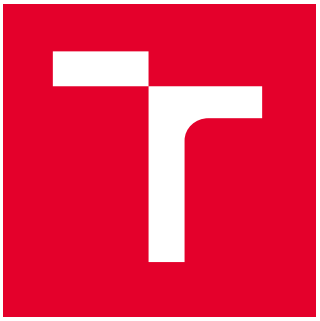


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

SOFTWAROVÝ NÁSTROJ PRO KONTROLU BEZPEČNÉHO NASTAVENÍ MYSQL A MARIADB DATABÁZÍ

A SOFTWARE TOOL FOR CHECKING THE SECURE SETUP OF MYSQL AND MARIADB DATABASES

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — Bc. Jindřich Šíma

AUTHOR

VEDOUCÍ PRÁCE — RNDr. Ing. Pavel Šeda, Ph.D.

SUPERVISOR

BRNO 2025

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Diplomová práce

magisterský navazující studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Bc. Jindřich Šíma ***ID:*** 231287

Ročník: 2 ***Akademický rok:*** 2024/25

NÁZEV TÉMATU:

Softwarový nástroj pro kontrolu bezpečného nastavení MySQL a MariaDB databází

POKYNY PRO VYPRACOVÁNÍ:

Hlavním cílem diplomové práce je návrh a implementace nástroje pro kontrolu nastavení databáze MariaDB a MySQL databází. Cílem práce je rovněž navrhnout parametry databázového systému, které musí být monitorovány, včetně jejich vhodného nastavení. V teoretické části práce analyzujte konfigurační soubory MariaDB a MySQL databáze a zhodnoťte jejich vliv na bezpečnost. Hlavním výstupem práce je technická specifikace doporučující bezpečné nastavení databázového systému a softwarový nástroj kontrolující nastavení dle vytvořené specifikace. Vytvořené řešení bude sloužit jako knihovna vytvořená v jazyce Python tak, aby generovala .pdf report s výstupem kontroly nastavení. Funkčnost aplikace důkladně otestujte v experimentálním prostředí.

DOPORUČENÁ LITERATURA:

[1] GARCIA-MOLINA, Hector, Jeffrey D. ULLMAN a Jennifer WIDOM. Database system implementation. Upper Saddle River: Prentice Hall, 2000. xv, 653 s. ISBN 0-13-040264-8. (EN)

[2] Obe, Regina O., and Leo S. Hsu. PostgreSQL: up and running: a practical guide to the advanced open source database." O'Reilly Media, Inc.", 2017.

Termín zadání: 10.2.2025 ***Termín odevzdání:*** 27.5.2025

Vedoucí práce: RNDr. Ing. Pavel Šeda, Ph.D.

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

ABSTRAKT

Tato diplomová práce se zaměřuje na návrh a realizaci testovacích programů pro ověření bezpečnosti konfigurací databázových systémů MySQL a MariaDB. Práce je rozdělena do tří částí.

První část poskytuje přehled typů databází, jejich nejčastějších zranitelností a útoků na ně. Druhá část je zaměřena na návrh a implementaci nástrojů pro automatizované testování bezpečnosti konfigurací databázových systémů. Třetí část se věnuje vytvoření testovacího prostředí pro praktické ověření funkčnosti nástrojů. Obsahuje demonstrace možných útoků na nedostatečně zabezpečené konfigurace a diskutuje limity řešení, možné scénáře nasazení a návrhy na další rozšíření.

Výsledkem práce jsou nástroje na automatické testování databázových systémů, které své poznatky zapisují do přehledného reportu ve formátu PDF. Výsledky přispívají ke zvýšení povědomí o důležitosti správného zabezpečení databázových systémů a ulehčení kontroly bezpečnosti konfigurací.

KLÍČOVÁ SLOVA

Automatizované testování, Bezpečnost databází, Databázové systémy, MariaDB, MySQL, Python, SSL/TLS, Zranitelnosti

ABSTRACT

This thesis focuses on the design and implementation of test programs to verify the security of MySQL and MariaDB database systems configuration. The thesis is divided into three parts.

The first part provides an overview of the types of databases, their most common vulnerabilities and attacks on them. The second part focuses on the design and implementation of tools for automated security testing of database system configurations. The third part is dedicated to the creation of a test environment for practical verification of the tools functionality. It includes demonstrations of possible attacks on poorly secured aonfigurations and discusses the limitations of the solution, possible deployment scenarios and suggestions for further improvements.

The work results in tools for automatic testing of database systems that write their findings in PDF report. The results contribute to raising awareness of the importance of proper security of database systems and providing configuration security checks.

KEYWORDS

Automated testing, Database management systems, Database security, MariaDB, MySQL, Python, SSL/TLS, Vulnerabilities

Vysázeno pomocí balíčku `thesis` verze 4.09; `https://latex.fekt.vut.cz/`

ŠÍMA, Jindřich. *Softwarový nástroj pro kontrolu bezpečného nastavení MySQL a MariaDB databází*. Diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2024. Vedoucí práce: RNDr. Ing. Pavel Šeda, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Bc. Jindřich Šíma
VUT ID autora:	231287
Typ práce:	Diplomová práce
Akademický rok:	2024/25
Téma závěrečné práce:	Softwarový nástroj pro kontrolu bezpečného nastavení MySQL a MariaDB databází

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

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

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Rád bych poděkoval vedoucímu diplomové práce panu RNDr. Ing. Pavlu Šedovi, Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam obrázků

Seznam tabulek

Úvod

Databázové systémy mají klíčovou roli v současném digitálním světě, kde jsou využívány k ukládání, správě a analýze velkých objemů dat. Správné nastavení databázových systémů, jako jsou MariaDB a MySQL, má zásadní vliv na zajištění jejich bezpečnosti a spolehlivosti. V kontextu rostoucího počtu kybernetických útoků a striktních regulačních požadavků je kladen stále větší důraz na ochranu dat a prevenci rizik vyplývajících z nevhodné konfigurace.

MariaDB a MySQL jsou široce používané databázové systémy. Svou popularitu získaly nejen díky své dostupnosti a výkonu, ale také díky své flexibilitě v konfiguraci. Tato flexibilita však přináší i rizika, protože špatně nastavené databázové systémy mohou být zranitelné vůči útokům, jejichž následkem bývá únik dat či odepření služby. Manuální kontrola konfigurace může být zdlouhavá, a proto je zapotřebí automatizovaných nástrojů, které by tento proces zefektivnily.

Práce je rozdělena do tří hlavních částí. První kapitola se zaměřuje na bezpečnost databází, nejčastější zranitelnosti, útoky a existující řešení. Druhá kapitola se věnuje návrhu a implementaci softwarových nástrojů. Třetí kapitola popisuje přípravu testovacího prostředí a možná nebezpečí při špatně zvolené konfiguraci. Součástí výstupu je také report s doporučeními pro bezpečné nastavení.

1 Databázové systémy

Za databázový systém se považuje celek spojen z **báze dat** (DB – Database) a **systému řízení báze dat** (DBMS – Database Management System).

Báze dat je obvykle organizovaná sbírka strukturovaných informací nebo dat, které jsou uloženy elektronicky v počítačovém systému. Data jsou v dnešní době nejčastěji formátována do řádků a sloupců v tabulkách, což zefektivňuje jejich zpracování a vyhledávání.

Systém řízení báze dat je softwarové řešení, které umožňuje uživateli práci s databází. Poskytované funkce a vlastnosti DBMS se mohou lišit, ale mezi základní patří:

- vytváření, upravování a dotazování na databázi
- řízení souběžnosti – k datům může přistupovat více uživatelů bez vzájemného konfliktu
- mechanismy pro zálohu a obnovu dat
- nástroje pro zabezpečení a integritu dat
- řízení přístupu – nastavení a případné omezení kdo má přístup k datům

V dnešní době je mnoho různých systémů řízení bází dat, ale mezi nejpoužívanější patří např. Oracle, MySQL, Microsoft SQL Server, PostgreSQL, MongoDB, Redis, Snowflake, ElasticSearch, IBM Db2, SQLite, Apache Cassandra, Microsoft Access, Splunk, Databricks, MariaDB [1]. Pro tuto práci jsou relevantní **MySQL** a **MariaDB**.

1.1 Rozdělení databází

Moderní databázové systémy se vyvíjejí s rostoucími nároky na ukládání, správu a analýzu dat. Každý typ databáze je navržen tak, aby odpovídal specifickým požadavkům a situacím, jako jsou vysoká dostupnost nebo zpracování velkých objemů nestrukturovaných dat. Tato část se zaměřuje na hlavní kategorie databází, jako **Relační databáze**, **Objektově orientované databáze**, **Distribuované databáze**, **Datové sklady** a **NoSQL databáze**. Každý z těchto přístupů nabízí unikátní vlastnosti a výhody pro různé scénáře využití v moderním IT prostředí [2].

1.1.1 Relační databáze

Relační databáze je typ databáze, která organizuje data do řádků a sloupců. Ty pak dohromady tvoří tabulku, v níž jsou data vzájemně propojena. Data jsou obvykle rozdělena do více tabulek, které lze spojit pomocí primárního a cizího klíče. Primární klíče fungují jako unikátní identifikátory v tabulkách a cizí klíče demonstrují různé

vztahy mezi tabulkami. Ty jsou obvykle znázorněny pomocí různých typů datových modelů [3].

Například databáze s tabulkou **Zákazníci** reprezentující zákazníky obchodu, kde je primární klíč **zakaznikID**, tabulkou **Produkty** reprezentující položky v obchodu s primárním klíčem **produktID**, tabulkou **Objednávky** reprezentující nákupy s primárním klíčem **objednavkaID** a cizí klíče **zakaznikID** a **produktID**. V této zjednodušené databázi je možné znázornit vyřizování objednávek přehledně a bez zbytečné redundance záznamů viz 1.1.

Obr. 1.1: Příklad relační databáze

Relační databáze jsou obvykle spojeny s transakčními databázemi, které hromadně provádějí příkazy neboli transakce. Vhodným příkladem je třeba bankovní převod. Z jednoho účtu je odečtena definovaná částka, která je přičtena na jiný účet. Odečítá a přičítá se celková částka a tato transakce nemůže probíhat po částech. Transakce mají charakteristické vlastnosti, které se skrývají pod zkratkou ACID (Atomicity Consistency Isolation Durability)

- **Atomicita** – změny dat se provádějí, jako by šlo o jedinou operaci. To znamená, že transakce je provedena celá, nebo neproběhne vůbec.
- **Konzistence** – data zůstávají v konzistentním stavu od začátku do konce.
- **Izolovanost** – mezistavy transakcí nejsou viditelné pro ostatní transakce. Díky tomu se souběžně probíhající transakce jeví jako serializované.
- **Trvalost** – po úspěšném dokončení transakce jsou změny v datech zachovány a nejsou zrušeny ani při selhání systému.

Tyto vlastnosti umožňují spolehlivé zpracování transakcí. [3]

SQL

Strukturovaný dotazovací jazyk (SQL – Structured Query Language) vytvořili Don Chamberlin a Ray Boyce pro společnost IBM. Je to standardní programovací jazyk pro interakci s relačními databázemi. Umožňuje správci databáze jednoduše přidávat, aktualizovat nebo mazat řádky. Dotazy SQL umožňují také uživatelům získávat

data z databází pomocí několika řádků kódu. Protože jsou relační databáze hodně spojovány s SQL, jsou také někdy nazývány „SQL databáze“. [3]

Výhody relačních databází

Hlavní výhodou relačních databází je možnost vytvářet smysluplné informace spojováním tabulek. Spojování tabulek umožňuje pochopit vztahy mezi daty, případně souvislost mezi tabulkami. Jazyk SQL umožňuje počítat, spojovat, seskupovat a kombinovat dotazy. Umí také provádět základní matematické a mezisoučtové funkce a logické transformace. Informace mohou být řazeny podle data, jména nebo libovolného sloupce. Díky těmto vlastnostem je relační přístup nejoblíbenějším dotazovacím nástrojem. Oproti jiným databázovým formátům má následující výhody:

- **Snížení redundance** – tabulky by měly zaznamenávat pouze jedinečné záznamy. Ostatní jinak redundantní hodnoty lze získat pomocí vztahů mezi tabulkami. Uložené procedury také omezují opakující se práci.
- **Snadné zálohování** – relační databáze nabízí snadné možnosti exportu a importu, takže zálohování a obnova jsou jednoduché. Exportování může probíhat při běhu databáze, což usnadňuje obnovu. Moderní databáze založené na cloudu mohou provádět nepřetržité zrcadlení, takže případná ztráta dat při obnově je obvykle v řádu sekund. [3]

1.1.2 Objektově orientované databáze

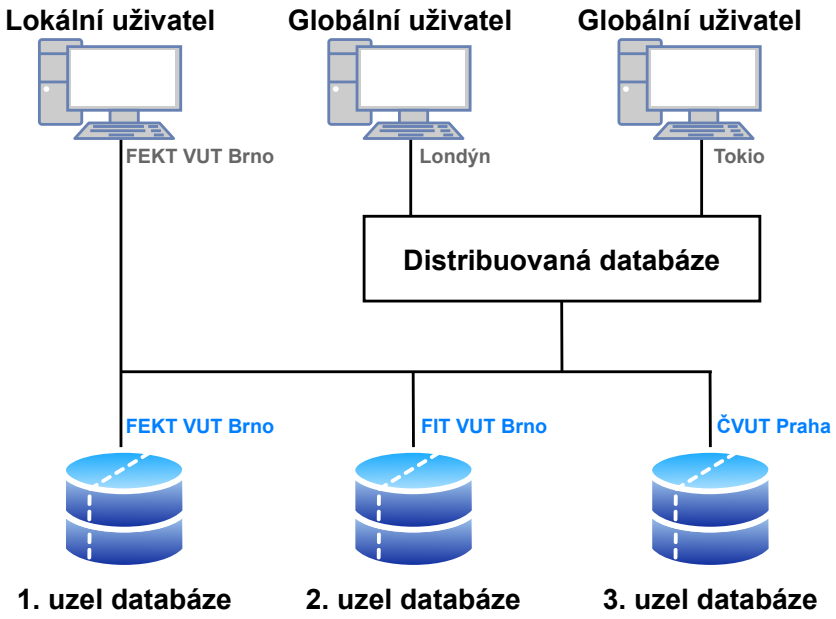
Objektově orientovaná databáze je taková databáze, ve které jsou informace reprezentovány ve formě objektů, podobně jako v objektově orientovaném programování. Tento přístup zahrnuje podporu pro třídy objektů, dědičnost vlastností i metod tříd. Umožňuje také vytváření podtříd a jejich instancí.

Mnoho vývojářů používá k vytváření aplikací objektově orientované programovací jazyky (C#, Java, PHP, Python). Použití relačních databází k ukládání a načítání objektů vyžaduje převádění mezi složitými objekty a řádky z více tabulek. V některých aplikacích je tedy jednodušší přistupovat k datům v podobě objektu, než k datům s mnoha vztahy uloženými ve více tabulkách. Tento typ databází se často používá ke zpracování složitých dat ve vědeckých odvětvích, strojírenství nebo telekomunikacích. [4, 5]

1.1.3 Distribuované databáze

Distribuované databáze jsou databáze, které jsou rozprostřeny na více místech a nesdílí fyzické komponenty. Obvykle fungují na dvou nebo více vzájemně propojených serverech. Každé místo, kde běží část databáze, se nazývá instance nebo uzel, viz 1.2.

Ukládání distribuovaných databází lze provést dvěma způsoby – replikací nebo fragmentací.



Obr. 1.2: Distribuovaná databáze

- **Replikace** – celé databáze jsou uloženy na dvou nebo více místech. Zvyšuje se tím dostupnost dat v různých lokalitách a požadavky mohou být zpracovávány paralelně. Nevýhodou je, že se data musí neustále aktualizovat. Jakákoli změna dat na jednom místě musí být provedena i na všech ostatních místech, jinak může dojít k nekonzistencím. To způsobuje velký režijní provoz.
- **Fragmentace** – databáze jsou fragmentovány (rozděleny na části) a každý fragment je uložen na různém místě, kde je zapotřebí. Nevytváří kopie dat, tudíž nekonzistence není problém. Fragmentaci lze provést horizontální a vertikální viz 1.3.
  - **Horizontální fragmentace** – rozdělení tabulek po řádcích ve více uzlech.
  - **Vertikální fragmentace** – rozdělení tabulek po sloupcích ve více uzlech. Každý uzel musí obsahovat společný identifikátor, aby bylo zajištěno bezztrátové spojení.

Distribuované databáze se používají například v multimediálních aplikacích, hotelových řetězcích, výrobních řídících systémech, armádních systémech a informačních systémech pro řízení. [6, 7, 8]

Původní tabulka

ID Zákazníka	Jméno	Příjmení	Město
1	Alice	Málá	Aš
2	Bob	Nuteš	Berlín
3	Cecil	Oráb	Cvikov
4	Eva	Sovová	Ejpovice

Vertikální fragmentace

ID Zákazníka	Jméno	Příjmení
1	Alice	Málá
2	Bob	Nuteš
3	Cecil	Oráb
4	Eva	Sovová

ID Zákazníka	Město
1	Aš
2	Berlín
3	Cvikov
4	Ejpovice

Horizontální fragmentace

ID Zákazníka	Jméno	Příjmení	Město
1	Alice	Málá	Aš
2	Bob	Nuteš	Berlín

ID Zákazníka	Jméno	Příjmení	Město
3	Cecil	Oráb	Cvikov
4	Eva	Sovová	Ejpovice

Obr. 1.3: Distribuovaná databáze – fragmentace

1.1.4 Datové sklady

Datový sklad je systém, který sdružuje data z různých zdrojů do jednoho centrálního a konzistentního datového úložiště. Datové sklady jsou určeny především k provádění dotazů a analýz a často obsahují velké množství historických dat. Pomáhají také připravit data pro jejich vytěžování (data mining), strojové učení a umělou inteligenci. Jejich analytické schopnosti umožňují organizacím získávat z dat cenné poznatky a zlepšovat tak rozhodování organizací. [9, 10]

1.1.5 NoSQL databáze

NoSQL databáze, které jsou někdy označovány jako nerelační databáze, umožňují ukládat a manipulovat s nestrukturovanými nebo také částečně strukturovanými daty. Aplikace mohou zpracovávat velký objem dat z různých zdrojů, jako jsou sociální média, chytré senzory a databáze externích stran. Tato různorodá data se nemusí hodit do relačního modelu. Vynucení tabulkových struktur může vést k redundanci, duplikaci dat a problémům s výkonem. Mezi hlavní výhody NoSQL databází patří flexibilita, škálovatelnost a vysoký výkon.

- **Flexibilita** – NoSQL databáze poskytují flexibilní schémata, která umožňují

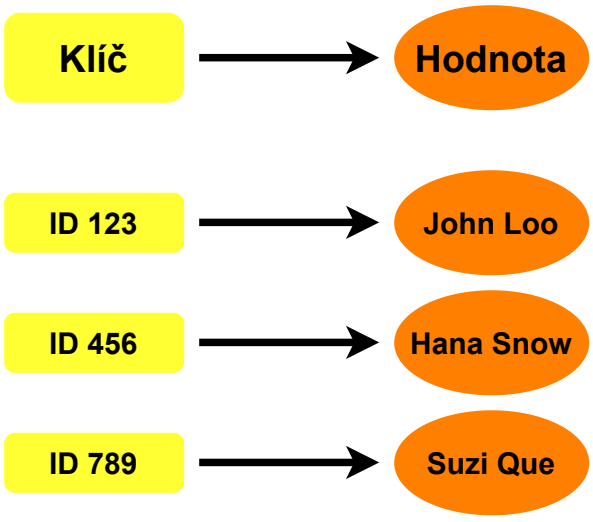
rychlejší a iterativnější vývoj. Proto jsou ideální pro částečně strukturovaná a nestrukturovaná data.

- **Škálovatelnost** – NoSQL databáze jsou navrženy tak, aby se místo drahých serverů daly škálovat pomocí distribuovaných hardwarových clusterů.
- **Vysoký výkon** – NoSQL databáze jsou optimalizovány pro specifické datové modely a přístupové vzory. Ty umožňují vyšší výkon než obdobné funkce pro relační databáze.

Nerelační databáze nabízejí organizaci dat mnoha způsoby. Díky různým datovým strukturám je lze použít pro analýzu dat, správu objemných dat a vývoj mobilních aplikací. Čtyři nejběžnější typy nerelačních databází jsou popsány v sekcích: **Databáze párů klíč-hodnota**, **Databáze dokumentů**, **Databáze sloupců** a **Grafové databáze**. [11, 12]

Databáze párů klíč-hodnota

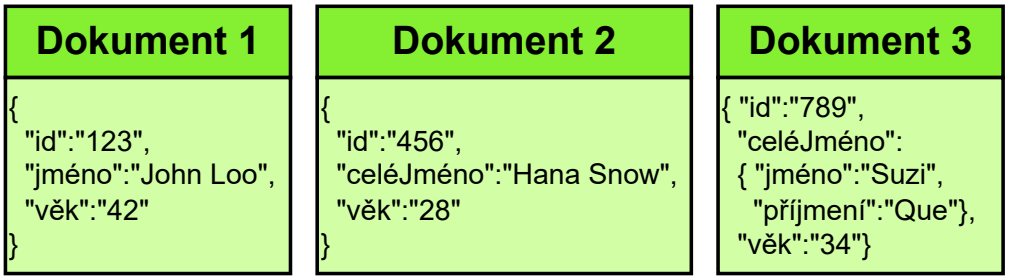
Databáze tohoto typu jsou vysoce dělitelné a umožňují horizontální rozšiřování na úrovni, které jiné NoSQL databáze nemohou dosáhnout. Data jsou ukládána jako kolekce párů klíč-hodnota, přičemž klíč slouží jako jedinečný identifikátor. Klíče a hodnoty mohou být cokoli, od jednoduchých objektů až po složitější složené objekty, viz 1.4. Jsou vhodné pro použití ve hrách, reklamních technologiích nebo tzv. internetu věcí (IoT – Internet of Things). [11, 13]



Obr. 1.4: Databáze párů klíč-hodnota (převzato a upraveno z [13])

Databáze dokumentů

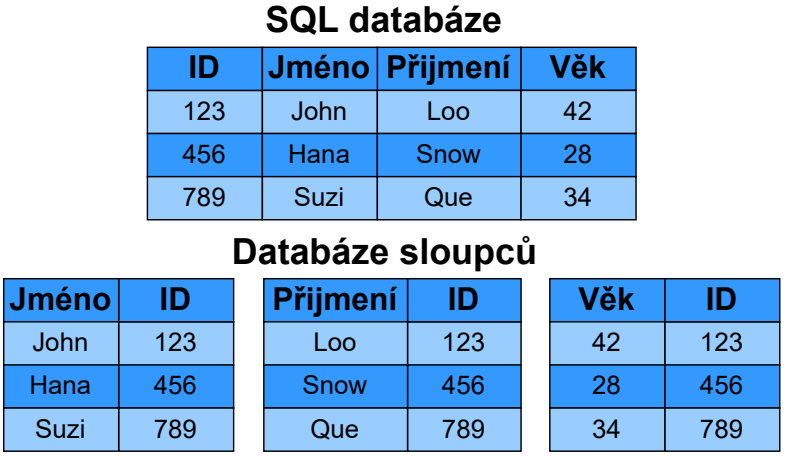
Databáze tohoto typu ukládají data jako dokumenty. Pomáhají při správě částečně strukturovaných dat. Data jsou obvykle uložena ve formátech JSON (JavaScript Object Notation), XML (Extensible Markup Language) nebo BSON (Binary JSON). Při použití v aplikacích zůstávají data pohromadě, čímž se snižuje množství překladů potřebných pro jejich použití. Díky tomu, že se schémata dat nemusí shodovat napříč dokumenty, získávají vývojáři větší volnost. U složitějších aplikací to ale může být problematické a vést i k poškození dat, viz 1.5. Obvykle se využívají pro katalogy, uživatelské profily nebo systémy pro správu obsahu. [13, 14, 15]



Obr. 1.5: Databáze dokumentů

Databáze sloupců

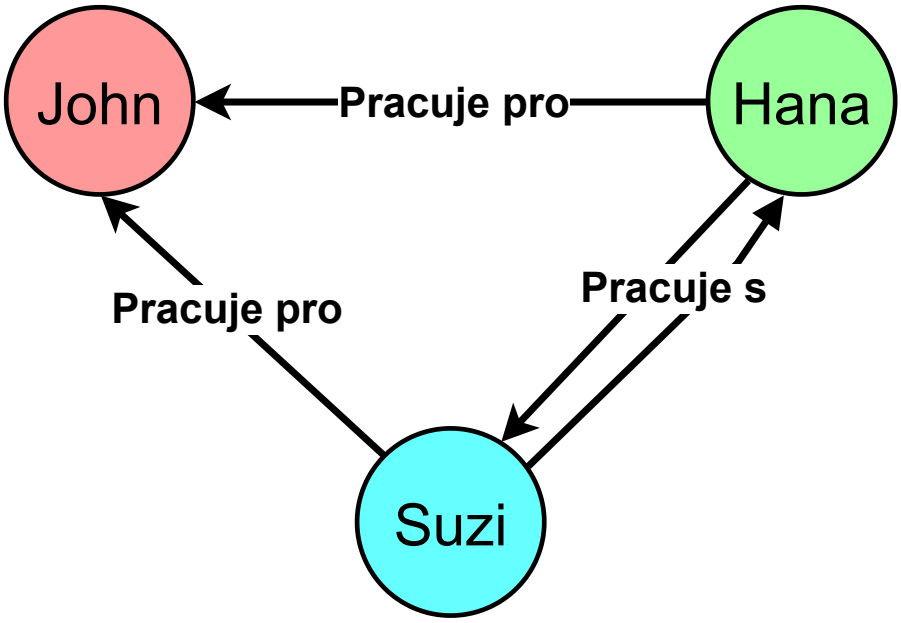
Databáze tohoto typu ukládají informace ve sloupcích, což umožňuje uživatelům přistupovat pouze ke konkrétním sloupcům, které potřebují, bez alokace další paměti na nerelevantní data. Snaží se eliminovat nedostatky databází dokumentů a klíč-hodnota, ale jedná se o složitější systém na správu, viz 1.6. [12, 13, 14]



Obr. 1.6: Databáze sloupců (převzato a upraveno z [13])

Grafové databáze

Databáze tohoto typu jsou vytvořeny tak, aby usnadňovaly vytváření a provoz aplikací, které pracují s vysoce propojenými soubory dat. Datové prvky jsou uloženy jako uzly, hrany nebo vlastnosti. Uzlem může být jakýkoliv objekt, místo nebo osoba, které mohou mít přiřazeny libovolný počet vlastností. Hrany definují vztahy mezi uzly a mají vždy počáteční uzel, koncový uzel, typ a směr, viz 1.7. Počet a druh hran, které uzel může mít, není omezen. Typické použití grafových databází jsou sociální sítě, doporučovací systémy, detekce podvodů a znalostní grafy. [11, 12, 13, 14, 15]



Obr. 1.7: Grafová databáze

1.2 MySQL

MySQL je open-source DBMS, který je vyvíjen a distribuován společností Oracle Corporation [16]. Název vznikl spojením *My*, což je jméno dcery spoluzakladatele Michaela Wideniuse a zkratkou jazyka *SQL*, který se používá pro práci s daty v relačních databázích [17].

Relační databáze ukládá data do samostatných tabulek místo toho, aby všechna data ukládala do jednoho velkého úložiště. Logický model nabízí flexibilní programovací prostředí díky objektům jako jsou databáze, tabulky, řádky a sloupce. Databázové struktury jsou organizovány do souborů optimalizovaných pro rychlost [16].

MySQL umožňuje uživatelům přímou interakci s databázemi pomocí SQL, ale častěji se používá spolu s jinými programy pro vytvoření aplikací, které potřebují

schopnosti relačních databází. MySQL používá mnoho webových aplikací a stránek, např. PayPal, GitHub, Facebook, Twitter a YouTube [18].

1.2.1 Konfigurační soubor

Konfigurační soubor je důležitý pro nastavení parametrů databázového serveru. V závislosti na operačním systému se hlavní konfigurační soubor nazývá buď `my.ini` (Windows) nebo `my.cnf` (Linux). Se jménem se zároveň mění i výchozí umístění konfiguračního souboru. Pro Windows to jsou:

- `C:\Windows\my.ini`
- `C:\my.ini`
- `C:\Program Files\MySQL\MySQL Server X.X\my.ini`

Pro Linux to jsou:

- `/etc/my.cnf`
- `/etc/mysql/my.cnf`
- `/usr/etc/my.cnf`
- `~/.my.cnf`

Když je nalezeno více hlavních konfiguračních souborů, tak se použije poslední (podle vypsání výše). [19]

```
[client]
port                            = 3306
socket                          = /tmp/mysql.sock
```

V `client` části se nastavuje konfigurace pro MySQL klienta a určuje např. port nebo socket, ke kterému se klient připojuje. [20, 21]

```
[mysqld_safe]
open_files_limit                = 8192
user                            = mysql
log-error                       = error.log
```

V `mysqld_safe` části se nastavují parametry pro bezpečnější start serveru. To je doporučený způsob spouštění MySQL serveru na Unixových systémech. Je možné zde nastavit maximální počet současně otevřených souborů, uživatele, pod kterým bude proces MySQL serveru běžet, a cestu k souboru, kam se budou zapisovat chyby MySQL serveru. [20, 22]

```
[mysqld]
port                            = 3306
socket                          = /tmp/mysql.sock

max_allowed_packet              = 16M
default_storage_engine          = InnoDB

max_connections                 = 151
max_user_connections            = 50

log_error                       = error.log
log_warnings                    = 2

sql_mode                        = TRADITIONAL,ANSI
```

V `mysqld` části se nachází nastavení pro MySQL server jako číslo portu, na kterém naslouchá na příchozí spojení, umístění socketového souboru, který se používá pro lokální spojení, limity na maximální velikost příchozích/odchozích paketů na server a nastavení výchozího způsobu ukládání tabulek. Můžeme zde být nastaven také maximální počet současných připojení na server, maximální počet současných připojení jednoho uživatele, cestu k souboru pro logování chyb, úroveň detailnosti logů a režimy pro interpretaci SQL dotazů. [20, 23]

```
[mysql]
no_auto_rehash
max_allowed_packet              = 16M
```

V `mysql` části se nachází nastavení nástroje mysql, který slouží pro komunikaci s databází, jako nastavení automatického znovuvytvoření hashovací tabulky a maximální velikost paketu pro příjem/odeslání. [20, 21]

```
[mysqldump]
max_allowed_packet              = 16M
```

V `mysqldump` části se nachází nastavení nástroje mysqldump, který slouží k zálohování databází a tabulek. [20, 24]

1.3 MariaDB

MariaDB je komunitou vyvíjený DBMS, který je nástupnickou větví MySQL. Hlavními důvody vzniku byly obavy z akvizice společností Oracle a udržení licence GNU GPL (GNU General Public License). Vývoj je veden některými z původních vývojářů MySQL. Název je odvozen od jména mladší dcery Michaela Wideniuse *Maria*. Projekt byl koupen soukromou společností K1 v roce 2024. [25, 26]

Jelikož vychází z MySQL, tak také pracuje s daty relačních databází, ale na rozdíl od MySQL obsahuje několik nových funkcí (úložné enginy Aria a XtraDB, clusterovací technologie Galera) a ve většině případů je rychlejší. MariaDB používá mnoho společností, např. RedHat [27], Wikipedia [28], Google [29], Nokia, a Samsung [30].

1.3.1 Konfigurační soubor

Konfigurační soubor je důležitý pro nastavení parametrů databázového serveru. Umožňuje nastavit většinu systémových proměnných serveru, když nejsou nastaveny, použije se jejich výchozí hodnota. V závislosti na operačním systému se konfigurační soubor nazývá buď `my.ini` (Windows) nebo `my.cnf` (Linux i Windows). Se jménem se zároveň mění i výchozí umístění konfiguračního souboru. Pro Windows to jsou hlavně:

- `C:\Windows\my.ini`
- `C:\my.ini`
- `C:\Program Files\MariaDB XX.X\my.ini`

Pro Linux to jsou především:

- `/etc/my.cnf`
- `/etc/mysql/my.cnf`
- `~/.my.cnf`

Při nalezení více konfiguračních souborů se přepíší hodnoty v závislosti na pořadí výše (nižší soubory mají přednost).

Skladba konfiguračních souborů se řídí následujícími pravidly:

- Řádky začínající mřížkou (#) jsou komentáře.
- Prázdné řádky jsou ignorovány.
- Názvy skupin nastavení používají hranaté závorky ([název-skupiny]), viz 1.1.
- Jeden název skupiny se může objevit vícekrát.
- Pomlčky (-) a podtržítka (_) jsou vzájemně zaměnitelné.
- Je možné se odkazovat i na jiné konfigurační soubory i zadat složku, ze které se použijí všechny `.cnf` a `.ini` soubory v abecedním pořadí. [31]

Tab. 1.1: Skupiny nastavení serveru v konfiguračním souboru MariaDB

Skupiny	**Popis**
[client-server]	nastavení pro komunikaci mezi klientem a serverem, např. socket a číslo portu
[server]	nastavení MariaDB serveru
[mysqld]	nastavení MariaDB i MySQL serveru
[mariadb]	nastavení MariaDB serveru
[mariadb-X.Y]	nastavení konkrétní verze MariaDB serveru, např. [mariadb-11.7]
[mariadbd]	nastavení MariaDB serveru, přidáno ve verzi MariaDB 10.4.6
[mariadbd-X.Y]	nastavení konkrétní verze MariaDB serveru, např. [mariadbd-11.7], přidáno ve verzi MariaDB 10.4.6

1.4 Bezpečnost databází

Zajištění bezpečnosti databází je nezbytné pro udržení důvěrnosti, integrity a dostupnosti uložených dat. Jedním z nejčastějších následků útoků na databáze je únik dat. Z definice je únik dat selháním ve snaze uchovat důvěrnost dat v databázi. Jak velkou škodu mohou úniky dat způsobit, záleží na různých faktorech:

- **Narušení duševního vlastnictví** – obchodní tajemství a vynálezy mohou být klíčové pro schopnost udržet si konkurenční výhodu na trhu. Pokud je toto duševní vlastnictví odcizeno nebo zveřejněno, může být těžké až nemožné, zachovat nebo obnovit konkurenční výhodu.
- **Poškození pověsti** – zákazníci nebo partneři nemusí být ochotni kupovat produkty nebo služby, když mají pocit, že nemohou důvěřovat, ochraně sdílených dat.
- **Pokuty nebo sankce** – finanční dopad za nedodržení globálních předpisů jako zákon Sarbanes-Oxley (Sarbanes-Oxley Act – SOX), standard PCI DSS (Payment Card Industry Data Security Standard – PCI DSS), který udává požadavky na zpracování dat držitelů platebních karet, nebo obecné nařízení o ochraně osobních údajů (General Data Protection Regulation – GDPR) mohou být likvidační. V nejhorším případě mohou pokuty přesáhnout několik milionů dolarů za porušení.
- **Náklady na nápravu a upozornění zákazníků** – kromě nákladů na oznámení porušení zákazníkovi musí organizace, která byla narušena zaplatit forenzní vyšetřování, krizové řízení a opravu postižených systémů. [32]

1.4.1 Nejčastější zranitelnosti/útoky

Vnitřní hrozby

Bezpečnostní zranitelnosti, které pocházejí z jednoho ze tří zdrojů s přístupem k databázím:

- **Zlomyslný zaměstnanec** – má přístup k databázím a chce úmyslně způsobit škodu.
  Jako příklad lze uvést únik dat zaměstnanců firmy Tesla, která vyrábí elektromobily. Ten proběhl v roce 2023 díky dvěma bývalým zaměstnancům dané firmy, kteří vydali osobní údaje více než 75 000 zaměstnanců německému médiu Handelsblatt. To uniklá data nezveřejnilo a předešlo tak jejich zneužití. [33]
- **Nedbalý zaměstnanec** – jeho neúmyslnou chybou v konfiguraci se databáze stává zranitelná.
- **Infiltrovaný hacker** – osoba, která není interní ani externí zaměstnanec získá přístup k přihlašovacím údajům k databázím pomocí sociálního inženýrství nebo jiným nekalým způsobem. [32]
  Tímto mechanismem došlo k ohrožení bezpečnosti osobních dat a údajů o platebních kartách 40 milionů zákazníku amerického řetězce maloobchodů Target v roce 2013. Útočníci pronikli do sítě společnosti pomocí zranitelnosti u dodavatele třetí strany. Společnost údajně zaplatila až 162 milionů dolarů za škody spojené s útokem, soudní poplatky a vyrovnání. [34]

Vnitřní hrozby jsou jedny z nejčastějších příčin bezpečnostních incidentů a často jsou důsledkem toho, že mají přístup k databázím i zaměstnanci, kteří jej nepotřebují.

Lidská chyba

Nehody, slabá hesla, sdílení hesel a další nerozumná nebo neinformovaná uživatelská chování jsou častou příčinou nahlášených úniků dat. Podle studie společnosti Mimecast se na 95 % případů narušení bezpečnosti v roce 2024 podílel lidský faktor [35].

Například v roce 2022 zaměstnanci Microsoftu omylem odhalili citlivé přihlašovací údaje k vlastním Azure serverům na GitHubu. Potenciální útočníci měli možnost proniknout na další místa infrastruktury po počátečním přístupu do interního systému. Microsoft nezaznamenal žádný přístup k citlivým datům ani neoprávněné použití přihlašovacích údajů. [36]

Využití zranitelností softwaru

Útočníci se živí hledáním a využíváním zranitelností ve všech typech softwaru. Všichni hlavní komerční dodavatelé softwaru a open source platformy vydávají pravidelné aktualizace, které tyto zranitelnosti řeší, ale logická časová prodleva mezi

odhalením zranitelnosti a implementací opravy či aktualizace SW zvyšuje riziko napadení. [32]

Jedním z typů zranitelností SW je takzvaná **zero-day** zranitelnost. Její jméno odkazuje na počet dní (nula), kdy o zranitelnosti věděl výrobce softwaru nebo zařízení. Tento typ zranitelností je obzvláště nebezpečný, protože vystavuje nebezpečí velké množství uživatelů do doby, než se o něm dozví výrobce softwaru. **Zero-day** útok je tedy útok, který využije **zero-day** zranitelnost k nasazení malwaru, krádeži dat nebo jinému poškození uživatelů či organizací.

Ukázkou **zero-day** zranitelnosti je například **Log4Shell**. To je zranitelnost knihovny Log4J, což je open source Java knihovna, která sloužila pro zaznamenávání chybových zpráv. Útočníci mohli zranitelnost využít ke vzdálenému ovládání skoro jakéhokoliv zařízení s Java aplikacemi. Jelikož se používá v populárních programech jako iCloud nebo Minecraft, byly ohroženy stovky milionů zařízení. Databáze zranitelností NIST ohodnotila riziko této zranitelnosti nejvyšší možnou hodnotou 10. Zranitelnost byla přítomna v knihovně od roku 2013, ale útočníci ji začali zneužívat až v roce 2021. Nedlouho poté byla opravena, ale bezpečnostní výzkumníci zaznamenali v době největšího provozu až 100 Log4Shell útoků za minutu. [37]

SQL a NoSQL injekční útoky

Hrozby specifické pro databáze, které obsahují vložení libovolného SQL nebo non-SQL útočného řetězce do databázových dotazů, které obsluhují webové aplikace. Organizace, které nepraktikují bezpečnostní standardy při vývoji webových aplikací a neprovádějí pravidelné testování zranitelností, jsou náchylné k těmto útokům. [32]

Například útoky na společnosti jako J.C.Penney, 7-Eleven, JetBlue a Heartland Payment Systems, které probíhaly od roku 2005 do roku 2012, za kterými stála jedna skupina útočníků. V těchto útocích byla SQL injekce využita pro neoprávněný přístup do systémů, ze kterých bylo odcizeno a prodáno celkem 160 milionů záznamů o platebních kartách. [38]

Přetečení vyrovnávací paměti

Přetečení vyrovnávací paměti se objevuje, když se proces snaží zapsat více dat do datového bloku než je jeho velikost. Útočníci mohou využít přebytek dat, uložený ve vedlejších blocích, jako základ pro útok na systém. [32]

Příkladem takového napadení byl počítačový červ **SQL Slammer** z roku 2003, který využil přetečení paměti Microsoft SQL Serveru 2000 ke spuštění libovolného kódu. Červ napadl téměř všechny zranitelné systémy (přibližně 75 tisíc serverů) připojené k internetu během pár minut po spuštění. Využíval UDP pakety na portu 1434 a díky své rychlosti způsobil odepření služeb propojených sítí (DDoS). Jako

dočasné řešení situace byly použity filtry, které nepropouštěly UDP pakety pro port 1434. Později Microsoft opravil chybu, kterou červ použil pro přetečení paměti. [39]

Malware

Malware (malicious software) je software, který je vytvořen přímo pro zneužití zranitelností nebo způsobení škod databázím jiným způsobem. Vstupním bodem pro malware může být jakékoliv koncové zařízení, které se připojuje do databázové sítě. Malwaru je mnoho druhů, mimo jiné i následující:

- **Spyware** – malware, který sbírá citlivé informace, jako uživatelská jména, hesla, údaje platebních karet a osobní informace. Ty pak bez vědomí oběti odesílá útočníkovi.
  Například spyware **Pegasus**, jenž byl vytvořen pro odposlouchání a následné sbírání dat z mobilních telefonů. Funguje na většině operačních systémů Android, iOS, BlackBerry, Windows Phone a Symbian. Po instalaci na cílený telefon po sobě nezanechává žádnou viditelnou stopu. Slouží k sledování hovorů, zachytávání SMS zpráv, monitorování pozice, sbírání hesel a fotografií. Pro jeho instalaci na rozdíl od sociálního inženýrství není potřeba žádná akce od oběti [40].
- **Červy** – malware, který se replikuje a rozšiřuje do dalších aplikací, systémů a zařízení bez lidského zásahu. Mohou být použiti pro sestavení botnetu.
- **Ransomware** – malware, jenž zamezí přístup oběti k jejím datům nebo zařízení a požaduje výkupné za jejich zpřístupnění nebo nezveřejnění. Ransomwary v roce 2022 zastupovaly 17 % všech kybernetických útoků.
  Například ransomware **WannaCry**, který se v roce 2017 rozšířil na více než 200 tisíc zařízení mimo jiné zasáhl firmy jako FedEx, Honda a Nissan. **WannaCry** se rozšířil pomocí zneužití zranitelnosti EternalBlue, který vyvinula americká agentura NSA pro svou vlastní potřebu, ale ta byla ukradena a zveřejněna skupinou Shadow Brokers. EternalBlue fungoval na starých, neaktualizovaných verzích Microsoft Windows, kterých ale bylo stále dost na to aby se **WannaCry** rychle rozšířil [41].
- **Trojští koně** – malware, který je součástí legitimního softwaru nebo se alespoň tak tváří. Po jeho spuštění může vytvořit „zadní vrátka“ na zařízení nebo nainstalovat dodatečný malware. [42]

Útoky na zálohy

Organizace, které nedokážou ochránit zálohy se stejně striktními opatřeními, jaké se používají pro samotnou databázi, jsou náchylné na útoky na zálohy. K jejich rozšíření přispívají i tyto prvky:

- **Rostoucí objem dat** – zachycení, ukládání a zpracování dat exponenciálně roste napříč skoro všemi organizacemi. Jakékoliv bezpečnostní nástroje a postupy musí být dobře škálovatelné aby splnily potřeby v blízké i daleké budoucnosti.
- **Přísnější regulační požadavky** – celosvětové prostředí pro dodržování předpisů stále roste ve složitosti, což ztěžuje jejich dodržování.
- **Nedostatek odborníků v oblasti kybernetické bezpečnosti** – v roce 2023 čelilo celosvětově odvětví kybernetické bezpečnosti nedostatku až 4 milionů odborníků. [32, 43]

Útoky odepření služby

Při útoku odepření služby (Denial of Service – DoS) útočník zahltí cíl, v tomto případě databázový server tolika požadavky, že server nebude schopen obsloužit legitimní uživatele. Často se stane nestabilním nebo se zhroutí, viz 1.8. [32]

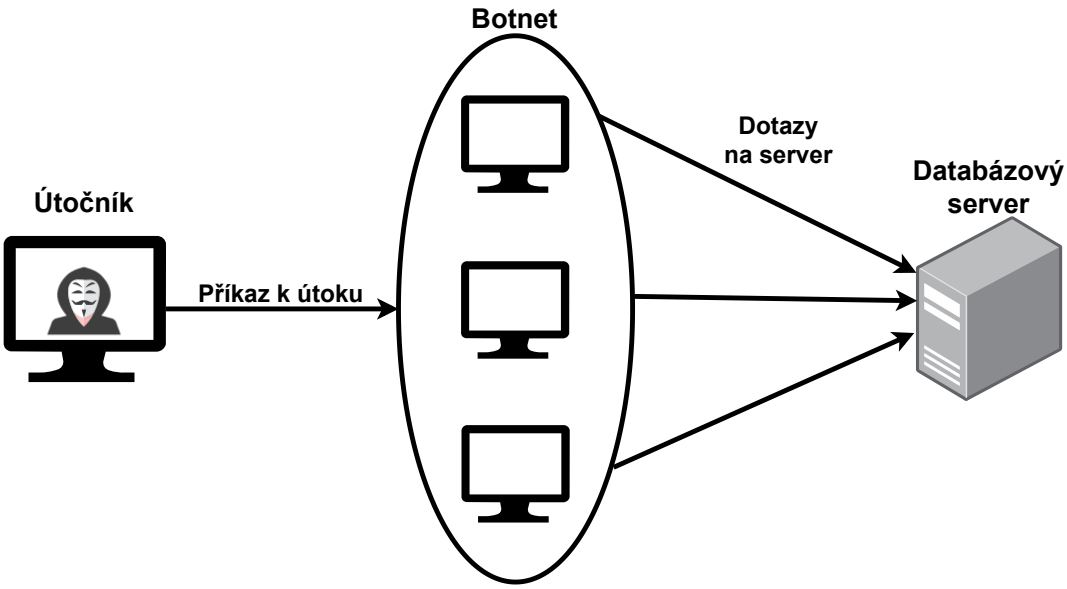
Obr. 1.8: Útok DoS na databázový server

Při distribuovaném útoku odepření služby (Distributed Denial of Service – DDoS) přichází zahlcení z více zdrojů (např. botnet), takže je útok těžší zastavit, viz 1.9. [32]

Například v roce 2019 použila Čína tzv. **Great Cannon** systém pro distribuovaný útok odepření služby na fórum, kde se obyvatelé Hong Kongu domlouvali na protestech [44]. **Great Cannon** je nástroj, který přebírá provoz na jednotlivé IP adresy (nebo z nich) a může libovolně nahrazovat nešifrovaný obsah škodlivým obsahem jako prostředník [45].

1.5 Nedostatky ve výchozím nastavení

Přestože jsou vynakládány obrovské finanční prostředky na ochranu dat různých prvků IT infrastruktury, mnohé z nich své vlastní úsilí zmaří tím, že tyto informace nakonec ukládají do špatně nakonfigurovaných databází. Ať už je to kvůli logistice



Obr. 1.9: Útok DDoS na databázový server

starších aplikací, nebo pohodlnosti správců, databáze nastavené pomocí výchozích nastavení jsou v rámci firem příliš časté. Výchozí konfigurace jsou snadnou kořistí pro dobře informované zloděje dat. Největší riziko představují následující výchozí nastavení:

- **Výchozí hesla a účty** – nejnebezpečnější a zároveň nejrozšířenější chyba v autentizaci je ponechání výchozích jmen a hesel pro správu databází. Rizikové je i nastavení konfigurací, které umožňují anonymní přihlášení.
- **Přímý přístup k tabulkám** – pokud aplikace může vytvářet vlastní příkazy typu `SELECT, UPDATE, INSERT, DELETE` a přímo přistupovat k tabulkám, jsou data zranitelná. Mezi nejlepší opatření patří přístup, kdy je aplikace naprogramována tak, že uživatel může přistupovat pouze k uloženým procedurám a má zakázaný přístup k tabulkám.
- **Zachování výchozích procedur** – výchozí uložené procedury mohou být v mnoha případech velkou zranitelností (procedura `xp_cmdshell` na Microsoft SQL serveru). Doporučení je zakázat nebo zcela odstranit výchozí uložené procedury.
- **Šifrovací klíče uložené s databází** – šifrování databáze může být účinnou vrstvou zabezpečení databáze, pokud je provedeno správně. Špatná konfigurace schémat jako například transparentní šifrování dat (Transparent Data Encryption – TDE), mohou vést k jejich prolomení. Při uložení klíče TDE vedle databáze je jako nalepení hesla na monitor. Doporučení je zabezpečit a ukládat šifrovací klíče na serveru, který není hostitelem databáze.
- **Nepotřebné aplikace a služby** – databáze jsou poskytovány s řadou pod-

půrných aplikací a služeb, které umožňují širokou skupinu funkcí. Avšak každý další nástroj může do databázového systému přinést další zranitelnosti. Doporučení je nepotřebné aplikace a služby zakázat nebo odinstalovat. [46]

1.6 Existující řešení

Před vývojem vlastního řešení je vhodné provést analýzu dostupných nástrojů, které se zaměřují na kontrolu MariaDB a MySQL databází.

1.6.1 MySQLTuner

MySQLTuner je skript napsaný v programovacím jazyku Perl, který umožňuje rychlou kontrolu a úpravu MySQL/MariaDB instalací ke zvýšení výkonu a stability. Skript získá aktuální konfigurační proměnné a stavová data pro jejich prezentaci ve stručném formátu a doporučí základní úpravy nastavení k optimalizaci výkonu. [47]

1.6.2 Percona

Percona je obsáhlá sada nástrojů pro správu a analýzu MySQL/MariaDB databází, s důrazem na výkon, replikaci a správu. Poskytuje možnosti automatické kontroly konfigurace a detekce potenciálních problémů.

Její výhody spočívají v širokých možnostech a flexibilitě, naopak nevýhodou je menší důraz na bezpečnostní aspekty a složitější interpretaci výstupů. [48]

2 Vývoj softwarového řešení

Tato kapotila se věnuje návrhu a implementaci softwarového nástroje pro automatickou kontrolu bezpečné konfigurace databází MySQL a MariaDB. Nástroj je vytvořen pomocí programovacího jazyka Python, který se nejčastěji využívá pro vývoj webových stránek, strojového učení a analýzu dat.

2.1 Funkce programu

Program se spouští z příkazové řádky například:

```
python .\main.py --setup-db --user <username>
--password <password> --host <host> --port <port>
```

Je možné zadat různé argumenty, viz 2.1.:

- **path** – cesta ke konfiguračnímu souboru databázového serveru. Pokud není zadán, tak se nastaví výchozí umístění podle operačního systému.
- **peth** – cesta ke spustitelnému souboru `mysql.exe` (i v MariaDB). Ve výchozím nastavení určeno pomocí funkce na základě operačního systému.
- **user** – uživatel, pod kterým se program pokouší přihlásit k databázovému serveru a databázím. Výchozí hodnota je nastavena na `root`.
- **password** – heslo pro přihlášení uživatele z argumentu `user`. Není nastavena žádná výchozí hodnota.
- **host** – adresa databázového serveru, na kterém se má provést testování. Výchozí hodnota nastavena na `localhost`.
- **port** – číslo portu, na kterém se nachází databázový server. Výchozí hodnota je `3306` (výchozí hodnota v MySQL i MariaDB).
- **dbname** – název databáze, ke které se chceme připojit. Není nastavena žádná výchozí hodnota.
- **name** – název výsledného PDF reportu. Není nastavena žádná výchozí hodnota, ale při nezadání tohoto argumentu se vytvoří název z adresy serveru, názvu databáze a čísla portu.
- **custom-latex-engine** – možnost nastavení vlastního překladače mezi LaTeX soubory a PDF. Výchozí hodnota je `pdflatex`.
- **language** – jazyk výsledného reportu. Implementována je jen výchozí možnost `en` (angličtina).
- **no-report** – argument, při jehož zadání se nevygeneruje výsledný PDF report.
- **setup-db** – argument, při kterém se vytvoří testovací databáze a uživatelé.

Tab. 2.1: Argumenty při spuštění programu

Název	**Flag**	**Typ**	**Výchozí hodnota**
Konfigurace	`-p, -path`	Cesta k souboru	Funkce
Spuštění	`-peth`	Cesta k souboru	Funkce
Uživatel	`-user`	String	root
Heslo	`-password`	String	×
Host	`-host`	String	localhost
Port	`-port`	String	3306
Název databáze	`-dbname`	String	×
Název reportu	`-name`	String	×
Latex engine	`-custom-latex -engine`	String	pdflatex
Jazyk	`-language`	Výběr (en/cz)	en
Bez repotu	`-no-report`	Akce	×
Vytvoř databázi	`-setup-db`	Akce	×

Zadané hodnoty argumentů zpracuje soubor `main.py`. Ten je poté předá instanci třídy `Session`, která řídí celý běh programu. Pokud je v argumentech vyžádáno vytvoření testovací databáze, tak se vytvoří pomocí souboru `experimental_db_setup.py` a připraví také uživatele a tabulky pomocí SQL příkazů v `create_db.sql`. Poté se program připojí k databázi pomocí zadaných přihlašovacích údajů, názvu databáze, portu a adresy serveru. Dále se pokusí o přečtení konfiguračního souboru, načtení oprávnění uživatelů na serveru a uloží je do paměti pro pozdější testování. Následně se provádí testy ze souboru `tests.py`, a to za předpokladu, že jsou splněny jejich požadavky a mají být testovány. Po dokončení posledního testu se výsledky testů předají do funkcí pro vytvoření úvodu, technických detailů, tabulky testů a samotných výsledků ze souboru `document_builder.py`. Nakonec se z LaTeX souborů (přípona .tex) vytvoří výsledný report ve formátu PDF a uzavře se databázové připojení. Průběhu programu je zobrazen v diagramu, viz 2.1.

2.2 Použité moduly a knihovny

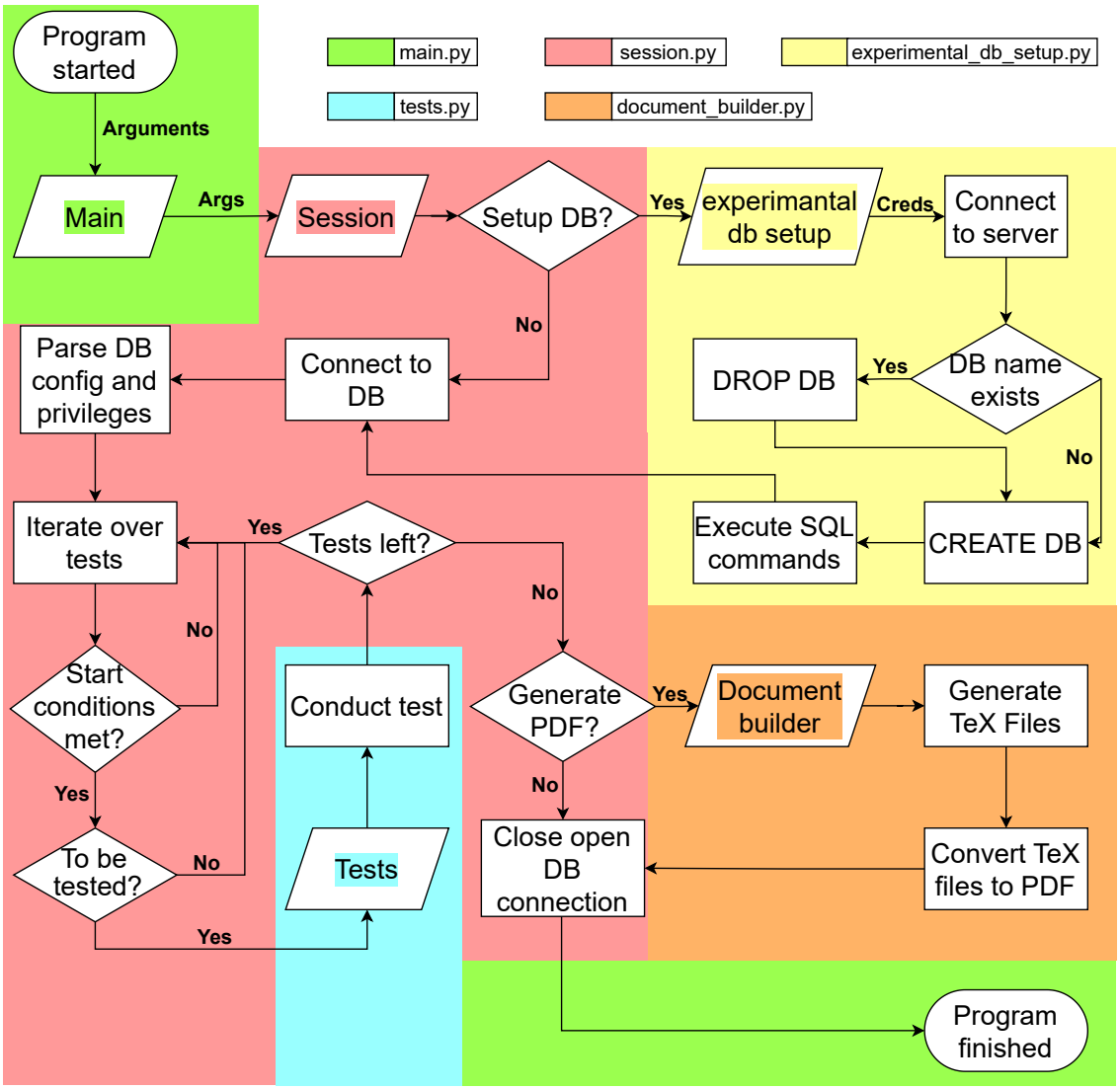
V této podkapitole jsou popsány moduly a knihovny, které jsou v programu použity. Některé jsou součástí běžné instalace Pythonu, jiné je potřeba doinstalovat. Jejich názvy se nacházejí v souboru `dependencies`. Na zařízeních s operačním systémem Windows je lze nainstalovat pomocí následujícího skriptu z příkazové řádky:

```
for /F %i in (dependencies) do pip install %i
```

Na zařízeních s operačním systémem Linux pomocí následujícího skriptu:

```
cat dependencies | xargs -n 1 pip install
```

- **Argparse** – usnadňuje psaní uživatelsky přívětivých rozhraní příkazového řádku. Program definuje, jaké argumenty požaduje a `argparse` zjistí, jak je analyzovat z `sys.argv` (seznam zadaných argumentů). Modul `argparse` také automaticky generuje nápovědu a hlášení o použití. Vypíše též chybu, pokud uživatel zadá neplatné argumenty.



Obr. 2.1: Funkce programu

- **Configparser** – poskytuje třídu `ConfigParser`, která implementuje základní konfigurační jazyk, jenž má podobnou strukturu jako konfigurační soubory systému Microsoft Windows s příponou `.ini`.
- **Logging** – definuje funkce a třídy, které implementují flexibilní systém logování událostí pro aplikace a knihovny. Lze ho využít pro pomoc při ladění programu, informace o aktuálním stavu průběhu programu a informace o výsledcích programu.
- **Matplotlib** – komplexní knihovna pro vytváření statických, animovaných a interaktivních vizualizací v jazyce Python. Podporuje grafy párových dat (bodové, spojové, sloupcové), statistických rozdělení (histogramy, koláčové, s odchylkami), trojrozměrných a objemových dat. [49]
- **Mysql-connector-python** – umožňuje komunikaci s databázovými systémy typu MySQL a MariaDB. Spravuje vytváření a ukončování spojení s databázemi, spouští příkazy SQL nad databází, získává a zpracovává data z výsledků dotazů.
- **Os** – poskytuje přenositelný způsob používání funkcí závislých na operačním systému. Lze ho využít pro čtení a zapisování souborů, manipulaci cest k souborům, vytváření dočasných souborů nebo adresářů a mnoho dalších funkcí.
- **Platform** – poskytuje informace o platformě, jako je operační systém, verze Pythonu, název a architekturu procesoru a síťový název počítače.
- **Pprint** – poskytuje možnost vypsání libovolných datových struktur jazyka Python ve formě, kterou lze použít jako vstup pro překladač. Pokud formátované struktury obsahují objekty, které nejsou základními typy jazyka Python, nemusí být vypsání možné.
- **Psutil** – multiplatformní knihovna pro získávání informací o běžících procesech a využití systému (procesor, paměť, disky, síť) v jazyce Python. Využívá se pro monitorování systému, profilování a omezení prostředků procesů a správu běžících procesů.
- **PyLaTeX** – knihovna jazyka Python, která slouží pro vytváření a kompilaci LaTeX souborů. Knihovna má dvě odlišná použití: generování plných PDF souborů a generování fragmentů LaTeX souborů. Generování plných PDF souborů je užitečné především v případě, když je veškerý text generován Pythonem, například při exportu dat z databáze. Fragmenty jsou pak užitečné v případě, že lze jen část vygenerovat automaticky, například napsání zprávy s několika matplotlib grafy. [50]
- **PyYAML** – knihovna pro práci s daty ve formátu YAML, který je navržený pro lidskou čitelnost, interakci se skriptovacími jazyky a používaný k serializaci dat.
- **Re** – modul pro práci s regulárními výrazy.

- **Subprocess** – modul umožňující spouštění procesů a komunikaci s nimi, například volání systémových příkazů.

2.3 MySQL Testy

V této části jsou popsány testy, které jsou implementovány v softwarovém řešení pro MySQL databáze. Výsledky testů jsou znázorněny v tabulce a grafu výsledného reportu. Pro tuto práci je implementováno 11 různých testů popsaných níže.

2.3.1 Encryption at transit

Testuje, zda uživatelé databáze vyžadují šifrování při přenosu dat, aby zajistili zabezpečenou komunikaci, která nemůže být odposlouchávána. Správně nastavené hodnoty `ssl_type` jsou `X509` a `ANY`. Dále se testuje, jestli je v nastavení serveru pro všechna připojení vynuceno použít zabezpečenou komunikaci pomocí `require secure transport`. Nakonec se kontroluje i hodnota proměnné `ssl_cipher`, jestli není prázdná. Pro testování je nutné připojení k testované databázi. Pokud alespoň jeden uživatel nevyžaduje šifrovanou komunikaci, vypíše se do tabulky výsledného reportu spolu s kontrolovanými proměnnými, viz 2.2.

Tab. 2.2: Příklad tabulek pro test Encryption at transit v MySQL

User	Host	SSL Type
public_user	%	×
test_user	%	×
root	localhost	×

Variable	Value
require_secure_transport	OFF
ssl_cipher	×

2.3.2 Encryption at rest

Testuje, zda jsou tabulkové prostory zašifrovány, aby se zabránilo neoprávněným přístupům k uloženým datům. K výsledku testu se vypíše tabulka s názvy tabulkových prostorů a jestli jsou zašifrované, nebo ne, viz 2.3. Pro testování je nutné připojení k testované databázi.

2.3.3 Insecure authentication plugins

Testuje, zda nejsou používány zastaralé nebo nevhodné metody na ukládání hesel uživatelů databáze. Test se zaměřuje na čtyři hlavní pluginy na ukládání hesel:

- **`mysql_native_password`** – používá se hashovací funkce založená na SHA1, která není považována za bezpečnou [51].

Tab. 2.3: Příklad tabulky pro test Encryption at rest v MySQL

Name	Space Type	Encryption
mysql	General	N
my_schema/public_info	Single	N
my_schema/private_info	Single	N
my_schema/secret_info	Single	Y

- **`mysql_old_password`** – používá se hashovací funkce, která není bezpečná a je poskytována hlavně pro zpětnou kompatibilitu [52].
- **`sha256_password`** – používá se SHA-256 pro hashování hesel a podporuje šifrování hesel při přenosu pomocí RSA [53].
- **`caching_sha2_password`** – používá se SHA-256 pro hashování hesel a pro lepší výkon využívá ukládání do mezipaměti na straně serveru. `Caching sha2 password` je výchozí autentizační plugin v novějších verzích MySQL a je považován za bezpečný. [54]

Je možné nastavit vlastní autentizační pluginy, ty ale mohou představovat bezpečnostní riziko v případě špatného nastavení. Výsledkem je tabulka v reportu, která uvádí, který plugin uživatelé využívají a zda je považován za bezpečný, viz 2.4.

Tab. 2.4: Příklad tabulky pro test Insecure authentication plugins v MySQL

User	Plugin	Security
private_user	caching_sha2_password	secure
public_user	mysql_native_password	insecure
test_user	caching_sha2_password	secure

2.3.4 Trust authentication

Tento test prověřuje, zda existují uživatelé databázového serveru, kteří se mohou přihlásit bez zadání hesla. Zaměřuje se na dva případy:

- **Uživatelé bez nastaveného hesla** – není potřebná znalost hesla pro přihlášení
- **Uživatelé využívající `auth_socket`** – autentizace probíhá na základě přihlášeného uživatele operačního systému, tedy pokud je pod stejným jménem i v tabulce uživatelů databáze [55]. Samotné použití `auth_socket` nemusí znamenat špatnou konfiguraci, ale když se útočník zmocní uživatelského účtu operačního systému, tak se může dostat k databázi bez použití hesla.

Jestliže se uživatel může přihlásit bez zadání hesla, zvyšuje se tím riziko neautorizovaného přístupu k databázi. Existuje-li alespoň jeden uživatel s nevhodným nastavením přihlášení k databázi, tak je k výstupu testu připojena tabulka uživatelů s popisem jejich prohřešku, viz 2.5.

Tab. 2.5: Příklad tabulky pro test Trust authentication v MySQL

User	Plugin	Security violation
public_user	auth_socket	insecure
test_user	caching_sha2_password	No password or NULL

2.3.5 Latest version of MySQL

Tento test kontroluje, zda je nainstalovaná verze MySQL serveru nejaktuálnější verzí MySQL serveru. Nejdříve se zjistí aktuální verze pomocí následujícího SQL dotazu:

```
SELECT VERSION();
```

Výsledek tohoto dotazu se poté porovná s nejnovější verzí. Ta se zjistí buď z dotazu na stránku MySQL Downloads [56], nebo se použije pevně zvolená hodnota (nejnovější verze v době vytváření testu byla 9.2.0).

Používání starších verzí představuje bezpečnostní rizika. Mohou disponovat zranitelnostmi, které byly opraveny v novějších verzích. Útočníci mohou tyto chyby zneužít k neoprávněnému přístupu k databázi, manipulaci s daty nebo k eskalaci oprávnění. Novější verze mohou mít lepší optimalizaci výkonu a opravené kritické chyby. Výstupem testu je textový popis výsledku porovnání nainstalované a nejnovější verze MySQL serveru.

2.3.6 Permissions test

Tento test vytvoří tabulku s přehledem uživatelů a jejich právy nad tabulkami v databázi. K získání oprávnění uživatelů se použije následující SQL dotaz nad tabulkou `table_privileges`, která obsahuje informace o přidělených právech.

```
SELECT grantee, table_schema, table_name, privilege_type
FROM information_schema.table_privileges
WHERE table_schema NOT IN ('mysql', 'information_schema',
'performance_schema', 'sys');
```

Test pomáhá odhalit, jestli nemá běžný uživatel přístup k tabulkám obsahující citlivé informace (např. osobní údaje nebo finanční transakce) a zda se dodržuje princip nejmenších oprávnění. Ten říká, že by každý uživatel měl mít pouze ta práva, která jsou nezbytná pro jeho práci. Výslednou tabulku, viz 2.6, je tedy potřeba zkontrolovat, zda oprávnění uživatelů odpovídá jejich pracovní náplni a nemají přebytečná oprávnění, potenciálně vedoucí k bezpečnostním rizikům.

Tab. 2.6: Příklad tabulky pro Permissions test v MySQL

User Type	**Table Schema**	**Table Name**	**Privilege Types**
'public_user'@'%'	my_schema	public_info	SELECT
'private_user'@'%'	my_schema	public_info	SELECT
'private_user'@'%'	my_schema	private_info	SELECT
'admin_user'@'%'	my_schema	secret_info	SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, ALTER

2.3.7 Loadable functions

Tento test ověřuje nastavení proměnné `local_infile` a vypisuje obsah tabulky `mysql.func`. Tím se zjišťuje, zda server umožňuje načítání souborů a zda jsou nahrány nějaké neoficiální funkce.

- **local_infile** – proměnná, která umožňuje uživatelům načítat soubory pomocí příkazů `LOAD DATA`. Může mít následující hodnoty:
  - **ON** – umožňuje uživatelům načítat soubory ze systému
  - **OFF** – zabraňuje použití příkazů `LOAD DATA`
- **mysql.func** – tabulky, která obsahuje záznamy o načtených funkcích (loadable functions) dříve známých také jako uživatelem definované funkce (user defined functions).

Proměnná `local_infile` by měla být nastavena na **OFF**, pokud není potřeba načítat soubory ze systému. Načtené funkce mohou představovat bezpečnostní riziko, protože umožňují přidávat neznámé funkce do databáze. Všechny načtené funkce jsou vypsány v tabulce, viz 2.7, k posouzení jejich nezbytnosti a pokud nejsou potřeba, měly by být odstraněny.

2.3.8 File system access

Tento test kontroluje, kam může MySQL server zapisovat a odkud může číst soubory v rámci operačního systému. To se provádí pomocí následujícího SQL dotazu:

Tab. 2.7: Příklad tabulky pro test Loadable functions v MySQL

Name	Ret	Dll	Type
my_custom_hash	0 (string)	C:\MySQL\plugins\my_hash.dll	0
malicious_backdoor	2 (real)	C:\MySQL\plugins\backdoor.dll	0

```
SELECT @@global.secure_file_priv;
```

Tato proměnná může nabývat tří hodnot:

- **Prázdná hodnota** – MySQL nemá omezený přístup a může číst/zapisovat kdekoliv v rámci oprávnění uživatele, pod kterým běží. To představuje bezpečnostní riziko, protože útočník by mohl potenciálně číst libovolné soubory a zapisovat na jakékoliv místo v systému.
- **Cesta ke složce** (např. C:\MySQL\Uploads) – MySQL může číst a zapisovat pouze do této složky. Bezpečné nastavení, které minimalizuje riziko neoprávněné manipulace se soubory mimo vyhrazenou složku.
- **NULL** – MySQL nemůže číst ani zapisovat žádné soubory. Toto je nejbezpečnější nastavení, pokud není potřeba pracovat s importem a exportem souborů. [57]

Součástí tohoto testu je i výpis uživatelů s oprávněním `FILE`, které jim umožňuje pracovat se soubory na úrovni databázového serveru. Tyto uživatele dostaneme pomocí SQL dotazu:

```
SELECT User, Host, File_priv
FROM mysql.user
WHERE File_priv = 'Y';
```

Díky výpisu uživatelů s tímto oprávněním, viz 2.8, můžeme zjistit, zda nemají oprávnění uživatelé, kteří by je mít neměli. Pokud se takoví uživatelé najdou, je doporučeno jim odebrat `FILE` oprávnění, aby se zamezilo úniku dat.

Tab. 2.8: Příklad tabulky pro test File system access v MySQL

User	Host	File_priv
public_user	%	Y
admin_user	localhost	Y
root	localhost	Y

2.3.9 Log configuration

Tento test kontroluje proměnné databázového serveru, které souvisí s logováním, s cílem zjistit, zda jsou správně nastaveny pro vysokou bezpečnost a optimální výkon. Logování může být užitečné z pohledu monitorování nebezpečné aktivity, ale pokud není správně nastaveno, tak představuje bezpečnostní riziko. Kontrolují se následující proměnné:

- **general_log** – obecný záznam toho, co MySQL server dělá. Server zaznamenává uživatele, kteří se připojují nebo odpojují a ukládá každý jejich SQL příkaz [58]. Může mít následující hodnoty:
  - **ON** – veškeré dotazy jsou zaznamenávány
  - **OFF** – zaznamenávání dotazu je vypnuto
- **log_raw** – proměnná udávající zda budou hesla zaznamenána v čitelném textu. Tuto volbu lze vynutit při spuštění serveru s volbou `--log-raw` [23, 58]. Může mít následující hodnoty:
  - **ON** – dotazy jsou zaznamenávány v prostém textu. Existuje možnost, že budou zaznamenány citlivé informace.
  - **OFF** – dotazy nezobrazí citlivé informace (např. hesla).
- **slow_query_log** – proměnná, která udává, jestli se budou vytvářet záznamy pomalých dotazů na databázi. Zaznamenávání těchto dotazů pomáhá k optimalizaci výkonu a zvýšení bezpečnosti databáze [59]. Může mít následující hodnoty:
  - **ON** – pomalé dotazy jsou zaznamenávány
  - **OFF** – databáze nezaznamenává pomalé dotazy
- **long_query_time** – proměnná udávající hranici v sekundách, kdy se začne dotaz registrovat jako pomalý. V základu je hodnota nastavena na 10 a může být změněna na hodnoty v rozmezí od nula do ekvivalentu 365 dní v sekundách. Při nastavení na nízkou hodnotu je více dotazů považováno za pomalé a může tím rychle narůst požadavek na místo pro ukládání záznamů. Při nastavení příliš vysoké hodnoty je možné, že se nebudou zaznamenávat všechny dlouhotrvající dotazy. [57]
- **innodb_strict_mode** – proměnná, která udává, jestli při kontrole neplatných nebo nekompatibilních nastavení tabulek v dotazech vrací chyby místo varování [60]. Může mít následující hodnoty:
  - **ON** – striktní režim je zapnut, zvyšuje bezpečnost a konzistenci dat
  - **OFF** – striktní režim je vypnut, může způsobit nekonzistenci dat

Stav těchto proměnných se získává z konfiguračního souboru, pokud se tam nachází. V opačném případě se zjistí z aktuálního stavu databázového serveru pomocí SQL dotazů. Výsledné hodnoty se zobrazí v tabulce výsledného reportu, viz 2.9.

Tab. 2.9: Příklad tabulky pro Log configuration v MySQL

Variable	Value
general_log	OFF
log_raw	ON
slow_query_log	ON
long_query_time	10
innodb_strict_mode	ON

2.3.10 Configuration of SSL

Tento test kontroluje, zda má databázový server nastavené proměnné `ssl_ca`, `ssl cert` a `ssl_key`, které jsou klíčové pro šifrované připojení mezi uživatelem a serverem. Hodnoty proměnných se získávají pomocí následujícího SQL dotazu:

```
SHOW VARIABLES
WHERE Variable_name
IN ('ssl_ca', 'ssl_cert', 'ssl_key');
```

- **ssl_ca** – obsahuje cestu k souboru certifikační autority ve formátu PEM, který má seznam důvěryhodných certifikačních autorit SSL
- **ssl_cert** – obsahuje cestu k certifikátu serveru
- **ssl_key** – obsahuje cestu k souboru se soukromým klíčem serveru

Hodnoty proměnných jsou vypsány v tabulce výsledného reportu, viz 2.10. Pokud jsou tyto proměnné správně nastaveny, tak spolu uživatel a server mohou šifrovaně komunikovat a server je ověřitelný uživateli. Pokud nejsou tyto proměnné nastaveny, nebo jsou nastaveny nesprávně, tak je spojení náchylné k odposlechu a MITM útokům.

Tab. 2.10: Příklad tabulky pro test Configuration of SSL

Variable	Value
ssl_ca	ca.pem
ssl_cert	server-cert.pem
ssl_key	server-key.pem

2.3.11 SUPER privileges

Tento test zjišťuje, kteří uživatelé databáze mají `SUPER` oprávnění. Toto oprávnění poskytuje rozsáhlé možnosti administrace a představuje bezpečnostní riziko, pokud

není přidělováno s opatrností. Uživatelé s tímto oprávněním jsou zaznamenáni v tabulce výsledného reportu, viz 2.11. Získat všechny uživatele s tímto oprávněním lze pomocí následujícího SQL dotazu:

```
SELECT User, Host, Super_priv
FROM mysql.user
WHERE Super_priv = 'Y';
```

Uživatel se `SUPER` oprávněním může mimo jiné měnit globální a systémové proměnné, ukončovat spojení jiných uživatelů, ignorovat `READ ONLY` režim, měnit bezpečnostní nastavení a obcházet je. Dnes se považuje `SUPER` oprávnění za zastaralé a předpokládá se jeho odstranění v některé z následujících verzí MySQL. Nahradit ho lze pomocí omezenějších dynamických oprávnění (např. `AUDIT_ADMIN`, `BACKUP_ADMIN`, `BINLOG_ADMIN`, `CONNECTION_ADMIN`, `ENCRYPTION_ KEY_ADMIN`, `FIREWALL_ADMIN`) [61].

Tab. 2.11: Příklad tabulky pro test SUPER privileges v MySQL

User	Host	Super_priv
admin_user	localhost	Y
root	localhost	Y

2.4 MariaDB Testy

V této části jsou popsány testy, které jsou implementovány v softwarovém řešení pro MariaDB databáze. Výsledky testů jsou znázorněny v tabulce a grafu výsledného reportu, viz B. Pro tuto práci je implementováno 11 různých testů popsaných níže.

2.4.1 Encryption at transit

Tento test kontroluje, jestli uživatelé musí při komunikaci se serverem používat zabezpečenou komunikaci, to lze zjistit ze sloupce `ssl_type` v tabulce **mysql.user**. Za vhodné hodnoty jsou považovány:

- **ANY** – uživatel musí použít TLS pro komunikaci se serverem, ale není požadován platný X509 certifikát (certifikát, který spojuje identitu s veřejným klíčem pomocí digitálního podpisu).
- **X509** – uživatel musí použít TLS pro komunikaci se serverem a musí mít platný X509 certifikát.

- **SPECIFIED** – uživatel musí použít TLS pro komunikaci se serverem a je přesněji zadán jeden z parametrů issuer (certifikační autorita), subject (předmět certifikátu) nebo cipher (šifrovací metoda). Při specifikaci pouze šifrovací metody není potřeba platný X509 certifikát, ale při ostatních dvou parametrech je vyžadován.

Nevyhovující hodnota je prázdný řetězec, který znamená, že uživatel může, ale nemusí použít TLS pro komunikaci se serverem. Uživatelé s nevyhovující hodnotou jsou zaznamenáni v tabulce výsledného reportu (prázdný řetězec nahrazen ×), viz 2.12. [62]

Dále test ověřuje hodnoty serverových proměnných `require_secure_transport` a `ssl_cipher`. `Ssl_cipher` obsahuje seznam povolených šifer pro TLS a za vhodnou hodnotu je považováno cokoliv kromě prázdného řetězce. `Require_secure_transport` udává, jestli je požadováno, aby všichni uživatelé využívali zabezpečenou komunikaci se serverem. Jedná se o proměnnou typu boolean, takže má jen stavy zapnuto (`ON`) a vypnuto (`OFF`). Za vhodnou hodnotu se považuje pouze `ON`, při kterém se pokus o připojení nezabezpečeně zamítne. Obě hodnoty jsou vypsány v tabulce výsledného reportu a u `ssl_cipher` je prázdný řetězec nahrazen ×, viz 2.12. [63, 64]

Tab. 2.12: Příklad tabulek pro test Encryption at transit v MariaDB

User	Host	SSL Type
public_user	%	×
test_user	%	×
root	localhost	×

Variable	Value
require_secure_transport	OFF
ssl_cipher	×

2.4.2 Encryption at rest

Tento test kontroluje, zda jsou tabulky používající úložný systém InnoDB (výchozí systém v MariaDB) šifrovány. Při použití šifrování jsou data šifrována při zápisu a dešifrována při čtení ze souborového systému. Kontroluje se výstup následujícího příkazu:

```
SELECT NAME, ENCRYPTION_SCHEME, CURRENT_KEY_ID
FROM INFORMATION_SCHEMA.INNODB_TABLESPACES_ENCRYPTION;
```

Sloupec `NAME` uvádí cestu k tabulce. Sloupec `ENCRYPTION_SCHEME` udává, jestli je tabulka šifrována. Vhodná hodnota je pouze 1 (tabulka zašifrována), nula značí, že tabulka šifrována není. Sloupec `KEY_ID` označuje identifikátor šifrovacího klíče. Pokud je alespoň jeden záznam v dotazované tabulce, tak se objeví ve výsledném reportu, viz 2.13. [65, 66]

Dále se kontrolují tři proměnné s předponou `innodb_encrypt`, konkrétně `tables`, `log` a `temporary_tables`. `Tables` udává, jestli je povoleno šifrování tabulek s nastavením tabulky `ENCRYPTED` na výchozí hodnotu (`DEFAULT`). `Log` udává, jestli se šifrují InnoDB redo logy (zajišťují spolehlivý zápis dat na disk). `Temporary_tables` udává, jestli jsou šifrovány i dočasné InnoDB tabulky. Všechny tři jsou typu boolean, ale `tables` má kromě možných hodnot zapnuto/vypnuto (`ON`/`OFF`) i třetí možnost vynutit (`FORCE`). Ta kromě povolení šifrování tabulek zakáže vytvářet nešifrované tabulky. U všech tří je vhodná hodnota `ON`. Hodnoty proměnných jsou vypsány v tabulce výsledného reportu, viz 2.13. [67]

Tab. 2.13: Příklad tabulek pro test Encryption at rest v MariaDB

Name	**Encryption scheme**	**Key ID**
db1/encrypted_table1	1	1
db1/encrypted_table2	0	NULL
db2/tab1	1	2

Variable	**Value**
innodb_encrypt_tables	OFF
innodb_encrypt_log	OFF
innodb_encrypt_temporary_tables	OFF

2.4.3 Insecure authentication plugins

Tento test kontroluje, jestli uživatelé nepoužívají zastaralé nebo nebezpečné metody při autentizaci. Porovnávají se hodnoty ve sloupci `plugin` v tabulce **mysql.user** se známými autentizačními metodami. Výsledek se zobrazí v tabulce výsledného reportu, viz 2.14. Podle bezpečnosti jsou metody rozděleny do následujících kategorií:

- **Secure** – autentizační metody jsou považovány za bezpečné
  - **ed25519** – metoda používající DSA nad eliptickými křivkami, vytvořen kvůli ztracení důvěry v bezpečnost metody `mysql_native_password` [68].
  - **gssapi** – metoda umožňující uživateli se autentizovat pomocí služeb používající GSSAPI (Generic Security Services Application Program Interface). V operačním systému Windows tato metoda podporuje Kerberos a NTLM autentizaci. V Unixových operačních systémech je nejčastěji používanou službou taky Kerberos. Nejčastěji se tato metoda používá při autentizaci Microsoft Active Directory. [69]
- **Warning** – autentizační metody jsou považovány za bezpečné, ale existuje riziko jejich zneužití

  - **unix_socket** – metoda umožňující uživateli použít přihlašovací údaje Unixového operačního systému. Výhodami jsou dobrá odolnost proti vzdálenému přístupu a útoku hrubou silou. Nevýhodou je, že při nesprávném nastavení bezpečnostní politiky operačního systému jsou ohrožena i data v databázi. Pokud se někdo dostane k přihlášenému účtu, má pak přístup do databáze bez nutnosti znát heslo. [70]
  - **named_pipe** – metoda umožňující uživateli použít přihlašovací údaje operačního systému Windows. Obdoba `unix_socket` pro Windows [71].
- **Insecure** – autentizační metody jsou zastaralé nebo se nepovažují za bezpečné
  - **mysql_old_password** – metoda používá hashovací funkci, která není bezpečná a je poskytována hlavně pro zpětnou kompatibilitu [52].
  - **mysql_native_password** – metoda používá hashovací funkci založenou na SHA1, která není považována za bezpečnou [51].
- **Unknown** – je použita neznámá autentizační metoda

Tab. 2.14: Příklad tabulky pro test Insecure authentication plugins v MariaDB

User	Plugin	Security
private_user	caching_sha2_password	secure
public_user	mysql_native_password	insecure
test_user	named_pipe	warning

2.4.4 Trust authentication

Tento test kontroluje, jestli se mohou uživatelé přihlásit k databázovému serveru bez zadání hesla. Zaměřuje se na dva případy:

- **Žádné heslo** – uživatelský účet má prázdné heslo (nebo NULL), při přihlašování stačí zadat prázdný řetězec. Tyto účty se zjistí pomocí následujícího dotazu:

```
SELECT user, host, plugin, authentication_string
FROM mysql.user
WHERE (authentication_string = ''
OR authentication_string IS NULL);
```

- **Named pipe/unix socket** – pro přihlášení do databáze stačí být přihlášen do operačního systému, což představuje bezpečnostní riziko v případě nesprávně nastavené bezpečnostní politiky nebo v situaci, kdy se neoprávněná osoba dostane k přihlášenému systému.

Pokud jsou objeveny tyto případy, tak se zobrazí v tabulce výsledného reportu, viz 2.15.

Tab. 2.15: Příklad tabulky pro test Trust authentication v MariaDB

User	Plugin	Security violation
public_user	auth_socket	warning
test_user	caching_sha2_password	No password or NULL

2.4.5 Latest version of MariaDB

Tento test kontroluje, jestli je nainstalovaná verze MariaDB serveru nejaktuálnější dostupnou verzí. Nainstalovaná verze se zjistí pomocí následujícího SQL dotazu:

```
SELECT VERSION();
```

Pokud se nepodaří takto získat aktuální verzi, tak se test dotáže pomocí příkazové řádky pomocí `mariadb --version`.

Aktuální verze MariaDB serveru se zjistí buď z dotazu na oficiální stránku Download MariaDB [72], nebo se použije pevně zvolená hodnota (nejnovější verze v době vytváření testu – 11.7.2).

Používání zastaralé verze databázového serveru představuje bezpečnostní rizika. Starší verze mohou disponovat bezpečnostními zranitelnostmi, které byly opraveny v novějších verzích. Útočníci mohou tyto chyby zneužít k neoprávněnému přístupu k databázi, manipulaci s daty nebo eskalaci oprávnění. Výstupem testu je text popisující výsledky porovnání nainstalované a nejnovější verze MariaDB serveru.

2.4.6 Permissions test

Tento test vytvoří tabulku s přehledem uživatelů a jejich právy nad tabulkami v databázi. K získání oprávnění uživatelů se použije následující SQL dotaz nad tabulkou `table_privileges`, která obsahuje informace o přidělených právech.

```
SELECT grantee, table_schema, table_name, privilege_type
FROM information_schema.table_privileges
WHERE table_schema NOT IN
('mysql', 'information_schema', 'performance_schema', 'sys');
```

Sloupec `grantee` obsahuje uživatele, který má dané oprávnění. `Table_schema` je sloupec s názvem databáze, ve kterém se nachází tabulka, ke které má uživatel

oprávnění. `Table_name` obsahuje samotný název tabulky a `privilege_type` je typ oprávnění, které má uživatel nad danou tabulkou.

Test pomáhá odhalit, jestli nemá běžný uživatel přístup k tabulkám obsahujícím citlivé informace (např. osobní údaje nebo finanční transakce) a jestli se dodržuje princip nejmenších oprávnění, který říká, že by každý uživatel měl mít pouze ta práva, která jsou nezbytná pro jeho činnost. Výslednou tabulku, viz 2.16, je potřeba manuálně zkontrolovat, zda oprávnění uživatelů odpovídají jejich činnostem a nemají přebytečná oprávnění, která by mohla vést k bezpečnostním rizikům.

Tab. 2.16: Příklad tabulky pro Permissions test v MariaDB

User Type	Table Schema	Table Name	Privilege Types
'public_user'@'%'	db1	public_info	SELECT
'private_user'@'%'	db1	public_info	SELECT, INSERT
'private_user'@'%'	db2	private_info	SELECT, INSERT
'admin_user'@'%'	secret_db	secret_info	SELECT, INSERT, UPDATE

2.4.7 User defined functions

Tento test kontroluje, zda jsou na databázový server přidány uživatelem definované funkce (UDF – User-Defined Functions) a kdo má oprávnění je vytvářet nebo odstraňovat. Všechny UDF jsou zaznamenány v tabulce `mysql.func`, kterou lze získat následujícím příkazem:

```
SELECT * FROM mysql.func;
```

Tabulka má čtyři sloupce: `Name` (název funkce), `Ret` (typ návratové hodnoty), `Dl` (název sdílené knihovny, odkud se má funkce načíst) a `Type` (`function` nebo `aggregate`) [73].

Pro vytvoření a smazání UDF je potřeba mít případné oprávnění nad `mysql` databází [74]. Uživatele s těmito oprávněními lze získat pomocí následujícího dotazu:

```
SELECT Grantee, Table_schema, Privilege_type
FROM information_schema.schema_privileges
WHERE Table_schema = 'mysql'
AND Privilege_type IN ('INSERT', 'DELETE');
```

Výsledky obou dotazů se zobrazí v tabulkách ve výsledném reportu, viz 2.17, pokud mají alespoň jeden záznam. Seznam UDF funkcí je třeba manuálně zkontrolovat, jestli se v něm nenachází neznámé funkce, které by mohly představovat bezpečnostní riziko. I seznam uživatelů je třeba zkontrolovat, jestli mají oprávnění nad `mysql` databází jen nezbytně nutní uživatelé.

Tab. 2.17: Příklad tabulek pro test User Defined Functions v MariaDB

Name	Library name	Type
spider_direct_sql	ha_spider.so	2
spider_columns	ha_spider.so	0
spider_copy_tables	ha_spider.so	2

Grantee	Table schema	Privileges
'admin_user'@'localhost'	mysql	INSERT
'root'@'localhost'	mysql	INSERT, DELETE

2.4.8 File system access

Tento test kontroluje, kam mají uživatelé s `FILE` oprávněním přístup, kteří uživatelé mají toto oprávnění a jestli mohou načítat data z lokálního zařízení.

Proměnná `secure_file_priv` udává, odkud ze souborového systému se mohou načítat data do databáze. Pokud je prázdná, je možné načítat data odkudkoliv ze systému [63]. Vhodné nastavení je cesta k určité složce, ze které chceme načítat data.

Proměnná `local_infile` je typu boolean, má tedy jen dva stavy zapnuto (`ON`) a vypnuto (`OFF`). Udává, jestli uživatel může načítat data ze svého systému do databáze [63]. Vhodné nastavení je vypnuto, aby se předešlo bezpečnostnímu riziku.

Pro zjištění uživatelů s `FILE` oprávněním lze použít dotaz:

```
SELECT User, Host, File_priv
FROM mysql.user
WHERE File_priv = 'Y';
```

Pokud má alespoň jeden uživatel toto oprávnění, tak se objeví v tabulce ve výsledném reportu, viz 2.18. Tato tabulka by měla být manuálně zkontrolována a zjištěná nadbytečná oprávnění by měla být odebrána, aby se zamezilo úniku citlivých informací.

Tab. 2.18: Příklad tabulek pro test File system access v MariaDB

Variable	Value
secure_file_priv	C:\MariaDB 11.7\data\
local_infile	OFF

User	Host	File_priv
public_user	%	Y
admin_user	localhost	Y
root	localhost	Y

2.4.9 Log configuration

Tento test kontroluje, jak je nastaveno logování MariaDB serveru. Kontrola probíhá zjištěním hodnot systémových proměnných serveru pomocí následujícího příkazu:

```
SHOW VARIABLES LIKE 'název_proměnné';
```

Proměnná `general_log` udává, jestli se mají zaznamenávat všechny SQL dotazy, přihlašování a odhlašování uživatelů. Je to proměnná, která má pouze stavy vypnuto/zapnuto [75]. Vhodná hodnota je vypnuto, protože na rozdíl od MySQL zde není možnost skrýt hesla v general logu. To by mohlo představovat bezpečnostní riziko, pokud má útočník přístup k logům.

Proměnná `slow_query_log` udává, jestli se zaznamenávají pomalé dotazy na databázi. Tato proměnná je typu boolean a vhodný stav je zapnuto, což pomáhá při ladění a optimalizaci výkonu databázového serveru [76]. Časový interval, po kterém se pomalý dotaz zaznamená, udává proměnná `long_query_time`. Zadané číslo představuje sekundy s přesností až na mikrosekundy [63]. Vhodnou hranici je těžké určit, protože závisí na velikosti databáze, co se považuje za pomalý dotaz. Pro účely této práce byla zvolena hranice deseti sekund. Pokud by byl časový interval příliš krátký, tak by se zaznamenal skoro každý dotaz a rychle by došlo místo na záznamy. Pokud by byl příliš dlouhý, tak nezaznamená žádný dotaz a byl by zbytečný.

Proměnná `log_bin` udává, jestli bude probíhat binární logování. To zaznamenává všechny změny v databázi a jak dlouho trvalo jejich vykonání. Dotazy typu `CREATE`, `ALTER`, `INSERT`, `UPDATE` a `DELETE` budou zaznamenány, ale dotazy typu `SELECT` a `SHOW` se nezaznamenají. Binární logování umožňuje replikaci dotazů a pomáhá při zálohování. Tyto logy mohou obsahovat citlivé informace i hesla, proto se musí chránit [77]. K tomu slouží proměnná `encrypt_binlog`, která udává, jestli budou logy zašifrovány [78]. Obě proměnné mají pouze stavy vypnuto/zapnuto a vhodná konfigurace je jen u obou zapnuto, protože binární logování je velice užitečné pro zjišťování změn v databázi, ale bez šifrování představuje bezpečnostní riziko.

Všechny proměnné jsou zaznamenány v tabulce výsledného reportu, viz 2.19.

Tab. 2.19: Příklad tabulky pro Log configuration v MariaDB

Variable	Value
general_log	OFF
slow_query_log	ON
long_query_time	10
log_bin	ON
encrypt_binlog	ON

2.4.10 Configuration of SSL

Tento test kontroluje, jestli server podporuje TLS a jestli jsou nastaveny základní parametry TLS přenosu. Kontrola probíhá zjištěním hodnot systémových proměnných serveru pomocí následujícího příkazu:

```
SHOW VARIABLES LIKE 'název_proměnné';
```

Proměnná `have_ssl` informuje, jestli server podporuje TLS pro bezpečné připojení uživatelů. Může nabýt tří následujících hodnot:

- **YES** – server podporuje TLS a je povoleno. Jediná vhodná konfigurace, protože využití TLS zvyšuje zabezpečení přenášených dat.
- **DISABLED** – server podporuje TLS, ale není povoleno.
- **NO** – server není zkompilován s TLS podporou, TLS tudíž nemůže být povoleno.

Základní parametry pro úspěšné TLS spojení jsou:

- **`ssl_ca`** – řetězec cesty k souboru s jedním nebo více X509 certifikáty důvěryhodných certifikačních autorit.
- **`ssl_cert`** – řetězec cesty k souboru s X509 certifikátem serveru.
- **`ssl_key`** – řetězec cesty k souboru se soukromým klíčem serveru [64].

Pokud tyto proměnné obsahují cestu k souboru, jsou považovány za správně nakonfigurované. Všechny proměnné jsou zobrazeny v tabulce výsledného reportu, viz 2.20.

2.4.11 SUPER privileges

Tento test zobrazuje uživatele, kteří mají `SUPER` oprávnění. Toto oprávnění poskytuje mimo jiné možnost odpojovat ostatní uživatele, nastavovat globální systémové

Tab. 2.20: Příklad tabulky pro Configuration of SSL v MariaDB

Variable	Value
have_ssl	YES
ssl_ca	/etc/mariadb/ssl/ca-cert.pem
ssl_cert	/etc/mariadb/ssl/server-cert.pem
ssl_key	/etc/mariadb/ssl/server-key.pem

proměnné, zapínat a vypínat logování i obcházení limitu `MAX_CONNECTIONS`. Výslednou tabulku uživatelů s tímto oprávněním, viz 2.21, lze získat pomocí následujícího příkazu:

```
SELECT User, Host, Super_priv
FROM mysql.user
WHERE Super_priv = 'Y';
```

V novějších verzích MariaDB serveru už nejsou oprávnění `SET USER`, `CONNECTION ADMIN`, `BINLOG ADMIN`, `REPLICA MONITOR`, `READ_ONLY ADMIN` a další součástí `SUPER` oprávnění a musí být přiděleny zvlášť. [79]

Protože toto oprávnění uděluje uživateli administrativní privilegia nad celým serverem, je potřeba manuálně zkontrolovat seznam uživatelů se `SUPER` oprávněním a odebrat ho uživatelům, pro které není nezbytné.

Tab. 2.21: Příklad tabulky pro test SUPER privileges v MariaDB

User	Host	Super_priv
admin_user	localhost	Y
root	localhost	Y

3 Experimentální testování

V této kapitole je popsána jak příprava testovacího prostředí, tak i rizika špatných konfigurací. Také jsou zde popsány limitace testů a cesty pro případný další rozvoj testovacích programů.

3.1 Příprava prostředí

Pro experimentální testování vytvořeného programu bylo vytvořeno prostředí pomocí následujících kroků:

- **Virtuální prostředí** – nainstalování čistého obrazu operačního systému Windows 10 pomocí virtualizačního softwaru Virtualbox.
- **Python** – stáhnutí a nainstalování aktuální verze Pythonu (3.13.3) z oficiálních stránek.
- **MariaDB server** – stáhnutí a nainstalování aktuální verze MariaDB serveru (11.7.2) z oficiálních stránek.
- **MikTeX** – stáhnutí a nainstalování aktuální verze správce TeX balíčků MikTeX (nainstaluje pouze potřebné balíčky) z oficiálních stránek.
- **GitLab** – stáhnutí aktuální verze testovaného programu z GitLabu.
- **Závislosti** – instalace pip balíčků a knihoven potřebných pro chod programu ze souboru `dependencies` pomocí:

```
for /F %i in (dependencies) do pip install %i
```

3.2 Testování

V této části jsou popsána úskalí a možné zneužití špatných konfigurací databázového serveru MariaDB. Jelikož si jsou MariaDB a MySQL servery velice podobné, tak jsou zde z důvodu snížení redundance uvedeny pouze MariaDB testy. U MySQL testů by testování vypadalo podobně. Největší rozdíl by byl nejspíše u konfigurace logů. MySQL má možnost skrýt citlivé informace pomocí nastavení proměnné `log_raw` a nevystavuje citlivé informace nebezpečí úniku dat.

3.2.1 Encryption at transit

Pokud uživatel nemá nastaveno `ssl_type` (vyžaduje bezpečnou komunikaci se serverem) může jeho komunikaci zachytit útočník a získat tím například jméno uživatele, hash hesla a typ pluginu pro získání hashe z hesla, viz 3.1. Je možné také zachytit dotazy uživatele na databázový server a odpovědi na ně, viz 3.2.

```
Login Request
  Username: root
  Password: 8c1b06a64df0f2b0ca65326d37b07fc7ff09343d
  Client Auth Plugin: mysql_native_password
```

Obr. 3.1: Zachycené přihlášení při komunikaci bez SSL/TLS

```
MySQL Protocol
  Request Command Query
    Statement: SELECT User, Password FROM mysql.user
MySQL Protocol - row packet
  text
    text: admin_user
  text
    text: *2470C0C06DEE42FD1618BB99005ADCA2EC9D1E19
```

Obr. 3.2: Zachycený dotaz a odpověď na něj při komunikaci bez SSL/TLS

Když uživatel vyžaduje bezpečné připojení a někdo se pokusí o nezabezpečenou komunikaci, tak server odmítne připojení i při zadání správných přihlašovacích údajů. Ta samá situace nastane, pokud server požaduje zabezpečenou komunikaci (pomocí proměnné `require secure transport`) a uživatel se snaží přihlásit bez SSL/TLS, viz 3.3.

```
ERROR 1045 (28000): Access denied for user 'tester'@'localhost'
 (using password: YES)
ERROR 3159 (08004): Connections using insecure transport
 are prohibited while --require_secure_transport=ON.
```

Obr. 3.3: Chybové zprávy při pokusu o přihlášení bez SSL/TLS

Jestliže je databázový server správně nastaven, tak se sestaví TLS spojení a komunikace mezi klientem a serverem je nečitelná pro kohokoliv, kdo ji zachytí, viz 3.4.

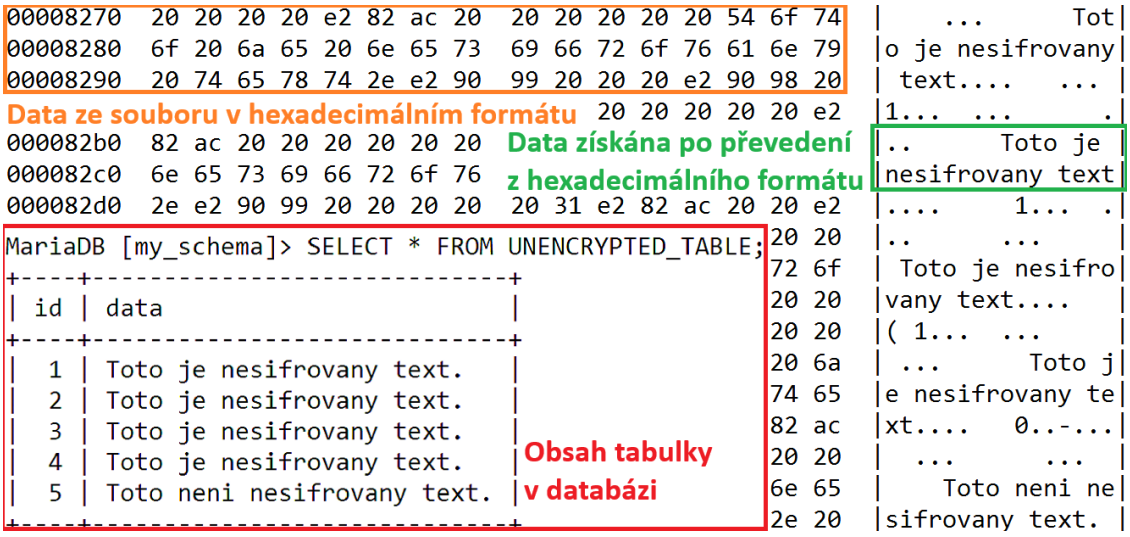
3.2.2 Encryption at rest

Když není nastaveno šifrování InnoDB tabulek, tak kdokoli, kdo má přístup k serveru, může přečíst obsah tabulek. Obsah tabulek je uložen v souborech s příponou `.ibd`, které po převedení na hexadecimální formát lze přečíst, pokud tabulky nebyly šifrovány, viz 3.5.

Při šifrování tabulek nelze nic vyčíst z původního `.ibd` souboru ani z hexadecimálního formátu, viz 3.6.

No.	Time	Source	Destination	Protocol	Lengt	Info
4	0.001037	::1	::1	MySQL	150	Server Greeting proto=10 version=11.7.2-MariaDB
5	0.001120	::1	::1	TCP	64	49818 → 3306 [ACK] Seq=1 Ack=87 Win=2618880 Len=0
6	0.001844	::1	::1	MySQL	100	Login Request user=
7	0.001907	::1	::1	TCP	64	3306 → 49818 [ACK] Seq=87 Ack=37 Win=2618880 Len=0
8	0.004480	::1	::1	TLSv1.2	235	Client Hello (SNI=localhost)
9	0.004553	::1	::1	TCP	64	3306 → 49818 [ACK] Seq=87 Ack=208 Win=2618624 Len=0
10	0.004693	::1	::1	TLSv1.2	160	Server Hello
11	0.004707	::1	::1	TCP	64	49818 → 3306 [ACK] Seq=208 Ack=183 Win=2618624 Len=0
12	0.004738	::1	::1	TLSv1.2	1293	Certificate
13	0.004749	::1	::1	TCP	64	49818 → 3306 [ACK] Seq=208 Ack=1412 Win=2617344 Len=0
14	0.010405	::1	::1	TLSv1.2	658	Server Key Exchange
15	0.010497	::1	::1	TCP	64	49818 → 3306 [ACK] Seq=208 Ack=2006 Win=2616832 Len=0
16	0.010522	::1	::1	TLSv1.2	111	Certificate Request
17	0.010527	::1	::1	TCP	64	49818 → 3306 [ACK] Seq=208 Ack=2053 Win=2616832 Len=0
18	0.010535	::1	::1	TLSv1.2	73	Server Hello Done
19	0.010538	::1	::1	TCP	64	49818 → 3306 [ACK] Seq=208 Ack=2062 Win=2616832 Len=0
20	0.011933	::1	::1	TLSv1.2	197	Certificate, Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
21	0.012000	::1	::1	TCP	64	3306 → 49818 [ACK] Seq=2062 Ack=341 Win=2618624 Len=0
22	0.012694	::1	::1	TLSv1.2	70	Change Cipher Spec
23	0.012769	::1	::1	TCP	64	49818 → 3306 [ACK] Seq=341 Ack=2068 Win=2616832 Len=0
24	0.012813	::1	::1	TLSv1.2	109	Encrypted Handshake Message
25	0.012820	::1	::1	TCP	64	49818 → 3306 [ACK] Seq=341 Ack=2113 Win=2616832 Len=0
26	0.029839	::1	::1	TLSv1.2	294	Application Data

Obr. 3.4: Zachycené přihlášení při komunikaci s SSL/TLS



Obr. 3.5: Čitelná data při nešifrování tabulek

3.2.3 Insecure authentication plugins

Při použití zastaralého autentizačního pluginu (např. `mysql native password`) pro přihlašování uživatele lze zachytit hash hesla (bez použití SSL/TLS), který lze prolomit pomocí útoků hrubou silou nebo slovníkovým útokem, viz 3.7.

Při použití SSL/TLS je možné použít útok hrubou silou pro možnost přihlášení přímo na server. Když se použijí bezpečné autentizační pluginy, odposlech i útok hrubou silou jsou prakticky nepoužitelné.

```
00008270  a4 c2 a4 c3 a1 27 e2 80  a6 c3 9f 4c c4 99 70 e2  |.....'.....L..p.|
00008280  84 a2 c4 8f e2 90 9b c2  b4 34 cb 87 55 e2 80 a2  |.........4..U...|
00008290  3e e2 90 86 e2 80 99 3a  20 c2 ae 43 40 c5 a5 c5  |>......: ..C@...|
000082a0  bb c2 ab 76 c5 81 e2 80  94 c5 9f c4 8c e2 90 88  |...v............|
000082b0  e2 80 a6 c4 8d 3f 34 c3  8e 6a c4 9b 6e c3 89 c3  |.....?4..j..n...|
000082c0  b6 3a 75 c4 99 49 c2 ad  e2 90 88 cb 99 c5 9f c5  |.:u..I..........|
000082d0  bb e2 90 81 e2 90 9a 78  c4 bd 73 2a 0a 26 e2 90  |.......x..s*.&..|
000082e0  9f 51 e2 80 b9 36 78 c5  bb e2 90 83 78 e2 90 96  |.Q...6x.....x...|
000082f0  c3 93 c4 82 c5 a4 e2 90  9c c4 84 68 26 74 cb 98  |...........h&t..|
00008300  c3 81 c5 a3 0a c4 91 c4  8d e2 90 95 24 4c e2 90  |............$L..|
00008310  87 61 66 c5 bb 42 c5 bb  4f 46 c5 be ef bf bd c5  |.af..B..OF......|
00008320  a2 c2 a4 c3 b4 c4 98 e2  90 95 c4 98 63 6a 79 c5  |............cjy.|
00008330  a0 ef bf bd 30 c3 8b 65  5c e2 82 ac 65 c5 99 c5  |....0..e\...e...|
00008340  87 e2 90 92 65 ef bf bd  e2 90 95 78 34 5c e2 80  |....e......x4\..|
00008350  ba 52 c4 84 3d c3 93 e2  90 81 e2 90 82 23 62 20  |.R..=........#b |
00008360  7d c5 bb c5 9e 30 c4 91  c5 95 e2 90 9b 20 c2 b7  |}....0....... ..|
00008370  e2 80 a1 48 c3 a4 61 c5  be 5e e2 90 81 c4 90 e2  |...H..a..^......|
```

Obr. 3.6: Nečitelná data při šifrování tabulek

```
+------------+----------------------+-------------------------------------------+
| User       | plugin               | authentication_string                     |
+------------+----------------------+-------------------------------------------+
| admin_user | mysql_native_password | *2470C0C06DEE42FD1618BB99005ADCA2EC9D1E19 |
+------------+----------------------+-------------------------------------------+
Dictionary cache built:
* Filename..: /usr/share/wordlists/rockyou.txt
* Passwords.: 14344391
* Bytes.....: 139921497
* Keyspace..: 14344384
* Runtime...: 2 secs

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 300 (MySQL4.1/MySQL5)
Hash.Target......: 2470c0c06dee42fd1618bb99005adca2ec9d1e19
2470c0c06dee42fd1618bb99005adca2ec9d1e19 password
```

Obr. 3.7: Získání hesla pomocí slovníkového útoku

3.2.4 Trust authentication

Pokud nemá uživatelský účet nastaveno žádné heslo, je možné se do něj přihlásit jen pomocí toho, že nebude žádné heslo zadáno, viz 3.8. To představuje velké bezpečnostní riziko, protože útočníkovi stačí pro přístup k databázovému serveru znát pouze jméno uživatele.

3.2.5 Latest version of MariaDB

Pro tento test byla vytvořena záloha pomocí následujícího příkazu:

```
MariaDB [(none)]> SELECT user, password from mysql.user;
+--------------+-------------------------------------------+
| User         | Password                                  |
+--------------+-------------------------------------------+
| public_user  |                                           |
+--------------+-------------------------------------------+
MariaDB [(none)]> exit
C:\Users\Tester>mariadb -u test_user
Welcome to the MariaDB monitor.  Commands end with ; or \g.
Your MariaDB connection id is 8
Server version: 11.7.2-MariaDB mariadb.org binary distribution

Copyright (c) 2000, 2018, Oracle, MariaDB Corporation Ab and others.
```

Obr. 3.8: Přihlášení k účtu bez hesla

```
mariadb-dump -u root -p -x -A > C:\Users\Tester\backup.sql
```

Potom byla odinstalována aktuální verze serveru MariaDB a nainstalována starší verze serveru MariaDB (10.6.22), do které se nahrálo nastavení a uživatelé ze zálohy pomocí:

```
mariadb --user root --password < C:\Users\Tester\backup.sql
```

Ve výsledném reportu bylo zaznamenáno, že není použita nejnovější verze MariaDB serveru. Zároveň bylo upozorněno na možnost zvýšeného rizika útoku kvůli zastaralé verzi serveru, viz 3.9. Po dokončení testu byla zpátky nainstalována aktuální verze serveru (11.7.2).

3.5 Latest version of MariaDB

Description This test verifies whether the database uses the latest software version. Outdated versions could contain security vulnerabilities that could be used by an attacker to compromise the database.

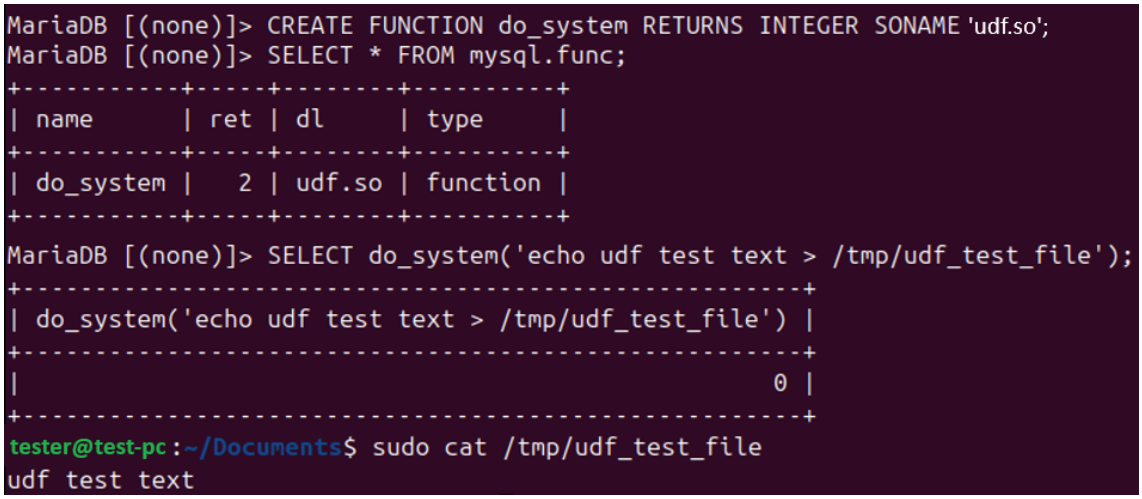
Database server uses outdated version of MariaDB 10.6.22 instead of latest version 11.7.2

Obr. 3.9: Upozornění ve výsledném reportu

3.2.6 User defined functions

Pro toto testování byl MariaDB server nainstalován na Ubuntu. Byla přidána kompromitovaná UDF (Uživatelem definovaná funkce) `do_system`. Díky ní bylo možno

spouštět příkazy v operačním systému zařízení, na kterém běží databázový server. Pro demonstraci byl jen vytvořen soubor (`udf_test_file`) a zapsán do něj text (udf test text), viz 3.10.



Obr. 3.10: Zneužití uživatelem definované funkce (UDF)

3.2.7 File system access

Pro tento test byl do složky `C:/Windows/System32` vložen testovací soubor `Super-Secrets.txt`. Při nastavené proměnné `secure file priv` na prázdnou hodnotu mohou uživatelé s oprávněním `file_priv` načítat soubory odkudkoli ze systému, viz 3.11. To může představovat riziko úniku dat ze systému hostující databázový server.

```
MariaDB [(none)]> LOAD DATA LOCAL INFILE 'C:/Windows/System32/SuperSecrets.txt'
    -> INTO TABLE my_schema.file_read LINES TERMINATED BY '\r\n';
MariaDB [(none)]> SELECT * FROM my_schema.file_read;
+-----------------------------------+
| line                              |
+-----------------------------------+
| First secret.                     |
| Second secret.                    |
| Third secret.                     |
| This is last ultimate secret (42). |
+-----------------------------------+
```

Obr. 3.11: Načtení hodnot ze systému hostující databázový server

3.2.8 Log configuration

K tomuto testu bylo potřeba zapnout logování `general_log` s ukládáním do souboru, viz 3.12.

```
MariaDB [(none)]> SET GLOBAL general_log = 'ON';
MariaDB [(none)]> SET GLOBAL log_output = 'FILE';
MariaDB [(none)]> SHOW VARIABLES LIKE 'general_log_file';
+------------------+----------------------+
| Variable_name    | Value                |
+------------------+----------------------+
| general_log_file | DESKTOP-JVIIPK1.log |
+------------------+----------------------+
MariaDB [(none)]> CREATE USER 'super_secret_user'@'localhost'
    -> IDENTIFIED BY 'SuperMegaSecretP4ssw0rd!666';
```

Obr. 3.12: Nastavení logování

V MariaDB není nastavení, které by zamezovalo ukládání citlivých informací do logů. Kdokoli, kdo k nim má přístup, může zjistit například hesla vytvořených uživatelů, viz 3.13. To znamená, že ani sebesilnější heslo není zárukou bezpečnosti, když je databázový server chybně nakonfigurován.

```
DESKTOP-JVIIPK1.log – Poznámkový blok
Soubor  Úpravy  Formát  Zobrazení  Nápověda
C:\Program Files\MariaDB 11.7\bin\mysqld.exe, Version: 11.7.2-MariaDB (mari
TCP Port: 3306, Named Pipe: MySQL
Time                 Id Command  Argument
250511 20:41:42     28 Query    SET GLOBAL log_output = 'FILE'
250511 20:41:57     28 Query    SHOW VARIABLES LIKE 'general_log_file'
250511 20:43:43     28 Query    CREATE USER 'super_secret_user'@'localhost'
IDENTIFIED BY 'SuperMegaSecretP4ssw0rd!666'
```

Obr. 3.13: Citlivé informace v `general_logu`

3.3 Limitace

Vytvořený nástroj vyžaduje přístup k databázovému serveru s oprávněními potřebnými pro čtení systémových proměnných, obsahů tabulek a při použití argumentu `--setup-db` práva k vytvoření nové databáze a uživatelů. Pokud se při spuštění programů zadá uživatel s nedostatečnými oprávněními, provede se omezený počet testů nebo se program rovnou ukončí.

Mezi různými verzemi databázových systémů mohou existovat rozdíly v dostupnosti nebo názvech proměnných. To může vést k částečnému nebo nepřesnému vyhodnocení testovacího nástroje. Funkčnost byla testována na aktuálních verzích MariaDB (11.7.2) a MySQL (9.2.0) serverech.

Nástroj pokrývá pouze běžně používané konfigurační aspekty. Nestandardní nebo proprietární úpravy konfigurace nemusejí být detekovány nebo správně vyhodnoceny. Když program odhalí potenciální problém, nemusí to vždy znamenat nesprávnou konfiguraci. Vždy záleží na konkrétním kontextu a je potřeba výsledky správně interpretovat (např. vypnuté šifrování nemusí být problém na testovací databázi s falešnými daty).

Limitace u testu autentizačních metod je způsob kontroly, kdy se porovnávají použité pluginy proti seznamům bezpečných a zastaralých metod. Pokud se tedy objeví nové bezpečné pluginy, nebo bezpečnostní chyby ve stávajících, bude potřeba upravit tyto seznamy, aby byla nastavení správně vyhodnocena.

Další omezení je u testu nastavení konfigurace SSL/TLS. Zde se kontroluje pouze, jestli jsou proměnné `ssl_ca`, `ssl_cert` a `ssl_key` ve formátu cesty k souboru. Netestuje se, jestli jsou to cesty k funkčním souborům.

3.4 Budoucí rozvoj

Navazující práce by se mohly zaměřit na zvýšení počtu stávajících testů, další typy databázových systémů (MongoDB, SQLite, Snowflake a další) a vylepšení již vytvořených testů.

Další funkcionalitou by mohla být možnost automatické nápravy nalezených chyb v konfiguraci. To by mohlo být realizováno formou vygenerování SQL příkazů nebo úprav konfiguračních souborů s popisem, proč a jak změny provést.

V současné době jsou všechny texty pouze v angličtině. Pro využití ve více zemích by bylo vhodné přidat podporu vícejazyčného výstupu.

3.5 Případy využití

Jedním z možných využití by mohl být pravidelný audit konfigurace databázových serverů. Administrátor databáze pravidelně spouští nástroj pro kontrolu, zda konfigurace databázového serveru odpovídá firemním bezpečnostním politikám. Výhodou by byla rychlá detekce neautorizovaných změn v konfiguraci, které by mohly ohrozit bezpečnost systému.

Dalším případem by mohla být kontrola nového prostředí před nasazením do produkce nebo při převzetí prostředí nového klienta. Přínosem by bylo snížení rizika

chyb a nedostatků konfigurace spolu s úsporou času při orientaci v neznámém prostředí.

Nástroje by se mohly využít i při diagnostice incidentů pro ověření aktuální konfigurace a identifikaci proměnných, které by mohly způsobovat problémy s výkonem nebo narušení bezpečnosti. Výhodou by bylo zrychlení řešení problémů a snížení případné doby výpadku služeb.

Závěr

Tato diplomová práce se zabývala návrhem a implementací softwarových nástrojů pro kontrolu bezpečného nastavení databázových systémů MariaDB a MySQL. Hlavním cílem bylo vytvořit nástroje, které umožní analyzovat konfigurace databázových systémů, identifikovat potenciální bezpečnostní rizika a generovat přehledný report s doporučeními pro jejich odstranění.

V první kapitole byly popsány základní charakteristiky databázových systémů MySQL a MariaDB a analyzovány klíčové aspekty bezpečnosti databází. Součástí teoretické rešerše byla také analýza nejčastějších zranitelností a útoků, které na databáze cílí. Tyto poznatky poskytly důležité východisko pro návrh vlastního řešení.

Druhá kapitola práce se zaměřila na implementaci testovacích nástrojů v jazyce Python. Byly navrženy testy pro kontrolu vybraných bezpečnostních parametrů MariaDB a MySQL databází. Nástroje zároveň umožňují generování výstupního reportu ve formátu PDF, což usnadňuje interpretaci výsledných kontrol.

Třetí kapitola popsala přípravu testovacího prostředí a demonstrovala útoky, které mohou při nevhodné konfiguraci databázových systémů způsobit únik dat, narušení důvěrnosti či integrity dat. Jsou v ní popsány také limitace a možnosti budoucího rozvoje vytvořených nástrojů.

Celkově lze práci hodnotit jako úspěšnou. Byly vytvořeny nástroje, které přispějí ke zvýšení bezpečnosti konfigurací databázových systémů MySQL a MariaDB. Uživatelům také poskytnou efektivní způsob kontroly a optimalizace jejich nastavení.

Literatura

[1] DB-Engines Ranking. *db-engines* [online]. [cit. 13. 10. 2024]. Dostupné z URL: <https://db-engines.com/en/ranking>.

[2] What Is a Database?. *Oracle* [online]. [cit. 3. 12. 2024]. Dostupné z URL: <https://www.oracle.com/database/what-is-database/>.

[3] What is a relational database?. *IBM Topics* [online]. [cit. 3. 12. 2024]. Dostupné z URL: <https://www.ibm.com/topics/relational-databases>.

[4] object-oriented database management system (OODBMS). *techtarget* [online]. [cit. 4. 12. 2024]. Dostupné z URL: <https://www.techtarget.com/searchoracle/definition/object-oriented-database-management-system>.

[5] What are Object Databases?. *predictiveanalyticstoday* [online]. [cit. 4. 12. 2024]. Dostupné z URL: <https://www.predictiveanalyticstoday.com/top-object-databases/>.

[6] What is a Distributed Database?. *MongoDB* [online]. [cit. 4. 12. 2024]. Dostupné z URL: <https://www.mongodb.com/resources/basics/databases/distributed-database#:~:text=In%20the%20most%20basic%20terms,computers%20consisting%20of%20individual%20nodes.>.

[7] Distributed Database System. *geeksforgeeks* [online]. [cit. 4. 12. 2024]. Dostupné z URL: <https://www.geeksforgeeks.org/distributed-database-system/>.

[8] What is a distributed database and how do they work?. *cockroachlabs* [online]. [cit. 4. 12. 2024]. Dostupné z URL: <https://www.cockroachlabs.com/blog/what-is-a-distributed-database/>.

[9] What Is a Data Warehouse?. *Oracle* [online]. [cit. 4. 12. 2024]. Dostupné z URL: <https://www.oracle.com/cz/database/what-is-a-data-warehouse/>.

[10] What is a data warehouse?. *IBM Topics* [online]. [cit. 4. 12. 2024]. Dostupné z URL: <https://www.ibm.com/topics/data-warehouse>.

[11] What is NoSQL?. *AWS Amazon* [online]. [cit. 4. 12. 2024]. Dostupné z URL: <https://aws.amazon.com/nosql/>.

[12] Databáze NoSQL – co je NoSQL?. *Azure Microsoft* [online]. [cit. 4. 12. 2024]. Dostupné z URL: <https://azure.microsoft.com/cs-cz/resources/cloud-computing-dictionary/what-is-nosql-database>.

[13] NoSQL Database Types. *phoenix NAP* [online]. [cit. 11. 12. 2024]. Dostupné z URL: <https://phoenixnap.com/kb/nosql-database-types>.

[14] What is a NoSQL database?. *IBM Topics* [online]. [cit. 4. 12. 2024]. Dostupné z URL: <https://www.ibm.com/topics/nosql-databases>.

[15] Comparing database types: how database types evolved to meet different needs. *Prisma's Data Guide* [online]. [cit. 5. 12. 2024]. Dostupné z URL: <https://www.prisma.io/dataguide/intro/comparing-database-types>.

[16] What is MySQL?. *dev.mysql* [online]. [cit. 13. 10. 2024]. Dostupné z URL: <https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html>.

[17] History of MySQL. *dev.mysql* [online]. [cit. 11. 12. 2024]. Dostupné z URL: <https://dev.mysql.com/doc/refman/8.0/en/history.html>.

[18] Powered by MySQL. *mysql* [online]. [cit. 11. 12. 2024]. Dostupné z URL: <https://www.mysql.com/>.

[19] Using Option Files. *mysql* [online]. [cit. 11. 12. 2024]. Dostupné z URL: <https://dev.mysql.com/doc/refman/8.4/en/option-files.html>.

[20] MySQL-Configuration-File. *getmoven* [online]. [cit. 27. 10. 2024]. Dostupné z URL: <https://getmoven.com/index.php/knowledgebase/18/Sample-my.cnf---MySQL-Configuration-File.html>.

[21] mysql Client Options. *mysql* [online]. [cit. 27. 10. 2024]. Dostupné z URL: <https://dev.mysql.com/doc/refman/8.4/en/mysql-command-options.html>.

[22] mysqld_safe — MySQL Server Startup Script. *mysql* [online]. [cit. 27. 10. 2024]. Dostupné z URL: <https://dev.mysql.com/doc/refman/8.4/en/mysqld-safe.html>.

[23] Server Command Options. *mysql* [online]. [cit. 27. 10. 2024]. Dostupné z URL: <https://dev.mysql.com/doc/refman/8.4/en/server-options.html>.

[24] mysqldump — A Database Backup Program. *mysql* [online]. [cit. 27. 10. 2024]. Dostupné z URL: <https://dev.mysql.com/doc/refman/8.4/en/mysqldump.html#mysqldump-option-file-options>.

[25] MariaDB. *MariaDB.org* [online]. [cit. 1. 12. 2024]. Dostupné z URL: <https://mariadb.org/en/>.

[26] K1 Acquires MariaDB, a Leading Database Software Company, and Appoints New CEO. *PR Newswire* [online]. [cit. 1. 12. 2024]. Dostupné z URL: <https://www.prnewswire.com/news-releases/k1-acquires-mariadb-a-leading-database-software-company-and-appoints-new-ceo-302243508.html>.

[27] Red Hat ditches MySQL, switches to MariaDB. *iTWire* [online]. [cit. 11. 12. 2024]. Dostupné z URL: <https://itwire.com/business-it-news/open-source/60292-red-hat-ditches-mysql-switches-to-mariadb>.

[28] Wikipedia Adopts MariaDB. *Wikimedia* [online]. [cit. 11. 12. 2024]. Dostupné z URL: <https://diff.wikimedia.org/2013/04/22/wikipedia-adopts-mariadb/>.

[29] Google swaps out MySQL, moves to MariaDB. *The Register* [online]. [cit. 11. 12. 2024]. Dostupné z URL: <https://www.theregister.com/2013/09/12/google_mariadb_mysql_migration/>.

[30] MariaDB database software is the heartbeat of critical services people rely on every day. *MariaDB* [online]. [cit. 11. 12. 2024]. Dostupné z URL: <https://mariadb.com/>.

[31] Configuring MariaDB with Option Files. *MariaDB* [online]. [cit. 3. 5. 2025]. Dostupné z URL: <https://mariadb.com/kb/en/configuring-mariadb-with-option-files/>.

[32] Database Security: Common threats and challenges, Why is it important?. *IBM Topics* [online]. [cit. 1. 12. 2024]. Dostupné z URL: <https://www.ibm.com/topics/database-security>.

[33] Former employees behind Tesla data breach. *CYFOR FORENSICS* [online]. [cit. 3. 5. 2025]. Dostupné z URL: <https://cyfor.co.uk/former-employees-behind-tesla-data-breach/#:~:text=Tesla%2C%20the%20electric%20car%20maker,the%20leak%20on%20ex%2Demployees.>.

[34] The Target Breach: A Historic Cyberattack with Lasting Consequences. *FRAMEWORK SECURITY* [online]. [cit. 3. 5. 2025]. Dostupné z URL: <https://www.frameworksec.com/post/the-target-breach-a-historic-cyberattack-with-lasting-consequences>.

[35] 95% of Data Breaches Tied to Human Error in 2024. *Infosecurity Magazine* [online]. [cit. 3. 5. 2025]. Dostupné z URL: <https://www.infosecurity-magazine.com/news/data-breaches-human-error/#:~:text=Human%20error%20contributed%20to%2095,accounting%20for%2080%25%20of%20incidents.>.

[36] Microsoft Employees Exposed Own Company's Internal Logins. *VICE* [online]. [cit. 3. 5. 2025]. Dostupné z URL: <https://www.vice.com/en/article/microsoft-employees-exposed-login-credentials-azure-github/>.

[37] What is a zero-day exploit?. *IBM* [online]. [cit. 4. 5. 2025]. Dostupné z URL: <https://www.ibm.com/think/topics/zero-day>.

[38] U.S. Says Ring Stole 160 Million Credit Card Numbers. *TheNewYorkTimes* [online]. [cit. 4. 5. 2025]. Dostupné z URL: <https://archive.nytimes.com/dealbook.nytimes.com/2013/07/25/arrests-planned-in-hacking-of-financial-companies/>.

[39] Remembering SQL Slammer. *NETSCOUT* [online]. [cit. 4. 5. 2025]. Dostupné z URL: <https://www.netscout.com/blog/asert/remembering-sql-slammer>.

[40] Pegasus (spyware). *Britannica* [online]. [cit. 4. 5. 2025]. Dostupné z URL: <https://www.britannica.com/topic/Pegasus-spyware>.

[41] What was the WannaCry ransomware attack?. *CLOUDFLARE* [online]. [cit. 4. 5. 2025]. Dostupné z URL: <https://www.cloudflare.com/learning/security/ransomware/wannacry-ransomware/>.

[42] Types of cyberthreats – Malware. *IBM* [online]. [cit. 4. 5. 2025]. Dostupné z URL: <https://www.ibm.com/think/topics/cyberthreats-types#Malware>.

[43] How the Economy, Skills Gap and Artificial Intelligence are Challenging the Global Cybersecurity Workforce 2023. *ISC2 CYBERSECURITY WORKFORCE STUDY* [online]. [cit. 7. 5. 2025]. Dostupné z URL: <https://media.isc2.org/-/media/Project/ISC2/Main/Media/documents/research/ISC2_Cybersecurity_Workforce_Study_2023.pdf?rev=28b46de71ce24e6ab7705f6e3da8637e>.

[44] Attack on forum of Hong Kong protesters. *COUNCIL on FOREIGN RELATIONS* [online]. [cit. 4. 5. 2025]. Dostupné z URL: <https://www.cfr.org/cyber-operations/attack-forum-hong-kong-protesters>.

[45] China's Great Cannon. *THECITIZENLAB* [online]. [cit. 4. 5. 2025]. Dostupné z URL: <https://citizenlab.ca/2015/04/chinas-great-cannon/>.

[46] Dodging 5 Dangerous Database Default Settings. *Dark Reading* [online]. [cit. 6. 12. 2024]. Dostupné z URL: <https://www.darkreading.com/application-security/dodging-5-dangerous-database-default-settings>.

[47] MySQLTuner-perl. *Github* [online]. [cit. 1. 12. 2024]. Dostupné z URL: <https://github.com/major/MySQLTuner-perl>.

[48] Keep MySQL 5.7 Secure and Performant. *Percona* [online]. [cit. 1. 12. 2024]. Dostupné z URL: <https://www.percona.com/post-mysql-5-7-eol-support>.

[49] Plot types. *matplotlib* [online]. [cit. 8. 12. 2024]. Dostupné z URL: <https://matplotlib.org/stable/plot_types/index>.

[50] PyLaTeX. *Github PyLaTeX* [online]. [cit. 8. 12. 2024]. Dostupné z URL: <https://jeltef.github.io/PyLaTeX/current/>.

[51] Authentication Plugin – mysql_native_password. *MariaDB* [online]. [cit. 20. 3. 2025]. Dostupné z URL: <https://mariadb.com/kb/en/authentication-plugin-mysql_native_password/>.

[52] Authentication Plugin – mysql_old_password. *MariaDB* [online]. [cit. 20. 3. 2025]. Dostupné z URL: <https://mariadb.com/kb/en/authentication-plugin-mysql_old_password/>.

[53] SHA-256 Pluggable Authentication. *MySQL* [online]. [cit. 21. 3. 2025]. Dostupné z URL: <https://dev.mysql.com/doc/refman/8.4/en/sha256-pluggable-authentication.html>.

[54] Caching SHA-2 Pluggable Authentication. *MySQL* [online]. [cit. 21. 3. 2025]. Dostupné z URL: <https://dev.mysql.com/doc/refman/8.4/en/caching-sha2-pluggable-authentication.html>.

[55] Socket Peer-Credential Pluggable Authentication. *MySQL* [online]. [cit. 21. 3. 2025]. Dostupné z URL: <https://dev.mysql.com/doc/refman/8.4/en/socket-pluggable-authentication.html>.

[56] MySQL Community Downloads – MySQL Community Server. *MySQL* [online]. [cit. 21. 3. 2025]. Dostupné z URL: <https://dev.mysql.com/downloads/mysql/>.

[57] Server System Variables. *MySQL* [online]. [cit. 23. 3. 2025]. Dostupné z URL: <https://dev.mysql.com/doc/refman/8.4/en/server-system-variables.html>.

[58] The General Query Log. *MySQL* [online]. [cit. 23. 3. 2025]. Dostupné z URL: <https://dev.mysql.com/doc/refman/8.4/en/query-log.html>.

[59] The Slow Query Log. *MySQL* [online]. [cit. 23. 3. 2025]. Dostupné z URL: <https://dev.mysql.com/doc/refman/8.4/en/slow-query-log.html>.

[60] InnoDB Startup Options and System Variables. *MySQL* [online]. [cit. 23. 3. 2025]. Dostupné z URL: <https://dev.mysql.com/doc/refman/8.4/en/innodb-parameters.html>.

[61] Privileges Provided by MySQL. *MySQL* [online]. [cit. 25. 3. 2025]. Dostupné z URL: <https://dev.mysql.com/doc/refman/8.4/en/privileges-provided.html>.

[62] GRANT. *MariaDB* [online]. [cit. 25. 4. 2025]. Dostupné z URL: <https://mariadb.com/kb/en/grant/#tls-options>.

[63] Server System Variables. *MariaDB* [online]. [cit. 25. 4. 2025]. Dostupné z URL: <https://mariadb.com/kb/en/server-system-variables/>.

[64] SSL/TLS System Variables. *MariaDB* [online]. [cit. 25. 4. 2025]. Dostupné z URL: <https://mariadb.com/kb/en/ssltls-system-variables/>.

[65] InnoDB Encryption Overview. *MariaDB* [online]. [cit. 25. 4. 2025]. Dostupné z URL: <https://mariadb.com/kb/en/innodb-encryption-overview/>.

[66] Information Schema INNODB_TABLESPACES_ENCRYPTION Table. *MariaDB* [online]. [cit. 25. 4. 2025]. Dostupné z URL: <https://mariadb.com/kb/en/information-schema-innodb_tablespaces_encryption-table/>.

[67] InnoDB System Variables. *MariaDB* [online]. [cit. 25. 4. 2025]. Dostupné z URL: <https://mariadb.com/kb/en/innodb-system-variables/>.

[68] Authentication Plugin – ed25519. *MariaDB* [online]. [cit. 25. 4. 2025]. Dostupné z URL: <https://mariadb.com/kb/en/authentication-plugin-ed25519/>.

[69] Authentication Plugin – GSSAPI. *MariaDB* [online]. [cit. 25. 4. 2025]. Dostupné z URL: <https://mariadb.com/kb/en/authentication-plugin-gssapi/>.

[70] Authentication Plugin – Unix Socket. *MariaDB* [online]. [cit. 25. 4. 2025]. Dostupné z URL: <https://mariadb.com/kb/en/authentication-plugin-unix-socket/>.

[71] Authentication Plugin – Named Pipe. *MariaDB* [online]. [cit. 25. 4. 2025]. Dostupné z URL: <https://mariadb.com/kb/en/authentication-plugin-named-pipe/>.

[72] Download MariaDB Community Server. *MariaDB* [online]. [cit. 26. 4. 2025]. Dostupné z URL: <https://mariadb.com/downloads/community/community-server/>.

[73] mysql.func Table. *MariaDB* [online]. [cit. 26. 4. 2025]. Dostupné z URL: <https://mariadb.com/kb/en/mysql-func-table/>.

[74] User-Defined Functions Security. *MariaDB* [online]. [cit. 26. 4. 2025]. Dostupné z URL: <https://mariadb.com/kb/en/user-defined-functions-security/>.

[75] General Query Log. *MariaDB* [online]. [cit. 27. 4. 2025]. Dostupné z URL: <https://mariadb.com/kb/en/general-query-log/>.

[76] Slow Query Log Overview. *MariaDB* [online]. [cit. 27. 4. 2025]. Dostupné z URL: <https://mariadb.com/kb/en/slow-query-log-overview/>.

[77] Overview of the Binary Log. *MariaDB* [online]. [cit. 27. 4. 2025]. Dostupné z URL: <https://mariadb.com/kb/en/overview-of-the-binary-log/>.

[78] Encrypting Binary Logs. *MariaDB* [online]. [cit. 27. 4. 2025]. Dostupné z URL: <https://mariadb.com/kb/en/encrypting-binary-logs/>.

[79] Super. *MariaDB* [online]. [cit. 27. 4. 2025]. Dostupné z URL: <https://mariadb.com/kb/en/grant/#super>.

Seznam symbolů a zkratek

ACID	atomicita konzistence izolovanost trvalost – Atomicity Consistency Isolation Durability
BSON	binární JSON – Binary JSON
DB	báze dat/databáze – Database
DBMS	systému řízení báze dat/databázový systém – Database Management System
DoS	odepření služby – Denial of service
DDoS	distribuovaný DoS – Distributed DoS
DSA	algoritmus digitálního podpisu – Digital Signature Algorithm
GDPR	obecné nařízení o ochraně osobních údajů – General Data Protection Regulation
GNU GPL	obecná veřejná licence GNU – GNU General Public License
GSSAPI	rozhraní programů pro přístup k bezpečnostním službám – Generic Security Service Application Programming Interface
IoT	internet věcí – Internet of Things
IP	internetový protokol – Internet Protocol
IT	informační technologie – Information Technology
JSON	JavaScriptový objektový zápis – JavaScript Object Notation
MITM	útok typu prostředník/člověk uprostřed – Man In The Middle attack
NIST	národní institut standardů a technologie – National Institute of Standards and Technology
NSA	národní bezpečnostní agentura – The National Security Agency
NTLM	autentizační protokol v Microsoft Windows – New Technology LAN Manager
PCI DSS	mezinárodní bezpečnostní standard nakládání s daty o držitelích platebních karet – Payment Card Industry Data Security Standard
PDF	přenosný formát dokumentů – Portable Document Format

PEM	souborový formát pro ukládání kryptografických dat – Privacy-Enhanced Mail
PHP	PHP: hypertextový preprocesor – PHP: Hypertext Preprocessor
SMS	služba krátkých textových zpráv – Short Message Service
SOX	zákon Sarbanes-Oxley – Sarbanes-Oxley Act
SQL	standardizovaný strukturovaný dotazovací jazyk – Structured Query Language
SSL	protokol poskytující zabezpečenou komunikaci – Secure Sockets Layer
SW	software/programové vybavení – Software
TDE	transparentní šifrování dat – Transparent Data Encryption
TLS	protokol poskytující zabezpečenou komunikaci – Transport Layer Security
UDF	uživatelem definované funkce – User-Defined Functions
UDP	protokol internetu, který neručí za spolehlivý přenos dat – User Datagram Protocol
XML	rozšiřitelný značkovací jazyk – Extensible Markup Language
YAML	YAML není značkovací jazyk – YAML Ain't Markup Language

A Obsah elektronické přílohy

Součástí práce je archiv formátu `.zip` obsahující soubory vytvořených programů. Vytvořené programy lze nalézt i v online repozitářích `https://gitlab.com/hc-tools1/hc-mariadb` a `https://gitlab.com/hc-tools1/hc-mysql`. Součástí elektronické přílohy jsou i příklady výstupních reportů.

- `/` ..... kořenový adresář přiložených archivů
  - `hc-mariadb/hc-mysql` ..... Soubory pro program testující databázový server
    - `example_db` ..... adresář s SQL skriptem
      - `create_db.sql` ..... SQL příkaz pro vytvoření testovací databáze
    - `examples` ..... adresář se skripty na testování programu
      - `library_usage.py` ..... skript testující rychlost programu
      - `paralel_testing.py` ..... skript testující výkon při paralelním spuštění
    - `latex_template` ..... adresář pro soubory výsledného reportu
      - `imgs` ..... adresář pro obrázky reportu
        - `fekt.png` ..... logo FEKTu použito v reportu
      - `main.tex` ..... hlavní TeX soubor se strukturou a použitými balíčky
    - `utils` ..... adresář pro pomocné nástroje
      - `errors.py` ..... definice chybových hlášek
      - `global_logger.py` ..... nastavení logování programu
      - `parsers.py` ..... funkce pro převádění mezi různými formáty
      - `utils.py` ..... další pomocné funkce
    - `LICENCE` ..... text MIT licence
    - `README.md` ..... soubor pro popis repozitáře na GitLabu
    - `dbaudit.py` ..... skript spouštějící generování dokumentace
    - `dependencies` ..... soubor s použitými moduly a knihovnami
    - `document_builder.py` ..... soubor s funkcemi pro generování reportu
    - `experimental_db_setup.py` ..... skript na vytvoření testovací databáze
    - `latex_generator.py` ..... soubor s funkcemi na převedení textu do LaTeXu
    - `main.py` ..... hlavní skript který se spouští
    - `session.py` ..... definice třídy session, pracuje s zadanými argumenty
    - `tests.py` ..... funkce pro vyhodnocení testů databáze
  - `priklad_reportu_MariaDB.pdf` ..... ukázka možného výstupu pro hc-mariadb
  - `priklad_reportu_MySQL.pdf` ..... ukázka možného výstupu pro hc-mysql

B Příklad výsledného reportu pro MariaDB

Database Configuration Security Compliance

2025-05-23 13:09

Tested System Information

Path to MariaDB Configuration: C:\Program Files\MariaDB 11.7\data
Database Name: experimental_db
User: root
Host: localhost
Port: 3306
CPU: AMD64 Family 25 Model 80 Stepping 0, AuthenticAMD (4 cores)
RAM: 3 GB
Storage Size: 49 GB
Operating System: Windows 10
MariaDB Version: 11.7.2

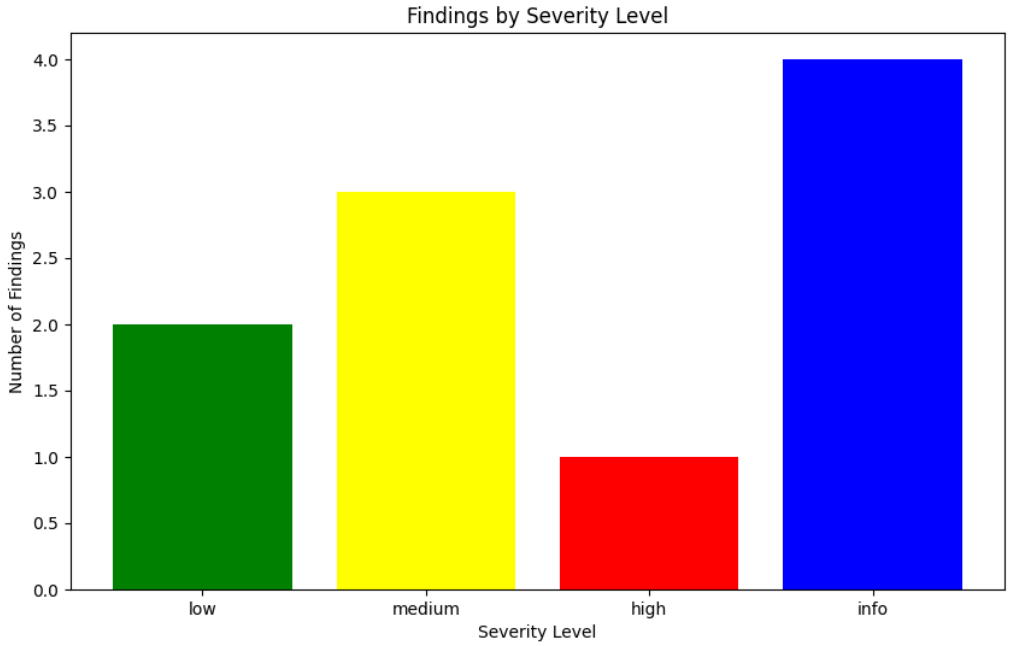
1 Tested areas

No.	area of interest	tested	compliant	severity
1	Encryption at transit	✓	×	low
2	Encryption at rest	✓	×	low
3	Insecure authentication plugins	✓	×	medium
4	Trust authentication	✓	×	high
5	Latest version of MariaDB	✓	✓	low
6	Permissions test	✓	×	info
7	User defined functions	✓	×	info
8	File system access	✓	×	medium
9	Log configuration	✓	×	info
10	Configuration of SSL	✓	×	medium
11	SUPER privileges	✓	×	info

2 Summary

During the test total of 10 misconfigurations were found.

Following figure shows graphically number of misconfigurations and their impact:



3 Technical details

3.1 Encryption at transit

Description This test verifies that database enforces encryption to ensure safe communication that cannot be eavesdropped. Improper configuration of encryption could lead to violation of CIA triade.

This test found that following users are not configured to ensure encrypted communication:

User	Host	SSL Type
root	::1	×
public_user	%	×
test_user	%	×
tester	%	×
super_secret_user	localhost	×

Clients aren't required to use form of secure transport. No SSL encryption cipher specified.

Variable	Value
require_secure_transport	OFF
ssl_cipher	×

3.2 Encryption at rest

Description This test verifies if database tablespaces are encrypted. If the value of encryption scheme is 1, it indicates that the tablespace is encrypted, while 0 indicates that the tablespace is not encrypted. The test will list all tablespaces along with their encryption status and ID of the key used for encryption.

Not all tablespaces are encrypted. **No record in information_schema.innodb_tablespaces_encryption table. Table encryption is disabled for all new and existing tables that have the ENCRYPTED table option set to DEFAULT. Encryption of the InnoDB redo log is disabled. Automatic encryption of the InnoDB temporary tablespace is disabled.**

Variable	Value
innodb_encrypt_tables	OFF
innodb_encrypt_log	OFF
innodb_encrypt_temporary_tables	OFF

3.3 Insecure authentication plugins

Description This test examines the 'mysql.user' table for outdated authentication plugins. Specifically, it identifies the use of 'mysql_old_password', which is highly insecure and has been removed in modern MariaDB versions, and 'mysql_native_password', which relies on the SHA1 hashing algorithm and is considered less secure than newer authentication plugins such as 'ed25519'. While 'mysql_native_password' is still widely used, its reliance on SHA1 makes it vulnerable to cryptographic weaknesses, and its use is discouraged in favor of stronger authentication plugins.

Database doesn't enforce secure authentication methods

User	Plugin	Security
root	mysql_native_password	insecure
public_user	ed25519	secure
test_user	mysql_native_password	insecure
private_user	mysql_old_password	insecure
admin_user	mysql_native_password	insecure
tester	mysql_native_password	insecure
super_secret_user	mysql_native_password	insecure

3.4 Trust authentication

Description Trust authentication permits unrestricted access to the database for any user without requiring a password. This configuration poses a significant security risk, as it allows potentially unauthorized individuals to gain access to sensitive data and perform unauthorized actions within the

database. Utilizing trust authentication undermines the fundamental principle of access control and compromises the confidentiality, integrity and availability of the database.

Database allows some users to connect without password.

User	Plugin	Password
test_user	mysql_native_password	No password or NULL

3.5 Latest version of MariaDB

Description This test verifies whether the database uses the latest software version. Outdated versions could contain security vulnerabilities that could be used by an attacker to compromise the database.

Database server uses latest version of MariaDB (11.7.2).

3.6 Permissions test

Description The following table provides a comprehensive overview of all privileges assigned within the specified database. This information is crucial for evaluating the access control mechanisms in place and identifying potential security vulnerabilities. A thorough permissions audit ensures that only authorized users have appropriate access rights, minimizing the risk of unauthorized data access or modification. Following table contains users and their permission on database tables:

User Type	Table Schema	Table Name	Privilege Types
'private_user'@'%'	my_schema	private_info	SELECT
'private_user'@'%'	my_schema	public_info	SELECT
'admin_user'@'localhost'	my_schema	private_info	SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, REFERENCES, INDEX, ALTER, CREATE VIEW, SHOW VIEW, TRIGGER, DELETE HISTORY
'admin_user'@'localhost'	my_schema	public_info	SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, REFERENCES, INDEX, ALTER, CREATE VIEW, SHOW VIEW, TRIGGER, DELETE HISTORY
'admin_user'@'localhost'	my_schema	secret_info	SELECT, INSERT, UPDATE, DELETE, CREATE, DROP, REFERENCES, INDEX, ALTER, CREATE VIEW, SHOW VIEW, TRIGGER, DELETE HISTORY
'test_user'@'%'	my_schema	public_info	SELECT
'public_user'@'%'	my_schema	public_info	SELECT

3.7 User defined functions

Description The purpose of this test is to verify that the MariaDB server is properly secured against potential abuse of user defined functions. The test will inspect the contents of the mysql.func table, which stores information about any custom functions that have been loaded into the server. Additionally, the test will check schema privileges table for users with privileges over mysql database.

No functions in mysql.func table. **Users with direct change privileges over `mysql` database:**

Grantee	Table schema	Privileges
'admin_user'@'localhost'	mysql	INSERT, DELETE

3.8 File system access

Description Tests that the MariaDB server is properly configured to restrict file system access and that only authorized users have the FILE privilege.

MariaDB server has unrestricted write/read access to files. Local is supported for `LOAD DATA INFILE` statements.

Variable	Value
secure_file_priv	
local_infile	ON

Users in following table have privilege to read/write to files.

User	Host	File_priv
root	localhost	Y
admin_user	localhost	Y
tester	%	Y

3.9 Log configuration

Description Verifies that the the logging configuration of a MariaDB server prevents sensitive data exposure and ensure compliance with security best practices.

General logging is turned off. **Slow query logging is off.** Long query time is set reasonably. **Binary logging is turned off. Encryption of binary logs is turned off.**

Variable	Value
general_log	OFF
slow_query_log	OFF
long_query_time	10.0
log_bin	OFF
encrypt_binlog	OFF

3.10 Configuration of SSL

Description This test verifies whether MariaDB has SSL enabled. Additionally, it ensures that the required SSL variables are correctly configured.

MariaDB server supports TLS and TLS is enabled. **No path in ssl_ca to file that contains trusted Certificate Authorities for TLS. No path in ssl_cert to the certificate for TLS. No path in ssl_key to the private key for TLS.**

Variable	Value
have_ssl	YES
ssl_ca	×
ssl_cert	×
ssl_key	×

3.11 SUPER privileges

Description This test checks which users have the SUPER privilege in the MariaDB database. The test queries the mysql.user table.

The following users have **SUPER** privileges, which grant them extensive control over the MariaDB server. This privilege allows modifying global settings, managing replication, and terminating processes. It should be restricted to administrative users only. Consider reviewing and revoking **SUPER** where it is not necessary.

User	Host	SUPER
root	localhost	Y
admin_user	localhost	Y