOUNDATION. *OWASP Top 10 - 2017: The Ten Most Critical Web Application Security Risks* [online]. 2017 [cit. 2022-03-02]. Dostupné z: https://github.com/OWASP/Top10/blob/master/2017/OWASP%5C%20Top%5C%2010-2017%5C%20(en).pdf.

16. OWASP FOUNDATION. *Top10: A09:2021: Security Logging and Monitoring Failures* [online]. 2021 [cit. 2022-03-02]. Dostupné z: https://owasp.org/Top10/A09_2021-Security_Logging_and_Monitoring_Failures/.

17. OWASP FOUNDATION. *Top10: A10:2021: Server-Side Request Forgery (SSRF)* [online]. 2021 [cit. 2022-03-02]. Dostupné z: https://owasp.org/Top10/A10_2021-Server-Side_Request_Forgery_%5C%28SSRF%5C%29/.

18. OWASP FOUNDATION. *OWASP API Security Project* [online]. 2018 [cit. 2022-04-23]. Dostupné z: https://owasp.org/www-project-api-security/.

19. OWASP FOUNDATION. *API1:2019: Broken Object Level Authorization* [online]. 2019 [cit. 2022-03-02]. Dostupné z: https://github.com/OWASP/API-Security/blob/master/2019/en/src/0xa1-broken-object-level-authorization.md.

20. OWASP FOUNDATION. *API2:2019: Broken User Authentication* [online]. 2019 [cit. 2022-03-02]. Dostupné z: https://github.com/OWASP/API-Security/blob/master/2019/en/src/0xa2-broken-user-authentication.md.

21. OWASP FOUNDATION. *API3:2019: Excessive Data Exposure* [online]. 2019 [cit. 2022-03-02]. Dostupné z: https://github.com/OWASP/API-Security/blob/master/2019/en/src/0xa3-excessive-data-exposure.md.

22. OWASP FOUNDATION. *API4:2019: Lack of Resources & Rate Limiting* [online]. 2019 [cit. 2022-03-02]. Dostupné z: https://github.com/OWASP/API-Security/blob/master/2019/en/src/0xa4-lack-of-resources-and-rate-limiting.md.

23. OWASP FOUNDATION. *API5:2019: Broken Function Level Authorization* [online]. 2019 [cit. 2022-03-02]. Dostupné z: https://github.com/OWASP/API-Security/blob/master/2019/en/src/0xa5-broken-function-level-authorization.md.

24. OWASP FOUNDATION. *API6:2019: Mass Assignment* [online]. 2019 [cit. 2022-03-02]. Dostupné z: https://github.com/OWASP/API-Security/blob/master/2019/en/src/0xa6-mass-assignment.md.

25. OWASP FOUNDATION. *API7:2019: Security Misconfiguration: Is the API Vulnerable?* [Online]. 2019 [cit. 2022-03-02]. Dostupné z: https://github.com/OWASP/API-Security/blob/master/2019/en/src/0xa7-security-misconfiguration.md.

26. OWASP FOUNDATION. *API8:2019: Injection* [online]. 2019 [cit. 2022-03-02]. Dostupné z: https://github.com/OWASP/API-Security/blob/master/2019/en/src/0xa8-injection.md.

27. OWASP FOUNDATION. *API9:2019: Improper Assets Management* [online]. 2019 [cit. 2022-03-02]. Dostupné z: https://github.com/OWASP/API-Security/blob/master/2019/en/src/0xa9-improper-assets-management.md.

28. OWASP FOUNDATION. *API10:2019: Insufficient Logging & Monitoring: Is the API Vulnerable?* [Online]. 2019 [cit. 2022-03-02]. Dostupné z: https://github.com/OWASP/API-Security/blob/master/2019/en/src/0xaa-insufficient-logging-monitoring.md.

29. *Seznamovák FIT ČVUT* [online]. 2012 [cit. 2022-04-21]. Dostupné z: https://seznamovak.fit.cvut.cz/.

30. JANOCHOVÁ, Markéta. *Webová aplikace pro Seznamovák FITu*. Praha, 2013. Dostupné také z: https://dspace.cvut.cz/handle/10467/19303. Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií. Vedoucí práce Helena Wallenfelsová.

31. VLČEK, Matěj. Seznamovák FIT 2021. *Buď FIT* [online]. 2021 [cit. 2022-04-23]. Dostupné z: https://casopis.fit.cvut.cz/seznamovak/seznamovak-fit-2021/.

32. *Magistrovák FIT ČVUT* [online]. 2019 [cit. 2022-04-21]. Dostupné z: https://magistrovak.fit.cvut.cz/.

33. PHP GROUP. *PHP. Version 8.1.3* [soft.]. 2021 [cit. 2022-02-22]. Dostupné z: https://www.php.net/.

34. NETTE FOUNDATION. *Nette Application. Version 4.0* [soft.]. 2021 [cit. 2022-02-13]. Dostupné z: https://nette.org/.

35. MARIADB FOUNDATION. *MariaDB server: The open source relational database. Version 10.7.3* [soft.]. 2022 [cit. 2022-02-22]. Dostupné z: https://mariadb.com/.

36. NETTE FOUNDATION. *Tracy. Version 3.0* [soft.]. 2022 [cit. 2022-02-22]. Dostupné z: https://tracy.nette.org/.

37. KOĎOUSKOVÁ, Barbora. *Architektura MVC: Definice, struktura, frameworky* [online]. 2021 [cit. 2022-03-02]. Dostupné z: https://www.rascasone.com/cs/blog/architektura-mvc-struktura-frameworky.

38. W3TECHS. *Historical yearly trends in the usage statistics of server-side programming languages for websites* [online]. 2009 [cit. 2022-02-22]. Dostupné z: https://w3techs.com/technologies/history_overview/programming_language/ms/y.

39. *PHP is insecure by default* [online]. 2016 [cit. 2022-02-22]. Dostupné z: https://learnwebtutorials.com/php-is-insecure-by-default.

40. NETTE FOUNDATION. *Nette Framework: Rychlý a pohodlný vývoj webových aplikací v PHP* [online]. 2021 [cit. 2022-02-15]. Dostupné z: https://nette.org/cs/#toc-features.

41. SKVORC, Bruno. *The Best PHP Framework for 2015: SitePoint Survey Results* [online]. 2015 [cit. 2022-02-15]. Dostupné z: https://www.sitepoint.com/best-php-framework-2015-sitepoint-survey-results/.

42. ORACLE CORPORATION. *MySQL database. Version 8.0.28* [soft.]. 2022 [cit. 2022-03-21]. Dostupné z: https://www.mysql.com/.

43. FREE SOFTWARE FOUNDATION. *GNU General Public License* [online] [cit. 2022-03-02]. Dostupné z: https://www.gnu.org/licenses/gpl-3.0.en.html.

44. ADERMANN, Nils; BOGGIANO, Jordi. *Composer. Version 2.3.4: A Dependency Manager for PHP* [soft.]. 2022 [cit. 2022-04-12]. Dostupné z: https://getcomposer.org/.

45. THE APACHE SOFTWARE FOUNDATION. *Apache HTTP Server. Version 2.4.53* [soft.]. 2022 [cit. 2022-03-21]. Dostupné z: https://httpd.apache.org/.

46. NETCRAFT LTD. *February 2022 Web Server Survey* [online]. 2022 [cit. 2022-03-21]. Dostupné z: https://news.netcraft.com/archives/category/web-server-survey/.

47. KUČEROVÁ, Michaela. *Aplikace Seznamovák Android. Version 2.0.0* [soft.]. 2022 [cit. 2022-04-12]. Dostupné z: `https://play.google.com/store/apps/details?id=cz.fit.seznamovak.mobile`.

48. KUČEROVÁ, Michaela. *Mobilní aplikace Seznamovák*. Praha, 2020. Dostupné také z: `https://dspace.cvut.cz/handle/10467/92864`. Bakalářská práce. České vysoké učení technické v Praze, Fakulta informačních technologií. Vedoucí práce Tomáš Nováček.

49. SEDLÁK, Daniel. *Aplikace Seznamovák iOS. Version 1.0* [soft.]. 2022 [cit. 2022-04-12]. Dostupné z: `https://apps.apple.com/us/app/id1475351883`.

50. KODR, Marek. *Aplikace Zpěvník. Version 1.0* [soft.]. 2022 [cit. 2022-04-12]. Dostupné z: `https://play.google.com/store/apps/details?id=cz.fit.seznamovak.zpevnik`.

51. NETTE FOUNDATION. *Random: Secure random string generator* [online] [cit. 2022-02-28]. Dostupné z: `https://api.nette.org/utils/master/Nette/Utils/Random.html`.

52. NIST. *Digital Identity Guidelines: Now Available* [online]. 2017 [cit. 2022-03-02]. Dostupné z: `https://pages.nist.gov/800-63-3/`.

53. PHP GROUP. *password_verify: Verifies that a password matches a hash* [online] [cit. 2022-02-28]. Dostupné z: `https://www.php.net/manual/en/function.password-verify.php`.

54. PHP GROUP. *hash_equals: Timing attack safe string comparison* [online] [cit. 2022-02-28]. Dostupné z: `https://www.php.net/manual/en/function.hash-equals.php`.

55. DIMON, Garrett. *Password reset email best practices* [online] [cit. 2022-04-19]. Dostupné z: `https://postmarkapp.com/guides/password-reset-email-best-practices`.

56. PHP GROUP. *crypt: One-way string hashing* [online] [cit. 2022-02-28]. Dostupné z: `https://www.php.net/manual/en/function.crypt.php`.

57. PHP GROUP. *CRYPT_BLOWFISH security fix details* [online]. 2011 [cit. 2022-02-28]. Dostupné z: `https://www.php.net/security/crypt_blowfish.php`.

58. PHP GROUP. *password_hash: Creates a password hash* [online] [cit. 2022-02-28]. Dostupné z: `https://www.php.net/manual/en/function.password-hash.php`.

59. LYON, Gordon. *Nmap Security Scanner. Version 7.92* [soft.]. 1997 [cit. 2022-03-21]. Dostupné z: `https://nmap.org/`.

60. LYON, Gordon. *Nmap: Usage and Examples* [online] [cit. 2022-03-02]. Dostupné z: `https://nmap.org/book/nse-usage.html`.

61. SULLO, Chris. *Nikto. Version 2.1.6* [online]. 2012 [cit. 2022-03-02]. Dostupné z: `https://github.com/sullo/nikto`. GitHubový repozitář projektu Nikto.

62. SULLO, Chris. *Nikto Wiki: Overview & Description* [online]. 2012 [cit. 2022-03-02]. Dostupné z: `https://github.com/sullo/nikto/wiki/Overview-&-Description`. GitHubový repozitář projektu Nikto.

63. PINUAGA, Ramon. *Dirb. Version 2.22* [online]. 2015 [cit. 2022-03-02]. Dostupné z: `https://salsa.debian.org/pkg-security-team/dirb`. GitLabový repozitář projektu Dirb.

64. HERTZOG, Raphaël. *About Nikto* [online]. 2015 [cit. 2022-03-02]. Dostupné z: `https://salsa.debian.org/pkg-security-team/dirb/-/blob/debian/master/README.txt`. GitLabový repozitář projektu Dirb.

65. RAPID7 LLC. *Metasploit Framework. Version 6.1.34* [online]. 2013 [cit. 2022-03-02]. Dostupné z: `https://www.metasploit.com/`.

66. OFFENSIVE SECURITY. *Metasploit Unleashed: Free Ethical Hacking Course* [online] [cit. 2022-03-02]. Dostupné z: `https://www.offensive-security.com/metasploit-unleashed/`.

67. OWASP FOUNDATION. *OWASP ZAP. Version 2.11.1* [soft.]. 2014 [cit. 2022-03-21]. Dostupné z: `https://www.zaproxy.org/`.

68. QUALYS, Inc. *SSL Report: seznamovak.fit.cvut.cz (147.32.232.27)* [online]. 2022 [cit. 2022-04-09]. Dostupné z: `https://www.ssllabs.com/ssltest/analyze.html?d=seznamovak.fit.cvut.cz`.

69. OWASP FOUNDATION, Inc. *Server Leaks Information via 'X-Powered-By' HTTP Response Header Field(s)* [online]. 2022 [cit. 2022-04-15]. Dostupné z: `https://www.zaproxy.org/docs/alerts/10037/`.

70. BERGMANN, Sebastian. *php-token-stream. Version 4.0.4: Wrapper around PHP's tokenizer extension.* [Soft.]. 2020 [cit. 2022-03-21]. Dostupné z: `https://github.com/sebastianbergmann/php-token-stream`. GitHubový repozitář projektu php-token-stream.

71. BERGMANN, Sebastian. *Packagist: phpunit/php-token-stream* [online]. 2020 [cit. 2022-03-02]. Dostupné z: `https://packagist.org/packages/phpunit/php-token-stream`.

72. SENSIO LABS. *Inflector Component. Version 5.4.3: Inflector converts words between their singular and plural forms (English only).* [Soft.]. 2020 [cit. 2022-03-21]. Dostupné z: `https://github.com/symfony/inflector`. GitHubový repozitář projektu Inflector.

73. SENSIO LABS. *New in Symfony 5.1: Deprecated the Inflector component* [online]. 2020 [cit. 2022-03-02]. Dostupné z: `https://symfony.com/blog/new-in-symfony-5-1-deprecated-the-inflector-component`.

74. FORUM OF INCIDENT RESPONSE AND SECURITY TEAMS. *Common Vulnerability Scoring System version 3.1: Specification Document (Revision 1)* [online] [cit. 2022-04-21]. Dostupné z: https://www.first.org/cvss/v3-1/cvss-v31-specification_r1.pdf.

75. FORUM OF INCIDENT RESPONSE AND SECURITY TEAMS. *Common Vulnerability Scoring System Version 3.1 Calculator* [online] [cit. 2022-04-21]. Dostupné z: https://www.first.org/cvss/calculator/3.1.

76. STEUBE, Jens; GRISTIN, Gabriele. *Hashcat. Version 6.2.5: Advanced password recovery* [soft.]. 2022 [cit. 2022-03-07]. Dostupné z: https://hashcat.net/hashcat/.

77. HASHCAT. *Soubor best64.rule* [online]. 2018 [cit. 2022-03-02]. Dostupné z: https://github.com/hashcat/hashcat/blob/master/rules/best64.rule. GitHubový repozitář projektu Hashcat.

78. *Benchmark Hashcat with Nvidia 3090, 3080, 2080 TI, 1080 TI, 2070S, TESLA P100* [online]. 2021 [cit. 2022-03-12]. Dostupné z: https://www.onlinehashcrack.com/tools-benchmark-hashcat-gtx-1080-ti-1070-ti-rtx-2080-ti-rtx-3090-3080.php.

79. AMAZON.COM, INC. *Amazon* [online]. 1994 [cit. 2022-04-21]. Dostupné z: https://www.amazon.com/.

80. AMAZON WEB SERVICES, INC. *Cloud Computing Service* [online]. 2006 [cit. 2022-04-21]. Dostupné z: https://aws.amazon.com/.

81. AMAZON WEB SERVICES, INC. *Amazon EC2 P4d Instances* [online]. 2020 [cit. 2022-03-12]. Dostupné z: https://aws.amazon.com/ec2/instance-types/p4/.

82. CROLEY, Sam. *Hashcat v6.1.1 benchmark on the Nvidia Tesla A100 PCIE variant GPU* [online]. 2020 [cit. 2022-03-12]. Dostupné z: https://gist.github.com/Chick3nman/d65bcd5c137626c0fcb05078bba9ca89.

Příloha A

Seznam použitých zkratek

API Application Programming Interface

ASCII American Standard Code for Information Interchange

ASVS Application Security Verification Standard

AWS Amazon Web Services

BEAST Browser Exploit Against SSL/TLS

CSRF Cross-Site Request Forgery

CVSS Common Vulnerability Scoring System

ČVUT České vysoké učení technické v Praze

DoS Denial of Service

DBMS Database Management System

FIRST Forum of Incident Response and Security Teams

FIT Fakulta informačních technologií ČVUT

GPL General Public Licence

HTTP Hypertext Transfer Protocol

HTTPS Hypertext Transfer Protocol Secure

HTML Hypertext Markup Language

JSON JavaScript Object Notation

LDAP Lightweight Directory Acess Protocol

MD5 Message-Digest algorithm 5

MFA Multi-Factor Authentication

MVC Model-View-Controller

MVP Model-View-Presenter

MySQL My Structured Query Language

NIST National Institute of Standards and Technology

NSE Nmap Scripting Engine

OWASP Open Web Application Security Project

OWASP ZAP Zed Attack Proxy

PHP PHP: Hypertext Preprocessor, nebo Personal Home Page

REST Representational State Transfer

SHA1 Secure Hash Algorithm 1

SQL Structured Query Language

TCP Transmission Control Protocol

TLS Transport Layer Security

URL Uniform Resource Locator

WYSIWYG What You See Is What You Get

XSS Cross-site Scripting

Příloha **B**

Seznam závislostí aplikace Seznamovák

nette/application základ pro tvorbu interaktivních webových aplikací

nette/bootstrap obsahuje kód pro inicializaci prostředí a spouštění aplikace

nette/caching balíček, který přidává práci s *cache*

nette/database obsahuje podporu pro různé druhy databází

nette/di balíček pro práci s *dependency injection*

nette/finder průzkumník souborů

nette/forms knihovna pro bezpečnou práci s formuláři

nette/http knihovna pro práci s HTTP

nette/mail knihovna pro posílání emailů

nette/robot-loader nástroj zajišťující automatické načítání použitých tříd

nette/security autorizace a autentizace s využitím ACL

nette/utils nástroje pro manipulaci s JSON formátem a generování hesel

latte/latte šablona pro rychlý a bezpečný vývoj frontendu v PHP

tracy/tracy nástroj pro ladění PHP kódu

ublaboo/datagrid umožňuje používat pohledy a filtry pro Nette framework

voda/date-input vstupní pole pro datum a čas

robmorgan/phinx správa migrace databáze

psr/http-message interface pro HTTP požadavek

tecnickcom/tcpdf umožňuje vygenerování PDF z databáze

dg/mysql-dump rozšíření pro stáhnutí MySQL databáze

codeception/codeception framework pro testování

radekdostal/nette-datetimepicker validátor zadávaných datumů

ondrs/hi rozšíření pro česká jména a příjmení

ext-json rozšíření pro zakódování objektů do formátu JSON

Příloha C

Vyhodnocení zranitelností dle metodiky CVSS

Název zranitelnosti	*Vector string* dle CVSS verze 3.1
Použití zastaralých síťových protokolů	AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:L/A:N
Slabé parametry pro hashovací algoritmus	AV:N/AC:H/PR:H/UI:N/S:U/C:H/I:H/A:N
Hashování s nedostatečnou entropií	AV:N/AC:H/PR:H/UI:N/S:U/C:H/I:H/A:N
Slabé požadavky na heslo	AV:N/AC:H/PR:H/UI:N/S:U/C:H/I:H/A:N
Stahování souborů bez kontroly integrity	AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:N/A:N
Používání zranitelných JS knihoven	AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:N/A:N
Používání neudržovaných komponent	AV:N/AC:H/PR:N/UI:N/S:U/C:H/I:N/A:N
Citlivé informace v přístupném souboru	AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N
Slabé šifry při síťové komunikaci	AV:N/AC:H/PR:N/UI:N/S:U/C:L/I:N/A:N
Zveřejnění informací v HTTP hlavičce	AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N
Chybějící hlavička X-Content-Type-Options	AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N
Chybějící hlavička X-Frame-Option	AV:N/AC:H/PR:N/UI:R/S:U/C:N/I:L/A:N
Chybějící hlavička Content-Security-Policy	AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N
Chybějící hlavička Expect-CT	AV:N/AC:H/PR:N/UI:R/S:U/C:L/I:N/A:N

Tabulka C.1: Nalezené zranitelnosti v aplikaci Seznamovák

Příloha D

Obsah přiloženého CD

- `readme.txt` ..... stručný popis obsahu CD
- `report` ..... složka se zprávami nástrojů o nalezených zranitelnostech
- `text` ..... text práce
  - `thesis` ..... zdrojová forma práce ve formátu LaTeX
  - `DP_Kriz_Jaroslav_2022.pdf` ..... text práce ve formátu PDF