FAKULTA INFORMAČNÍCH TECHNOLOGIÍ ČVUT V PRAZE

# Zadání bakalářské práce

| | |
|---|---|
| **Název:** | Bezpečnostní analýza zálohovacího nástroje Duplicati |
| **Student:** | Radek Večerník |
| **Vedoucí:** | Ing. Jiří Dostál, Ph.D. |
| **Studijní program:** | Informatika |
| **Obor / specializace:** | Bezpečnost a informační technologie |
| **Katedra:** | Katedra počítačových systémů |
| **Platnost zadání:** | do konce letního semestru 2023/2024 |

## Pokyny pro vypracování

1) Seznamte se s problematikou zálohování souborových systémů.
2) Seznamte se softwarem pro tvorbu záloh Duplicati (https://www.duplicati.com/).
3) Analyzujte použití kryptografie při vytváření záloh a další funkcionality zálohovacího nástroje, jako je kontrola integrity, inkrementální zálohy a uživatelské rozhraní.
4) Proveďte bezpečnostní analýzu nástroje na základě poznatků z předchozího bodu. Zaměřte se také na souborový formát záloh a metadata, lokální databázi, kontrolu vstupů od uživatele a využití hesel a šifrovacích klíčů.
5) Naleznete-li nějaké zranitelnosti, demonstrujte je napsáním nástroje, který na ně zaútočí, nebo popište podmínky, za kterých by takový útok uspěl. Navrhněte opravná opatření.

---

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

Bakalářská práce

# BEZPEČNOSTNÍ ANALÝZA ZÁLOHOVACÍHO NÁSTROJE DUPLICATI

**Radek Večerník**

Fakulta informačních technologií
Katedra informační bezpečnosti
Vedoucí: Ing. Jiří Dostál, Ph.D.
11. května 2023

# Obsah

**Poděkování** vi

**Prohlášení** vii

**Abstrakt** viii

**Seznam zkratek** ix

**Úvod** 1

**1 Problematika zálohování počítačových dat** 3
- 1.1 Základní pojmy . . . 3
- 1.2 Metody zálohování . . . 4
  - 1.2.1 Úplná záloha . . . 4
  - 1.2.2 Rozdílová záloha . . . 4
  - 1.2.3 Inkrementální záloha . . . 4
- 1.3 Plánování a verzování záloh . . . 4
- 1.4 Typy úložišť a paměťových médií . . . 5
- 1.5 Existující současné řešení . . . 5

**2 Zálohovací nástroj Duplicati** 7
- 2.1 Funkcionality . . . 7
  - 2.1.1 Inkrementální zálohy, deduplikace a komprese . . . 7
  - 2.1.2 Šifrování záloh a zabezpečení . . . 8
  - 2.1.3 Rozhraní . . . 8
  - 2.1.4 Podpora úložišť . . . 9
  - 2.1.5 Další vlastnosti . . . 9
- 2.2 Proces tvorby zálohy . . . 9
- 2.3 Zkoumání klíčových částí . . . 11

**3 Analýza a rozbor nástroje** 13
- 3.1 Využívané moduly a knihovny . . . 13
- 3.2 Formát souborů se zálohou . . . 14
  - 3.2.1 Soubor typu AES . . . 14
  - 3.2.2 Soubor typu ZIP . . . 16
  - 3.2.3 Soubor zálohy dblock . . . 16
  - 3.2.4 Soubor zálohy dlist . . . 16
  - 3.2.5 Soubor zálohy dindex . . . 17
  - 3.2.6 Formát metadat . . . 17
- 3.3 Databáze . . . 17
  - 3.3.1 Lokální databáze zálohy . . . 18
  - 3.3.2 Databáze serveru . . . 19
- 3.4 Zabezpečení přístupu . . . 20
  - 3.4.1 Nastavení vstupního hesla . . . 20

3.4.2 Přihlášení pomocí vstupního hesla . . . 21
3.4.3 Přihlášení pomocí ikony na liště . . . 22
3.4.4 Práce na serveru a ukládání informací o přístupu . . . 22
3.5 Šifrovací klíč . . . 24
3.5.1 Vstup z webového uživatelského rozhraní . . . 24
3.5.2 Vstup z rozhraní příkazové řádky . . . 25
3.5.3 Uložení a použití klíče . . . 26

**4 Vyhodnocení bezpečnostních zranitelností a jejich řešení 29**
4.1 Použitá kryptografie a autentizace . . . 29
4.2 Nalezené zranitelnosti . . . 30
4.2.1 Zranitelností modulu AES Crypt . . . 30
4.2.2 Zneužitelnost dat v databázi serveru . . . 31
4.2.3 Nešifrovaná webová komunikace . . . 32
4.2.4 Dekompresní bomba . . . 32
4.3 Další nalezené chyby . . . 33
4.3.1 Neošetřená výjimka při autentizaci . . . 33
4.3.2 Zvláštní chování webového rozhraní . . . 33
4.3.3 Modul 7z . . . 34
4.4 Shrnutí bezpečnosti . . . 34
4.4.1 Odposlech webové komunikace . . . 34
4.4.2 Útok Man-in-the-middle . . . 36
4.4.3 Citlivé informace v databázi v čitelné podobě . . . 37
4.4.4 Povolení použít slabé vstupní heslo . . . 37
4.4.5 Oslabení klíče se znalostí generující funkce . . . 37
4.4.6 Dekompresní bomba . . . 38

**5 Závěr 39**

**A Příprava dekompresní bomby 41**

**B Pozorování síťové komunikace 43**

**Obsah přiloženého média 49**

# Seznam obrázků

2.1 Rozhraní ikony na liště . . . . . 8
2.2 Blokový formát zálohy . . . . . 10

3.1 Nastavení vstupního hesla . . . . . 20
3.2 Přihlašovací formulář . . . . . 21
3.3 Neošetřená výjimka při změně hesla . . . . . 23
3.4 Nastavení šifrovacího klíče . . . . . 25

4.1 Diagram ilustrace odposlechu . . . . . 36

# Seznam tabulek

3.1 Schéma souboru AES . . . . . 15
3.2 Hlavička souboru ZIP . . . . . 15
3.3 Zápatí souboru ZIP . . . . . 15

4.1 Parametry CVSS metodiky . . . . . 35
4.2 Přehled CVSS ohodnocení . . . . . 38

B.1 Pozorování síťové komunikace . . . . . 43

# Seznam ukázek

3.1 Obsah manifest souboru . . . . . 16
3.2 Obsah souboru se seznamem bloků . . . . . 17
A.1 Příprava dekompresní bomby . . . . . 41

*Chtěl bych především poděkovat svému vedoucímu Ing. Jiřímu Dostálovi, Ph.D. za odborné rady a ochotnou pomoc při zpracování této práce. Dále bych rád poděkoval mé rodině za podporu při studiu.*

# Prohlášení

Prohlašuji, že jsem předloženou práci vypracoval samostatně a že jsem uvedl veškeré použité informační zdroje v souladu s Metodickým pokynem o dodržování etických principů při přípravě vysokoškolských závěrečných prací. Beru na vědomí, že se na moji práci vztahují práva a povinnosti vyplývající ze zákona č. 121/2000 Sb., autorského zákona, ve znění pozdějších předpisů, zejména skutečnost, že České vysoké učení technické v Praze má právo na uzavření licenční smlouvy o užití této práce jako školního díla podle § 60 odst. 1 citovaného zákona.

V Praze dne 11. května 2023 .....

# Abstrakt

Tato práce se zabývá bezpečnostní analýzou zálohovacího nástroje Duplicati. V práci je přiblížena problematika zálohování a je popsán nástroj Duplicati s jeho vlastnostmi. Bezpečnostní analýza nástroje zkoumá formát zálohy, který je v nástroji použit, a prověřuje bezpečnost při práci s citlivými daty. Na základě analýzy bylo nalezeno několik bezpečnostních zranitelností. Identifikované zranitelnosti jsou ohodnoceny metodikou CVSS a jsou navržena opravná řešení.

Práce upozorňuje, že některé použití nástroje může vést k zásadnímu oslabení bezpečnosti s fatálními následky. Nejzásadnější zranitelností je odposlouchávání nešifrované webové komunikace při použití nástroje na síťovém rozhraní. Další kritickou zranitelností nástroje je povolení použít příliš slabé vstupní heslo.

**Klíčová slova** bezpečnostní analýza, zálohovací nástroj, Duplicati, Duplicati 2.0, bezpečnostní zranitelnost

# Abstract

This thesis addresses the security analysis of the Duplicati backup tool. The thesis describes the issue of backup and describes the Duplicati tool and its features. The security analysis of the tool examines the backup format of the tool and verifies the safety of working with sensitive data. Based on the analysis, several security vulnerabilities were found. Identified vulnerabilities are evaluated using the CVSS methodology and corrective solutions are proposed.

The work warns that some use of the tool can lead to a fundamental weakening of security with fatal consequences. The most critical vulnerability is the eavesdropping of unencrypted web traffic when the tool is used on the network interface. Another critical vulnerability in the tool is the possibility of using a password that is too weak.

**Keywords** security analysis, backup tool, Duplicati, Duplicati 2.0, security vulnerability

# Seznam zkratek

| | |
|---|---|
| AES | Advanced Encryption Standard |
| API | Application Programming Interface |
| CGI | Common Gateway Interface |
| CLI | Command Line Interface |
| CRC | Cyclic Redundancy Check |
| CVSS | Common Vulnerability Scoring System |
| FTP | File Transfer Protocol |
| GNU | GNU's not Unix! |
| GPG | GNU Privacy Guard |
| GPL | GNU General Public License |
| GUID | Globally Unique Identifier |
| HMAC | Hash-Based Message Authentication Codes |
| IV | Inicializační vektor |
| LGPL | GNU Lesser General Public License |
| LVM | Logical Volume Manager |
| MITM | Man-in-the-Middle |
| NAS | Network Attached Storage |
| NIST | National Institute of Standards and Technology |
| NTFS | New Technology File System |
| MD5 | Message-Digest algorithm 5 |
| PBKDF2 | Password-Based Key Derivation Function 2 |
| PGP | Pretty Good Privacy |
| REST | Representational State Transfer |
| RC4 | Rivest Cipher 4 |
| RFC | Request for Comments |
| SAN | Storage Area Network |
| SFTP | SSH File Transfer Protocol |
| SHA | Secure Hash Algorithm |
| SSH | Secure Shell Protocol |
| SSL | Secure Sockets Layer |
| TNO | Trust No One |
| TSL | Transport Security Layer |
| UNC | Universal Naming Convention |
| USN | Update Sequence Number |
| UTC | Universal Coordinated Time |
| VSS | Volume Shadow-copy Service |
| WebDAV | Web Distributed Authoring and Versioning |
| XSRF | Cross-site Request Forgery |

# Úvod

V dnešní době digitalizace stoupá cena informací a tím ruku v ruce roste i množství počítačových dat. Pro většinu společností se jedná o naprosto esenciální prvek jejich podnikání. I pro mnohé běžné uživatele mají jejich osobní dokumenty a fotografie nevyčíslitelnou hodnotu. Řada webových aplikací nebo online služeb využívá značné množství dat a často se spoléhá na své databáze. Samotné programy jsou také někde uloženy. Velkým rizikem je ztráta uložených dat a ta je častokrát katastrofální a zničující. Podle průzkumu společnosti Price Waterhouse Coopers přibližně 70 % malých firem, co zažijí závažnou ztrátu dat, do roka skončí s byznysem [1].

Velmi důležitou prevencí proti ztrátě dat je zálohování. Existuje mnoho různých softwarových nástrojů pro tvorbu záloh. Jedním z nich je zálohovací nástroj Duplicati. Jedná se o open-source software, který umožňuje uživatelům zálohovat data na různé typy úložišť, jako jsou lokální disky, cloudová úložiště a síťová úložiště. Duplicati podporuje šifrování dat pomocí silného kryptografického algoritmu AES-256. Kromě toho umožňuje kontrolu integrity záloh, inkrementální zálohování a má uživatelsky přívětivé rozhraní.

Tato práce má za cíl poskytnout ucelený pohled na bezpečnostní aspekty zálohovacího nástroje Duplicati. Dále také slouží k zvýšení povědomí o problematice zálohování souborových systémů a ochraně dat. Dalším cílem je navrhnout a poskytnout doporučení pro opravná opatření ke zvýšení bezpečnosti aplikace a pomoci uživatelům a vývojářům této aplikace při zajištění ochrany dat.

První kapitola seznámí s problematikou zálohování souborových systémů. Dále ve druhé kapitole bude představen zálohovací nástroj Duplicati, jeho funkcionality a použití. V následující kapitole bude provedena analýza Duplicati, zaměřená na souborový formát záloh, metadata, lokální databázi, kontrolu vstupů od uživatele a využití hesel a šifrovacích klíčů. V poslední kapitole budou popsány nalezené zranitelnosti, a jakým způsobem by mohly být zneužité. Pokud bude třeba, budou navržena opravná opatření k odstranění nalezených zranitelností.

Téma této práce bylo inspirováno diplomovou prací Bezpečnostní analýza Drive Snapshot. Práci zpracoval Michal Bambuch, který také dělal bezpečnostní analýzu zálohovacího nástroje. Pro svoji práci si vybral komerční nástroj Drive Snapshot, který se zaměřuje na zálohování disků pro operační systém Microsoft Windows. [2]

# Kapitola 1

# Problematika zálohování počítačových dat

Zálohování dat je důležitou součástí zabezpečení souborových systémů a ochrany proti ztrátě dat. Tato kapitola vysvětluje, co to záloha je, a popisuje několik pojmů, které jsou se zálohováním spjaté. Na konci kapitoly je uvedeno několik alternativ k nástroji Duplicati.

## 1.1 Základní pojmy

Počítačová data jsou organizována do *souborů* a ty jsou ve stromové struktuře ukládány do *adresářů* neboli *složek*. Tento způsob organizace se označuje jako *souborový systém*. Fyzické úložiště, kam se data ukládají, je zařízení, které se nazývá *disk* (anglicky *drive*). Disk může být rozdělen na *oddíly*. Logická jednotka, která je vytvořena na disku nebo jen na některých oddílech, se jmenuje *svazek* (anglicky *volume*), ale občas se může označit také jako *disk*, pokud to není v kontextu zavádějící. Svazek je formátován do určitého souborového systému.

*Záloha* je kopie počítačových dat, která je většinou uložena na jiném úložišti než originální data. *Obnova dat ze zálohy* znamená využití vytvořené kopie tedy zálohy k nahrazení poškozených nebo ztracených dat. *Zálohování* je praktika, která slouží k ochraně dat před ztrátou tím, že se vytvoří jejich záloha. Zálohováním se někdy chybně označuje archivace dat a synchronizace dat.

*Archivace dat* je proces, kde se komprimují a konzervují data, která nejsou aktivně využívána, ale je důležité je uchovat. Je to řešení pro uchovávání dat po dlouhou dobu [3]. Na rozdíl od zálohování se při archivaci data jen přesouvají a nevytváří se redundantní kopie. Vhodným médiem, na které se archiv vytváří, je takové médium, které má primárně nízkou cenu a dlouhou životnost. Protože se s archivy moc často nemanipuluje, není podstatná rychlost zápisu ani čtení daného média. V praxi se proto často používají optické disky, které mají právě tyto vlastnosti.

*Synchronizace dat* je metoda pro zajištění integrity a koherence dat na různých místech. Synchronizace řeší problematiku práce na jedné množině dat z více míst. *„Klíčový rozdíl mezi zálohováním a synchronizací je, že zálohování je jednosměrný proces, zatímco synchronizace je obousměrný proces kopírování.“* [4] To se potýká s několika výzvami, jako je především aktualizace změn v reálném čase.

V kontextu se zálohou se mluví o primárním a sekundárním úložišti. *Primární úložiště* je to, kde se nacházejí původní data, která se zálohují. *Sekundární úložiště* je místo, kam se záloha vytváří. Toto místo se označuje jako *destinace zálohy* nebo jednoduše *úložiště zálohy*. Sekundárních úložišť může být více.

Zálohování se dá dělit podle režimu systému na online a offline zálohu. Při *online záloze* probíhá proces zálohování za chodu systémů. To přináší několik úskalí, jako je uzamčení a vyhrazení

souborů některými procesy. Na druhé straně *offline záloha* se provádí, když systém neběží. Zavádí se na začátku spuštění počítače předtím, než začnou běžet ostatní procesy. Alternativně se může využít zavedením externího zařízení, které zálohu provede.

## 1.2 Metody zálohování

Je více druhů metod vytváření záloh než prostá kopie celého vzoru. Základními třemi druhy těchto metod je *úplná záloha*, *rozdílová záloha* a *inkrementální* neboli *přírůstková záloha* [5]. Dalšími možnostmi je například reverzní přírůstková záloha a průběžná záloha. Tato sekce vysvětlí a popíše základní tři metody.

### 1.2.1 Úplná záloha

Základem je metoda *úplné zálohy*, na které staví ostatní metody. Při vytváření zálohy se klonují úplně všechna data. Záloha je potom identická jako původní data. Výhodou je minimální čas při obnově dat, ale doba při vytváření je naopak delší. Největším nedostatkem je velká náročnost na úložné místo a je velmi nepraktická pro udržování více verzí, které odpovídají různým časovým bodům.

### 1.2.2 Rozdílová záloha

Střední cestou mezi úplnou zálohou a inkrementální zálohou je *rozdílová záloha*. Tato metoda vytváří zálohu dat pouze provedených změn od provedení poslední úplné zálohy a je tedy nejprve nutné provést úplnou zálohu. Jelikož se zálohují jen změny, tvorba zálohy je rychlá. Při obnově dat ze zálohy se postupuje nejprve obnovou úplné zálohy a poté obnovou poslední rozdílové zálohy. Jednou za čas je nutné udělat úplnou zálohu, aby se zmenšila velikost dat při tvorbě rozdílové zálohy.

### 1.2.3 Inkrementální záloha

*Inkrementální* neboli *přírůstková* metoda vytváří zálohu dat pouze provedených změn od poslední provedené zálohy. Na začátku je také nutná prvotní úplná záloha. Na rozdíl od rozdílové zálohy objem dat jedné zálohy je minimální a je tedy velmi rychlá při tvorbě záloh. Na druhou stranu při obnově dat se musí nejdříve obnovit úplná záloha a poté úplně všechny inkrementální zálohy, což dělá obnovu velmi pomalou. Tato metoda neprodukuje téměř žádná duplicitní data, ale pro obnovu dat je důležité udržovat všechny zálohy.

## 1.3 Plánování a verzování záloh

Při aktivním používání a upravování dat je nutné zálohovat data pravidelně. Katastrofa ztráty dat může přijít každý den. Důležitou součástí tvorby záloh je automatizace, aby se minimalizovaly chyby při zálohování a nezapomínalo se na vytváření zálohy. Zálohovací nástroje nám tuto část usnadňují pomocí různých plánovačů a automatického časového spouštění. Frekvence tvorby zálohy se odvíjí od spousty skutečností například cennost dat, objem dat a frekvence změn. Vzhledem k náročnosti replikace dat je nutné i vhodně časovat, kdy se záloha vytvoří, jestli přes den nebo jestli o víkendu v noci. Jak vhodně plánovat zálohování podrobněji popisuje kniha The Backup Bible [6].

S častým zálohování přichází otázka udržování verzí záloh podle času vytvoření. Záloha umožňuje jednoduchou implementaci *verzování*[1] a častokrát se tak i používá. Stačí si jen udržovat

[1] *Verzování* je uchovávání změn a historie dat [7].

starší zálohy a s využitím metod uvedených v sekci 1.2 jsou jednotlivé verze zálohy menší. U inkrementální zálohy se verze vytvářejí samovolně. Alternativně se může použít strategie *rotování záloh*. Existují tři základní schémata: fronta, generační schéma a schéma Hanojských věží [8]. Verzování záloh má výhodu například při nutnosti obnovy starších dat. Může se totiž stát, že se poškozená data propíší i do zálohy a problém se detekuje až později.

## 1.4 Typy úložišť a paměťových médií

Z důvodu snížení rizika ztráty dat současně se ztrátou zálohy není doporučeno ukládat zálohu na stejném místě, kde se nacházejí původní data. Je dobré mít tedy primární úložiště oddělené od sekundárního na odlišné lokaci. Různá média mají různé vlastnosti a jsou odolnější proti odlišným typům poruch. Součástí kvality zálohy je tedy i kvalita média, na kterém se nachází. V následujícím listu jsou popsaná nejběžnější úložiště, kam data lze zálohovat, a jsou srovnány jejich výhody a nedostatky.

**Magnetická páska** Magnetické pásky byly dlouhá léta oblíbeným řešením pro zálohování a archivaci. Jedná se o poměrně levné médium, ale mechanika pro čtení a zápis na pásku je několikanásobně dražší než mechanika pro pevný nebo optický disk. K datům se přistupuje sekvenčně, proto rychlost čtení a zápisu navazujících dat je vysoká.

**Pevný disk** Nejběžnějším paměťovým úložištěm je pevný disk (HDD). Je to velmi levné a dostupné médium. Na druhou stranu je velmi náchylné na poškození a má omezenou dobu životnosti, proto není vhodné pro zálohování. Selhání pevného disku je nejčastější příčinou ztráty dat s 38 procenty všech případů [9].

**SSD disk** Solid-state drive (SSD) je flash disk a je oblíbený kvůli rychlosti čtení a zápisu. Je obecně méně poruchový, jelikož neobsahuje žádné pohyblivé části. Nevýhodou je poměr ceny ke kapacitě úložiště. Je vhodný pro zálohování jen malého objemu dat.

**Optický disk** CD, DVD a Blue-ray disky jsou disky, které využívají laserové paprsky pro čtení a zapisování dat. Nevynikají ani v rychlosti, ani v kapacitě a jsou náchylné na vnější vlivy a podmínky. CD disky mají životnost nanejvýš 10 let a DVD disky pouhé tři roky [10].

**Síťové úložiště** Network-attached storage (NAS, česky *úložiště připojené k síti*) je systém úložných zařízení připojených k síti, aby byly centralizované a neustále k dispozici z více míst. Přináší všechny výhody jako cloudové úložiště, ale jsou rychlejší a důvěryhodnější. Jsou tu velké nároky na administraci, které se vyplatí jen v profesionálním prostředí [11].

**Cloudové úložiště** Cloudové úložiště je služba zprostředkovávající virtuální úložiště na vzdáleném serveru, který je spravovaný třetí stranou [12]. Mezi známé patří Google Drive, Microsoft OneDrive, Dropbox a Mega. Výhodou je jednoduchá použitelnost, dostupnost, nízká cena a sdílení s více uživateli. Nevýhodou může být pomalost přenosu dat a riziko neoprávněného, nelegálního zneužití citlivých dat.

## 1.5 Existující současné řešení

Pro zálohování existuje řada nástrojů, které chrání data a vytvářejí zálohy. Některé operační systémy, jako je Windows a Mac OS, mají vestavěný základní nástroj pro zálohování. Většina poskytovatelů cloudových úložišť nabízí zálohovací nástroje. Komerční řešení nabízejí pokročilejší vlastnosti, ale i bezplatná řešení mají své silné stránky. Podle žebříčku G2 jsou nejpopulárnějšími komerčními nástroj: *„Veeam Backup & Replication™ a Acronis Cyber Protect Cloud“* [13]. Mezi opensource nástroji jsou kromě Duplicati známé například: *„Clonezilla, Amanda Network*

*Backup a Cobian Backup“* [14]. Další zajímavý komerční nástroj je Drive Snapshot, který se zaměřuje na zálohování disků pro operační systém Microsoft Windows a pro který udělal bezpečnostní analýzu Michal Bambuch [2].

# Kapitola 2
# Zálohovací nástroj Duplicati

Duplicati 2.0[1] je opensource software vydaný pod licencí GNU Lesser General Public License (LGPL) [15], proto je zdarma k užití i pro komerční účely. Především se zaměřuje na zálohování souborů a složek na síťové nebo cloudové úložiště, ale umí vytvářet i lokální zálohy. Je převážně napsaný v C#, díky tomu je cross-platformní a je tedy podporován na Windows, Linux i MacOS.

Nástroj je stále ve vývoji a je aktivně vyvíjen. Poslední verze 2.0.6.104 vyšla 15. 6. 2022. Tato práce se bude zaměřovat na stabilnější beta verzi 2.0.6.3, která vyšla 17. 6. 2021. Požadavkem je mít .NET (s minimální verzí 4.7.1), nebo Mono (s minimální verzí 5.10.0), aby mohl být C# kód přeložen. [16]

## 2.1 Funkcionality

Duplicati nabízí spoustu užitečných funkcionalit. I přestože se jedná o bezplatný nástroj, obsahuje několik pokročilých vlastnosti. Tato sekce vyjmenuje a popíše hlavní vlastnosti, kterými se Duplicati liší od ostatních nástrojů.

Pro úplnost je také dobré objasnit, které vlastnosti tento nástroj naopak neumí. Duplicati není ani archivační, ani synchronizační nástroj. I kdyby se tak dal použít, tak není pro tyto operace optimalizovaný. Duplicati neumí dělat offline zálohu, je zaměřen výhradně na online zálohování. Není určen pro zálohování celých disků včetně metadat, informací o oddílech a bootovací sekce[2], ale je určen pro zálohování souborů a složek. Duplicati umí zálohovat lokální data, kde nástroj běží, ale neumí zálohovat vzdáleně uložená data. [17]

### 2.1.1 Inkrementální zálohy, deduplikace a komprese

Pro zálohování je použita metoda inkrementální zálohy. Ta poskytuje výhodu, že už po první záloze, která musí být úplná, další zálohy nejsou náročné na objem dat. To snižuje prostor, čas a zátěž na síti při nahrávání záloh. Velikost je také snížena deduplikací tím, že nástroj ukládá stejné kusy dat pouze jednou. Dalším mechanismem pro snížení konečné velikosti zálohy je komprese dat. Duplicati podporuje kompresní algoritmy DEFLATE a LZMA2 [18]. Jelikož je komprimace dat někdy časově náročná, tak se přeskakuje komprimace souborů známých formátů, které jsou už dobře komprimované. Takovými formáty jsou archivy nebo média, jako je například JPEG a MP3.

[1] Oficiální webová stránka pro Duplicati 2.0 je `https://www.duplicati.com`.

[2] Bootování se označuje proces zavádění systému při spuštění počítače. Bootovací sekce nese informace potřebné pro bootování.

### 2.1.2 Šifrování záloh a zabezpečení

Pro bezpečnou manipulaci přes síť a pro bezpečné ukládání na vzdálených úložištích jsou soubory se zálohou šifrovány. Je používána šifra AES-256, která je považována za velmi bezpečnou. Pro každou zálohu se dá nastavit šifrovací heslo nebo Duplicati nabízí možnost šifrovat přes lokální instanci GNU Privacy Guard (GPG). Nástroj je navržen tak, aby splňoval princip Trust No One (TNO) [17]. Šifruje se lokálně a šifrovací hesla neputují nikam ven.

Pro přístup k nástroji se dá nastavit heslo. Tímto heslem se chrání před neoprávněném vstupem ze síťového rozhraní nebo neoprávněným uživatelům. Jelikož může Duplicati mít přístup k celému disku včetně systémových souborů a souborů všech uživatelů, toto vstupní heslo by měl znát pouze administrátor. Bohužel pro jednu instanci nástroje existuje maximálně jen jedno heslo. Není tak možné mít více uživatelů s přístupem s odlišnými oprávněními nebo evidovat, kdo provedl jaké změny.

### 2.1.3 Rozhraní

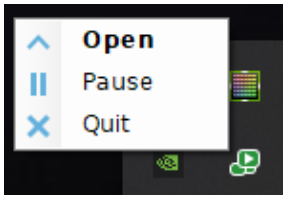
Architektura nástroje je udělána tak, aby nástroj byl přístupný ze síťového rozhraní, a zároveň aby měl přívětivé uživatelské rozhraní. Proto je hlavní částí nástroje webový server, na který se dá připojit ze sítě pomocí běžného prohlížeče. To umožňuje zařízením, která nemají monitor nebo se k nim složitě přistupuje (např. server, NAS aj.), být jednoduše spravována z jiného zařízení v síti, a navíc nepřekáží ani na lokálním zařízení.

Dalšími výhodami webového rozhraní je zálohování na pozadí a snadný vývoj nástroje. Je ale důležité mít na paměti neoprávněný přístup, když se zpřístupňuje úložiště zařízení na lokální síť. Je nezbytné buď používat vstupní heslo, nebo zakázat vzdálený přístup. Server nepoužívá TSL a je tedy zranitelný proti jednoduchým útokům, proto by *nikdy* neměl běžet na zařízení, která jsou přístupná z internetu [19].

Pokud Duplikati je spuštěn na lokálním zařízení, může se ovládat pomocí nativního menu na liště, které ukazuje obrázek 2.1. Jedná se o jednoduché grafické rozhraní, které poskytuje pouze tři možnosti: *otevřít*, *pauza* a *odejít*. Otevření spustí prohlížeč s URL adresou webového rozhraní. Pauza pozastaví všechny úkony například vytváření zálohy a odesílání na vzdálené úložiště. Tlačítko odejít ukončí činnost serveru.

Další způsob, jak Duplicati používat, je na příkazovém řádku skrze CLI, které nástroj nabízí bez omezení. Veškeré možnosti, které jsou přístupné z webového rozhraní, CLI také nabízí. Pro programátory a administrátory to umožňuje použitelnost nástroje v programech nebo v jednoduchých skriptech.

Nástroj lze registrovat jako Windows službu. Služba bude spouštět webový server nástroje při spuštění systému a bude udržovat server zapnutý. Pokud webový server nebude reagovat na pravidelnou kontrolu služby, služba server restartuje. Ve výchozím stavu má služba systémové oprávnění a je proto doporučeno nastavit si vstupní heslo do webového rozhraní [17]. Služba může ale být nainstalovaná i pod lokálním uživatelem.




**Obrázek 2.1** Duplicati – rozhraní ikony na liště se třemi možnostmi.

### 2.1.4 Podpora úložišť

Hlavní předností nástroje Duplicati je bohatá podpora různých druhů úložišť. V základu jsou podporovány všechny připojené disky, ke kterým lze přistoupit UNC cestou. Dále je umožněno nahrávat zálohy skrze síťové protokoly, kterými jsou FTP, SFTP a WebDAV. Duplicati je postaven, aby pracoval s cloudovými úložišti, a umožňuje je používat od velké řady různých poskytovatelů: *„Amazon S3, Azure blob, B2 Cloud Storage, Box.com, Dropbox, Google Cloud Storage, Google Drive, HubiC, Jottacloud, Mega.nz, Microsoft Office 365 Groups, Microsoft OneDrive for Business, Microsoft OneDrive, Microsoft SharePoint, OpenStack Simple Storage, Rackspace CloudFiles, Rclone, Sia Decentralized Cloud, Tardigrade Decentralized Cloud Storage, Tencent COS.“* [17] Dalšími podporovanými cloudovými službami jsou služby s rozhraním OpenStack Object Storage (Swift) a s rozhraním kompatibilním s S3 (Simple Storage Service).

### 2.1.5 Další vlastnosti

Pro pravidelné zálohy je vestavěný plánovač, který spouští zálohování automaticky v naplánovaný čas. Lze vybrat interval, jak často zálohovat nebo který konkretní den a čas. Pokud se v naplánovaný čas nepovedlo zálohu provést, nástroj zahájí zálohování, jakmile je to možné. Pravidelné zálohování nástrojem lze také spouštět pomocí Windows Scheduler nebo na Linuxu pomocí *cron*, které si může uživatel nastavit podle své potřeby.

Duplicati je navržen tak, aby byl co nejvíce bezporuchový a aby byl robustní při případné chybě. Když se přeruší zálohování, může nástroj zahájit zálohování později znovu a zazálohovat vše, co bylo vynecháno. Kontroluje se integrita nahraných záloh, aby se minimalizovala pravděpodobnost, že se zálohy uloží poškozené. Kontrola probíhá tak, že se náhodně stáhne několik nahraných souborů zálohy, obnoví se a porovná se jejich obsah. V případě poškození zálohy nebo lokální databáze má nástroj opravná řešení. Přenos dat po síti je robustní a dokáže se vypořádat se záseky.

Aby mohl Duplicati provést online zálohu musí si poradit s otevřenými soubory a databázemi, které mají svoje soubory neustále uzamčené po celou dobu běhu. Tuto úlohu pro Windows řeší Volume Shadow-copy Service (VSS), který dokáže spolehlivě udělat snapshot[3] NTFS souborového systému. VSS dělá to, že oznámí programům, že bude vytvořen snapshot disku. Programy, které mají nějaké uzamčené a vyhrazené soubory, vyprázdní svoji mezipaměť do souborů a pokračují v činnosti až po vytvoření snapshotu.

Obdobou pro Linux je Logical Volume Manager (LVM), který pracuje velmi podobně jako VSS, ale místo toho, aby programy vyprázdnily mezipaměť, LVM registruje zápisy do paměti a ty potom přidává do snapshotu. Duplicati využívá vytvořený snapshot a z něho vytváří zálohu. Pro používání služby VSS nebo LVM potřebuje mít nástroj systémové oprávnění, protože služba nevyhnutelně umožňuje přístup k celému disku.

## 2.2 Proces tvorby zálohy

Pro důkladnější seznámení s nástrojem Duplicati je dobré pochopit, jakým způsobem jsou zálohy vytvářeny. Duplicati 2.0 zavádí nový formát blokového úložiště pro ukládání záloh uživatelů na vzdálených souborových serverech [20]. Tento unikátní způsob ukládání dat je inspirovaný blokovými úložišti. Blokové úložiště ukládá data po rozdělených částech, takzvaných *blocích* [21]. Toho využívají plošná síťová úložiště (SAN) tak, že bloky ukládají tam, kde je to nejefektivnější, a tak aby mohla vracet data co nejrychleji.

Duplicati také rozděluje všechny soubory po malých blocích s pevně danou velikostí a ukládá stejné bloky pouze jednou, což přirozeně poskytuje deduplikaci. Výchozí velikost bloku je 100 KiB, ale jednotlivé bloky mohou být menší než nastavená velikost bloku. Každý blok je identifikován

[3] Snapshot je neměnný „snímek“ disku nebo jeho části v určitý časový okamžik.

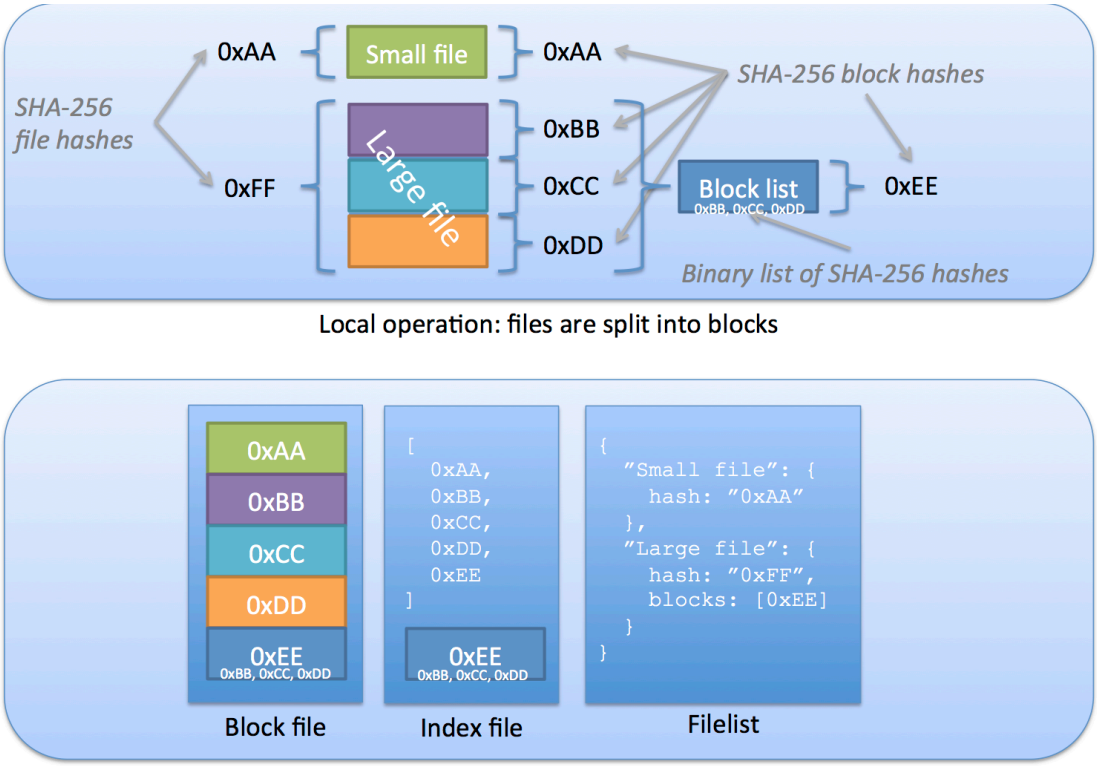
pomocí svého hashe a soubory jsou vyjádřeny jako list hashů. Pro výpočet hashe je použit algoritmus SHA-256 nebo se může nastavit použití algoritmu MD5.

Aby se se zálohou lépe pracovalo, jsou souborové reprezentace seskupeny a odděleny od bloků. Aby u velkých souborů reprezentace nezabíraly příliš místa, jsou listy s hashi ukládány binárně v blocích a potom v reprezentacích stačí udržovat odkazy (hashe) na bloky s listem hashů (viz obrázek 2.2). Pro lepší manipulaci s bloky jsou bloky rozděleny do uskupení. Těmito uskupeními jsou soubory s příponou `.dblock` pro bloky a `.dlist` pro reprezentace. Soubory záloh jsou podrobněji zkoumány při analýze v sekci 3.2 Formát souborů se zálohou.

Aby Duplicati mohl podporovat velkou řadu poskytovatelů úložišť, používá velmi zjednodušené rozhraní na komunikaci s nimi. Rozhraní je omezeno na pouhé čtyři operace: *GET*, *PUT*, *LIST*, *DELETE* [18]. Z toho důvodu není možné uloženou zálohu na serveru upravovat, ale místo toho jen nahrát novou a smazat starou. Rozdělení do bloků snižuje přenos zbytečných dat a přenášejí se jen změny od poslední zálohy.

Bloková architektura záloh umožňuje vytvářet inkrementální zálohy a přitom ukládat jen poslední zálohu. Tím je snížena doba a náročnost obnovy, velikost zálohy a také pravděpodobnost poškození zálohy. Pro vytvoření nové zálohy stačí nahrát uskupení bloků se změnami a nahrát nové uskupení souborových reprezentací s upravenými odkazy. Při obnově dat ze zálohy se nemusí obnovit úplná záloha a postupně všechny inkrementální zálohy, ale stačí mít reprezentace souborů a podle toho stáhnout uskupení s potřebnými bloky, ze kterých je Duplicati schopný zpátky zkonstruovat soubory. Pro verzování stačí udržovat historii uskupení s reprezentacemi, a to umožňuje rotování záloh, které je u inkrementální zálohy téměř nemožné.

Po čase se ukládá zbytečné množství nepoužívaných bloků, proto se musejí čas od času odma-




■ **Obrázek 2.2** Blokový formát zálohy, převzato [18].

závat staré bloky. Také se musejí slučovat uskupení s malým počtem bloků. Pravidelná údržba zvyšuje velikost přenesených dat při tvorbě zálohy, ale uvolňuje místo úložiště. Duplicati se rozhoduje, jestli udělá údržbu podle překročení nastavitelných prahových hodnot a je navrhnut tak, aby se při tom zbavil alespoň jednoho celého uskupení zbytečných bloků.

Pro snížení objemu stažených dat si Duplicati udržuje souborovou strukturu zálohy v lokální databázi. Lokální databáze je v podstatě replika uložené zálohy vyjma datových bloků. Kromě souborových reprezentací je v lokální databázi uloženo, v jakém uskupení se bloky nacházejí a které bloky jsou bloky s listem hashů. Jelikož je lokální databáze redundantní, je tedy postradatelná. Aby při opravě lokální databáze nebylo zapotřebí stahovat náhodné uskupení bloků a zjišťovat, co obsahují, existuje v uložené záloze 3. typ souborů, který má příponu `.dindex`. V těchto souborech jsou uloženy všechny potřebné informace na zrekonstruování lokální databáze kromě těch, které jsou už obsaženy v uskupení se souborovými reprezentacemi. K sestavení lokální databáze postačují jen tyto dva typy souborů.

## 2.3 Zkoumání klíčových částí

Cílem této práce je nalézt a vyhodnotit bezpečnostní zranitelnosti. Před tím je ale potřeba nejprve provést analýzu a identifikovat zranitelnosti. Proto byly pro analýzu vybrány klíčové části nástroje, které by mohly být problematické z hlediska bezpečnosti. Protože nástroj šifruje soubory se zálohou, je dobré prozkoumat, jaký formát mají soubory vytvořené zálohy. Analýza prozkoumá a popíše strukturu a jejich účel. Obdobně budou zkoumána uložená metadata o zálohovaných souborech. Zjistí se, jak nástroj přistupuje k šifrování a jaké případné knihovny používá, aby zajistil šifrování souborů. Nástroj si informace o zálohách drží v lokální databázi. Analýza zjistí, jaké informace se v databázi nachází, jaké tabulky se používají a na co se používají.

Slabým bodem každé šifry je odhalení tajných klíčů. Proto je důležité prozkoumat, jak se s šifrovacími klíči pracuje. Bude zkoumán vstup od uživatele z webového rozhraní i z rozhraní příkazové řádky. Protože je zálohování automatizované, musí si nástroj klíče někam ukládat. Obdobnou citlivou informací je vstupní heslo do webového rozhraní. Bude se zkoumat vstup od uživatele, jak funguje přihlášení a jakým způsobem se vstupní heslo nastavuje. Jelikož nástroj umožňuje přístup bez zadání hesla pomocí ikony na liště, musí se buď vstupní heslo někam ukládat, nebo existuje alternativní přístup k webovému rozhraní. Součástí analýzy bude zkoumání databáze serveru, kam si webový server ukládá nastavení.

# Kapitola 3
# Analýza a rozbor nástroje

Analýza má za úkol prozkoumat nástroj Duplicati a zkontrolovat, jestli řeší bezpečnost a jestli neexistují nějaké bezpečnostní zranitelnosti, které by mohl útočník zneužít. Tato kapitola popisuje postup při analýze funkcionalit vybraných na základě poznatků a zkoumání v předešlé podkapitole 2.3.

Nástroj Duplicati verze 2.0.6.3-beta byl zkoumán na 64bitovém operačním systému Windows 11 Home. Pro analýzu kódu bylo použito vývojářské prostředí Visual Studio 2022[1]. Dalšími použitými nástroji byly Visual Studio Code,[2] DB Browser for SQLite[3] a Wireshark[4]. Pro analýzu byly vytvořeny testovací zálohy s různými nastaveními, které zálohovaly velké množství souborů různých velikostí.

## 3.1 Využívané moduly a knihovny

Duplicati distribuuje práci do několika standardních modulů. To má výhodu nejen usnadnění vývoje a flexibilitu, ale i to, že obnova ze souborů se zálohou se dá přinejhorším provést i bez nástroje Duplicati. Nástroj převážně nespoléhá na systémové knihovny, aby se vyhnul problémům vznikajících z odlišností mezi systémy.

Pro šifrování jsou použity dva moduly: AES Crypt[5] a GNU Privacy Guard (GPG)[6] [18]. Jako výchozí modul je AES Crypt a v nástroji ho reprezentuje C# knihovna *SharpAESCrypt* s verzí 1.3.3. AES Crypt je opensource nástroj, který zajišťuje šifrování jednotlivých souborů pomocí šifry AES-256 a je velmi jednoduchý na použití. AES-256 je bezpečná symetrická šifra a roku 2002 byla přijata jako americký standard [22]. Pro výstupní šifrovaný soubor si AES Crypt implementuje vlastní souborový formát s příponou `.aes`. Tento nástroj má pár známých bezpečnostních zranitelností a zdá se, že není aktivně vyvíjen [23]. Dopad těchto bezpečnostních zranitelností na nástroj Duplicati je přiblížen v podsekci 4.2.1.

Druhý modul používá program GPG, který je implementován podle standardu OpenPGP. OpenPGP je volně dostupný standard podle kryptografického programu Pretty Good Privacy (PGP). Tento program vyvinul Philip R. Zimmermann v roce 1991 za účelem, aby navýšil bezpečnost a soukromí ve společnosti [24]. Program GNU Privacy Guard lze volně používat, upravovat a šířit pod stejnou licencí GNU General Public License [25]. Podle doporučením RFC 4880 implementace OpenPGP buď musí, nebo by měla zahrnovat řadu známých algoritmů jako jsou

[1] Visual Studio 2022 dostupný na `https://visualstudio.microsoft.com/vs/`.
[2] Visual Studio Code dostupný na `https://code.visualstudio.com`.
[3] DB Browser for SQLite dostupný na `https://sqlitebrowser.org`.
[4] Wireshark dostupný na `https://www.wireshark.org`.
[5] AES Crypt dostupný na `https://www.aescrypt.com`.
[6] GNU Privacy Guard dostupný na `https://gnupg.org`.

asymetrické šifry RSA a Elgamal, symetrické šifry TripleDES a AES-128, hashovací algoritmus SHA-1 a kompresní algoritmus ZIP [26]. V programu Duplicati je modul GPG volán jako spustitelný program. To má výhodu toho, že uživatel si může nakonfigurovat, s jakým nastavením se bude program volat. Modul GPG je využit obdobně jako modul AES Crypt a šifruje jednotlivé soubory se zálohou, které potom dostávají příponu `.gpg`.

Podobně jako u šifrováni využívá Duplicati pro kompresi dva moduly: ZIP a 7z. Ve výchozím nastavení komprimuje soubory do formátu ZIP a využívá kompresní algoritmus DEFLATE. Modul ZIP je implementován pomocí C# knihovna *SharpCompress*[7], která má řadu dalších kompresních algoritmů, které ale Duplicati nepoužívá.

Pro lepší a rychlejší kompresi má sloužit modul 7z, který používá LZMA2 algoritmus. Pro tento algoritmus slouží knihovna *managed-lzma*[8]. Modul 7z má řadu zásadních problémů a v aktuální verzi není doporučený [27]. Při analýze byla testována záloha používající tento modul, ale při vytváření zálohy se vždy nástroj neočekávaně ukončil. Modul ZIP je tak v této verzi nástroje jediný modul pro komprimaci souborů, ale tato skutečnost není nikde dokumentovaná.

## 3.2 Formát souborů se zálohou

Nástroj používá tři druhy souborů: *dblock*, *dlist* a *dindex*. Soubory mají zřetězené přípony podle postupně aplikovaných vlastností. Nejprve se soubory komprimují jako archivy typu ZIP nebo popřípadě 7z. Následně se soubory šifrují, pokud je nastaven šifrovací klíč, buď modulem AES Crypt, nebo altelrnativně modulem GNU Privacy Guard a podle toho získají příslušnou příponu `.aes` respektive `.gpg`. Výsledná přípona souboru může vypadat například `.dblock.zip.aes`.

Pozornost byla věnována nejen vlastním souborovým formátům, ale i formátům, které nástroj používá ve výchozím nastavení: ZIP a AES. Popisu každého formátu je věnována samostatná podsekci. Binární soubory byly zanalyzovány v Hex Editoru[9] pro Visual Studio Code. Bylo provedeno prozkoumání schématu a formátu souborů vytvořené nástrojem.

### 3.2.1 Soubor typu AES

Duplicati předává šifrování knihovně *SharpCompress*. Schéma souboru popisuje tabulka 3.1. U číselných hodnot je pořadí bajtů v big-endian. V hlavičce se nachází opakovatelná sekce s dodatkem. Tato sekce se bude opakovat, dokud délka poslední sekce není nulová. Analýzou bylo zjištěno, že nástroj Duplicati nijak nevyužívá dodatkové sekce a zůstávají pouze dvě výchozí vytvořené sekce knihovnou *SharpCompress*. První sekce má délku 33 B, identifikátor „CREATED_BY“ a její obsah je „SharpAESCrypt v1.3.3.0“, který určuje, kdo nebo co soubor vytvořilo. Druhá sekce má délku 128 B a je úplně prázdná, protože slouží pro pozdější zápis nešifrovaného obsahu bez nutnosti úpravy délky souboru. Hlavička má potom stejnou velikost ve všech souborech zálohy o velikosti 268 bajtů.

Po dodatkové sekci následuje inicializační vektor (IV), který slouží společně s šifrovacím heslem k odšifrování sekundárního inicializačního vektoru a klíče. Na konci hlavičky je kontrolní HMAC udržující integritu a autentičnost zašifrovaného sekundárního inicializačního vektoru a klíče. Zbytek informací v hlavičce nemusí být důvěryhodný a je tedy možné neidentifikovatelně měnit verzi souboru. Na konci celého souboru je další HMAC, který zajišťuje integritu a autentičnost šifrované zprávy. Christopher Wellons uvedl důkaz konceptu, že není zajištěna autentičnost bajtu, který určuje velikost posledního šifrovaného bloku (velikost souboru v modulu 16), a tím je narušená integrita šifrované zprávy [28].

[7] Knihovna *SharpCompress* dostupná na `https://github.com/adamhathcock/sharpcompress`.
[8] Knihovna *managed-lzma* dostupná na `https://github.com/weltkante/managed-lzma`.
[9] Hex Editor dostupný na `https://github.com/microsoft/vscode-hexeditor`.

**Tabulka 3.1** Datové schéma souboru AES, odvozeno [29].

| počet bajtů | obsah |
|---|---|
| 3 | identifikátor „AES“ |
| 1 | verze (0x02) |
| 1 | rezervováno (0x00) |
| ... | začátek dodatkové sekce |
| 2 | délka identifikátoru a obsahu dodatku (N + M) |
| N | identifikátor dodatku ukončen nulovým bajtem |
| M | obsah dodatku |
| ... | konec dodatkové sekce |
| 16 | IV k šifrování sekundární IV a klíče |
| 16 | zašifrovaný sekundární IV k šifrování zprávy |
| 32 | zašifrovaný sekundární klíč k šifrování zprávy |
| 32 | HMAC |
| $n$ | zašifrovaná zpráva |
| 1 | velikosti souboru v modulu 16 |
| 32 | HMAC |

**Tabulka 3.2** Hlavička centrálního adresáře ve srovnání s hlavičkou celého ZIP souboru, odvozeno [30].

| počet bajtů | obsah | hlavička souboru |
|---|---|---|
| 4 | identifikátor „PK\1\2“ (resp. „PK\3\4“) | X |
| 2 | tvůrce verze | |
| 2 | minimální nutná verze | X |
| 2 | příznakové bity | X |
| 2 | metoda komprese | X |
| 2 | čas poslední modifikace souboru | X |
| 2 | datum poslední modifikace souboru | X |
| 4 | CRC-32 nekomprimovaných dat | X |
| 4 | velikost komprimovaných dat | X |
| 4 | velikost nekomprimovaných dat | X |
| 2 | délka jména souboru ($n$) | X |
| 2 | délka extra pole ($m$) | X |
| 2 | délka komentáře souboru ($k$) | |
| 2 | číslo disku, kde soubor začíná | |
| 2 | interní atributy souboru | |
| 4 | externí atributy souboru | |
| 4 | celkový počet bajtů po začátek souboru | |
| $n$ | jméno souboru | X |
| $m$ | extra pole | X |
| $k$ | komentář souboru | |

**Tabulka 3.3** Zápatí centrálního adresáře v souboru ZIP, odvozeno [30].

| počet bajtů | obsah |
|---|---|
| 4 | identifikátor „PK\5\6“ |
| 2 | číslo aktuálního disku |
| 2 | číslo disku, kde se nachází začátek centrálního adresáře |
| 2 | počet záznamů centrálního adresáře na aktuálním disku |
| 2 | celkový počet záznamů centrálního adresáře |
| 4 | velikost centrálního adresáře |
| 4 | celkový počet bajtů po začátek centrálního adresáře |
| 2 | délka komentáře ($n$) |
| $n$ | komentář |

### 3.2.2 Soubor typu ZIP

Pro vyvození závěrů o zranitelnosti v sekci 4.2.1 bylo nezbytné pochopit formát souboru ZIP. Na rozdíl od jiných souborů ZIP nemusí mít validní hlavičku na začátku souboru, aby mohl být používaný. Důležitý je *centrální adresář*, který se nachází na samotném konci souboru. Centrální adresář má hlavičku, obsah a zápatí, kde hlavička obsahuje stejné informace jako hlavička ZIP souboru a několik informací navíc. Tabulka 3.2 popisuje hlavičku centrální adresáře a sděluje, které informace se objevují v hlavičce celého ZIP souboru (sloupec napravo). Číselné hodnoty mají pořadí bajtů v little-endian. Soubor, o kterém se v tabulce píše, je uložený a archivovaný soubor uvnitř ZIP souboru.

Kompresi zajišťuje knihovna *SharpCompress* a obdobně jako u šifrování jsou hodnoty hlavičky ponechány bez úpravy. Tvůrce verze je hodnota 20, která podle dokumentace znamená nevyužito, tedy není určená [31]. Příznakové bity určují stav souboru, například jestli je šifrovaný, nastavení kompresního algoritmu a tak podobně. Soubory použité nástrojem Duplicati mají nastaveny příznakové bity na nulu. Metoda komprese je nastavena na hodnotu 8, která znamená metodu DEFLATE. Formát času a data je ve formátu MS-DOS, který má dvousekundovou přesnost času. Více metadat o uloženém souboru formát nedovoluje, proto si musí Duplicati udržovat metadata o souborech jinde (viz podsekce 3.2.6). Cyklická redundantní kontrola CRC-32 zajišťuje integritu uloženého souboru. Extra pole a komentáře nástroj nevyužívá a mají tedy nulovou délku.

### 3.2.3 Soubor zálohy dblock

Jak vysvětluje podkapitola 2.2 soubor typu *dblock* slouží jako uskupení bloků blokové architektury nástroje. Tento typ souboru se občas v nástroji a dokumentaci vyskytuje pod názvem *vzdálený svazek* (anglicky *remote volume*). Soubor má pevně danou velikost podle nastavení, které je ve výchozím stavu 50 MB. Velikost nemusí být, ale splněna přesně a může být menší, i když je snaha nástroje slučovat plně nezaplněné vzdálené svazky dohromady. Duplicati má konzistentní jmennou konvenci. Název souboru typu *dblock* se skládá následovně: „duplicati-b“ + identifikátor v hexadecimálním kódování + „.dblock“. Identifikátor je unikátní náhodné číslo o velikosti 128 bitů, který nic neprozrazuje o uložené záloze.

Jelikož tento soubor je archiv, obsahem jsou soubory. Nachází se zde soubor `manifest`, který obsahuje podstatné informace archivu například verzi formátu, která zajišťuje zpětnou kompatibilitu. Soubor `manifest` je ve formátu JSON a je zobrazen v ukázce 3.1. Ostatní soubory v archivu jsou soubory s bloky, které obsahují různá data. Názvem blokového souboru je jeho hash v base64 kódování.

**Ukázka 3.1** Obsah souboru `manifest`

```
{"Version":2,"Created":"20230301T145838Z","Encoding":"utf8","Blocksize":1024
    00,"BlockHash":"SHA256","FileHash":"SHA256","AppVersion":"2.0.6.3"}
```

### 3.2.4 Soubor zálohy dlist

Soubor typu *dlist* slouží k ukládání adresářové struktury a jmen souborů v ní. Obsahuje souborové reprezentace, podle kterých se Duplicati řídí a podle kterých by mohl skládat bloky dat tak, aby zvládl rekonstruovat zálohovaný obsah. Díky této vlastnosti může soubor sloužit jako verze zálohy. Součást názvu je datum a čas, aby se snadno zjistilo, o jakou verzi zálohy jde. Název se skládá ze tří částí následovně: „duplicati-“ + datum a čas v ISO formátu + „.dlist“. Aby se předešlo problému s časovými zónami je čas uváděn v UTC [18].

Obsahem tohoto archivu jsou tři soubory: `manifest`, `fileset` a `filelist.json`. Soubor `manifest` je stejný jako u soubory typu *dblock*. V souboru `fileset` jsou ve formátu JSON informace o celé verzi zálohy. Jediná informace, kterou nástroj do souboru ukládá, je informace

o tom, jestli je tato verze zálohy buď úplná, nebo částečná. Poslední soubor `filelist.json` obsahuje seznam všech složek a souborů, které do zálohy patří, jejich umístění a hashe, které slouží jako odkazy na bloky s obsahem a na bloky s metadaty.

### 3.2.5 Soubor zálohy dindex

Významem souboru typu *dindex* je snížení toku dat při obnově lokální databáze. Tento soubor obsahuje redundantní informace a je postradatelný, ale bez něho by bylo zapotřebí stahovat náhodné *dblock* soubory, které jsou řádově objemnější, aby se doplnila zde uložená informace. Název se stejně jako u *dblock* souboru odvozuje od svého identifikátoru a skládá se obdobně: „duplicati-i“ + identifikátor v hexadecimálním kódování + „.dindex“, kde identifikátor je náhodné unikátní číslo stejně jako u *dblock* souboru.

Tento archiv obsahuje `manifest` a dvě podsložky `list` a `vol`. Soubor `manifest` je stejný jako u *dblock* souboru. V podadresáři `vol` je soubor s názvem nějakého existujícího *dblock* souboru. Obsahem tohoto souboru je seznam všech bloků příslušného *dblock* souboru ve formátu JSON. Zkrácený obsah souboru je zobrazen v ukázce 3.2. Podsložka `list` obsahuje všechny bloky s listem hashů, na které se odkazují velké soubory složené z více bloků. Pokud žádné takové bloky nejsou, podadresář `list` je vynechán. Níže je uvedená stromová struktura souborového rozložení uvnitř *dindex* souboru.

- `manifest` ..... informace o archivu ve formátu JSON
- `list` ..... adresář s bloky
  - *<hash-base64>* ..... blok s binárním listem bloků
- `vol`
  - `duplicati-b`*<id-hex>*`.dblock.zip` ..... seznam bloků ve formátu JSON

**Ukázka 3.2** Obsah souboru se seznamem bloků v *dblock* souboru

```
{"blocks":[{"hash":"HqIF8hV0rrLLJCI1tt+bHvsZnnpZVohK7vFGogXE2lQ=","size":102
    400},
 Přeskočení dlouhého výpisu…
],"volumehash":"sohQAahfGqobOTEAjyb1n0WQS2kWSrzJgQ10OqkQ3x8=","volumesize":5
    2428321}
```

### 3.2.6 Formát metadat

Při komprimaci se do archivu ZIP nemohou uložit metadata archivovaného souboru. Datum a čas poslední modifikace jsou sice v archivu uložené, ale jsou uložené ve starém MS-DOS formátu, který ztrácí přesnost. Nástroj proto ukládá metadata souborů a složek zvlášť do bloků. Bloky jsou uloženy mezi ostatními bloky zálohy a u reprezentací souborů (v souboru *dlist*) jsou uloženy odkazy na bloky s jejich metadaty.

Obdobně jako u běžných bloků mohou být metadata rozdělena do více bloků, pokud jejich velikost přesáhne velikost jednoho bloku. Potom odkaz u souborových reprezentací ukazuje na blok s listem bloků. Obsahem bloků s metadaty jsou záznamy metadat uložené ve formátu JSON. V metadatech jsou uložená přístupová práva, datum a čas posledního zápisu, datum a čas vytvoření a další vlastnosti. Datum a čas jsou uložené v UTC a pomocí funkce *JsonConvert.SerializeObject* jsou převedené z .NET objektu *System.DateTime* na řetězec. Hodnoty s časem neztrácejí přesnost.

## 3.3 Databáze

Nástroj si vytváří lokální databázi pro každou instanci zálohy a kromě toho si server webového rozhraní nástroje udržuje svojí databázi. Jedná se o SQLite databáze uložené v lokálním úložišti

uživatele. Na Windows jsou ve výchozím stavu umístěny v adresáři `%LocalAppData%\Duplicati\` a na ostatních operačních systémech v adresáři `~/.config/Duplicati`. Umístění databází se může změnit pomocí proměnné prostředí `DUPLICATI_DATA_FOLDER`.

### 3.3.1 Lokální databáze zálohy

V lokální databázi se kromě struktury zálohovaných dat nachází struktura vzdálené zálohy, nastavení a záznamové logy. V této verzi nástroje má databáze interní verzi 11 a obsahuje 19 tabulek a řadu indexů, které optimalizují rychlost častých dotazů. Při analýze bylo nahlíženo do databáze testovacích záloh a do zdrojových souborů zejména do souboru `Schema.sql`[10].

Souborová struktura zálohovaných dat je rozdělena do dvou tabulek *PathPrefix* a *FileLookup*. *PathPrefix* slouží k ušetření místa a obsahuje začátek cesty k souborům. *FileLookup* obsahuje jména souborů a složek, odkaz na blokovou sadu a odkaz na metadata. Sloučením těchto dvou tabulek se získá úplná cesta souboru či složky. Soubory jsou navázané na strukturu zálohy přes tabulku *FilesetEntry* na tabulku *Fileset*. Tabulka *Fileset* zastupuje uskupení souborových reprezentací (*dlist* soubor) a udržuje si informace spjaté s verzí vytvořené zálohy.

Bloková sada, na kterou se soubor odkazuje, je celkový obsah souboru, který je u větších souborů složený z několika bloků. Blokovou sadu reprezentuje tabulka *Blockset* a pomocí tabulky *BlocksetEntry* se může bloková sada seřazeně rozložit na jednotlivé bloky. Úplně všechny bloky jsou uložené v tabulce *Block*. Pro každý blok a blokovou sadu se pamatuje hash, který je v databázi uložen v base64 kódování. Pro větší soubory existují k odpovídajícím blokovým sadám bloky s listem bloků, které jsou obsaženy v tabulce *BlocklistHash*. Jelikož metadata jsou zastoupená také sadami bloků, tabulka *Metadataset* mapuje souborové odkazy na sady bloků s metadaty.

Pro přehled, které soubory jsou na vzdáleném úložišti, se v tabulce *Remotevolume* ukládá název, hash, typ a další informace o souborech zálohy. Aby nástroj mohl správně slučovat bloky nebo odmazávat nepotřebná data, jsou v databázi pomocné tabulky *IndexBlockLink*, *DeletedBlock* a *DuplicateBlock*. Tabulka *ChangeJournalData* usnadňuje zjišťování změn na NTFS souborovém systému, jelikož si poznamenává změny z USN změnového žurnálu. USN (Update Sequence Number) změnový žurnál (anglicky *USN Change Journal*) je žurnál souborového systému NTFS, do kterého se zapisují změny provedené na disku [32].

Lokální databáze slouží také k zaznamenávání událostí a k ukládání logů. Historie událostí a operací nástroje je zapsaná v tabulce *Operation*, na kterou se odkazují například logy nebo uskupení souborových reprezentací (tabulka *Fileset*). Obdobně se eviduje historie dotazů na vzdálené úložiště v tabulce *RemoteOperation*. Celkem jsou čtyři typy dotazů: *GET*, *PUT*, *LIST*, *DELETE* [18]. Tabulka *RemoteOperation* obsahuje odeslaná data v JSON formátu a odkaz na operaci, která dotaz vyvolala. Logy se zaznamenávají v tabulce *LogData* a jejich obsah je uložen v JSON formátu.

Nejzajímavější tabulka z pohledu bezpečnosti je tabulka *Configuration*. Nastavení zálohy se ukládají do této tabulky v podobě klíč-hodnota. Vyskytují se tu jen neměnná nastavení zálohy jako je hashovací funkce, velikost bloku, hash šifrovacího hesla a sůl. Hashovací funkce je funkce použitá na hashování bloků a souborů, nikoliv na hashování hesla. Sůl je složená ve formátu: „v“ + číslo verze soli + „:“ + náhodné 256bitové číslo v hexadecimálním kódování. Pokud není záloha šifrovaná, na místo hashe hesla je uložená hodnota „no-encryption“. Poslední tabulka je tabulka *Version*, ve které je jen jedna hodnota s interní verzí databáze.

[10] Zdrojový soubor `Schema.sql` dostupný na `https://github.com/duplicati/duplicati/blob/v2.0.6.3-2.0.6.3_beta_2021-06-17/Duplicati/Library/Main/Database/Database%20schema/Schema.sql`. [16]

### 3.3.2 Databáze serveru

Na operačním systému Windows je databáze serveru šifrovaná. Duplicati používá šifrování, které je součástí databázové knihovny *System.Data.SQLite*. Tato knihovna používá proudovou 128bitovou šifru RC4 s derivací klíče pomocí hashovací funkce SHA-1 [33]. Tato šifra je označená za slabou a neměla by se používat [34]. Duplicati ji používá jen proto, aby ztížil útok skenováním řetězců na pevném disku [19].

Pro odemčení databáze se může použít možnost `--unencrypted-database`. Klíč k šifrování se skládá z názvu aplikace aktuální verze nástroje a z řetězce „_Key_42“, kde název aplikace v této verzi nástroje je jednoduše „Duplicati“. Klíč se může nastavit pomocí `DB_KEY_ENV_NAME` proměnné prostředí. Pokud není Duplicati k dispozici, je možné databázi odemknout pomocí jednoduchého skriptu[11]. Ve skriptu, který napsal Surlyhacker, je použita stejná knihovna *System.Data.SQLite*, aby se aplikovalo stejné použití šifry [35].

V databázi webového serveru se nachází celkem 12 tabulek. Databáze slouží především k udržování instancí záloh, informací o nich, logů a nastavení. Interní verze databáze je verze 6. Při analýze se používala především odemčená databáze testovací instance serveru a zdrojový soubor `Schema.sql`[12].

Záznamy instancí záloh se nachází v tabulce *Backup*, kde se kromě jména a popisku uchovává lokace lokální databáze a URL adresa vzdáleného úložiště. V této URL je zapsán protokol připojení, cíl a parametry k připojení včetně přihlašovacího hesla nebo OAuth tokenu. Protokolem připojení na začátku adresy je například: `file://`, `ftp://` nebo `googledrive://`. V celé databázi se může na místě identifikačního čísla do tabulky *Backup* objevit záporné číslo $-1$, které značí, že záznam se váže na všechny instance záloh.

Tabulka *Source* obsahuje úplnou cestu ke všem lokacím s originálními daty pro každou instanci zálohy. Filtry pro výběr nebo vyloučení souborů se nachází v tabulce *Filter*. Pomocí těchto dvou tabulek lze určit, které soubory se zpracovávají při tvorbě zálohy a kterou instancí zálohy. V tabulce *Metadata* jsou uloženy informativní vlastnosti o zálohách, které slouží k informativnímu přehledu pro uživatele a neměly by být použity při výpočtu. Mezi tyto informace patří například, jak velká jsou originální data, jak velká je záloha, kolik místa zbývá, počet souborů, jak dlouho trvala záloha, a další.

Tabulka *Option* obsahuje nastavení každé zálohy a také nastavení serveru. Pro nastavení serveru se na místě identifikátoru instance zálohy používá hodnota $-2$. Záznamy této tabulky jsou v uskupení klíč-hodnota. Do nastavení instance zálohy jsou zahrnuta nastavení, například velikost vzdáleného svazku (*dblock* soubor), maximální velikost zálohovaného souboru, počet udržovaných verzí zálohy, doba udržování verzí, modul pro kompresi, modul pro šifrování a šifrovací heslo, které se vyskytuje jako prostý text. Mezi nastavení serveru patří port serveru, síťové rozhraní, povolené zařízení, ale také maximální přenosovou rychlost, priorita vlákna nebo kontrola aktualizace. Také se zde ukládá hash vstupního hesla a jeho sůl, a dokonce prosté heslo a hash hesla přístupu pomocí ikony na liště.

Nastavení plánovače a informace o plánování automatického spouštění je uloženo v tabulce *Schedule*. Je zde uložen čas posledního průběhu zálohy a čas naplánované budoucí zálohy. Čas je uložen ve formátu unixového času. Záznamy logů a chybových logů se nachází v tabulkách *Log* a *ErrorLog* a obsahují informační zprávu, odkaz na instanci zálohy a popřípadě vyvolanou výjimku. Tabulka *Notification* obsahuje upozornění, které uživatel ještě zatím nepotvrdil.

Jako kontejner pro nastavení uživatelského rozhraní slouží tabulka *UIStorage*. Nastavení uživatelského rozhraní je také v podobě klíč-hodnota. Pro evidenci dočasných souborů s delší životností existuje tabulka *TempFile*. V tabulce *Version* je jen jeden záznam a to záznam s interní verzí databáze.

Kompromitace této databáze by mohlo mít fatální dopad. Nejdůležitější je tabulka *Backup*,

[11] Skript dostupný na `https://forum.duplicati.com/t/which-tool-can-open-encrypted-db/2372/17`.

[12] Zdrojový soubor `Schema.sql` dostupný na `https://github.com/duplicati/duplicati/blob/v2.0.6.3-2.0.6.3_beta_2021-06-17/Duplicati/Server/Database/Database%20schema/Schema.sql`. [16]

kde se uchovávají přístupové údaje ke vzdálenému úložišti, a tabulka *Option*, kde se nacházejí v nešifrované podobě šifrovací klíče k zálohám. S těmito údaji by měl útočník úplný přístup k obsahu záloh a při obnově dat ze zálohy by mohl vzdáleně modifikovat obsah na disku.

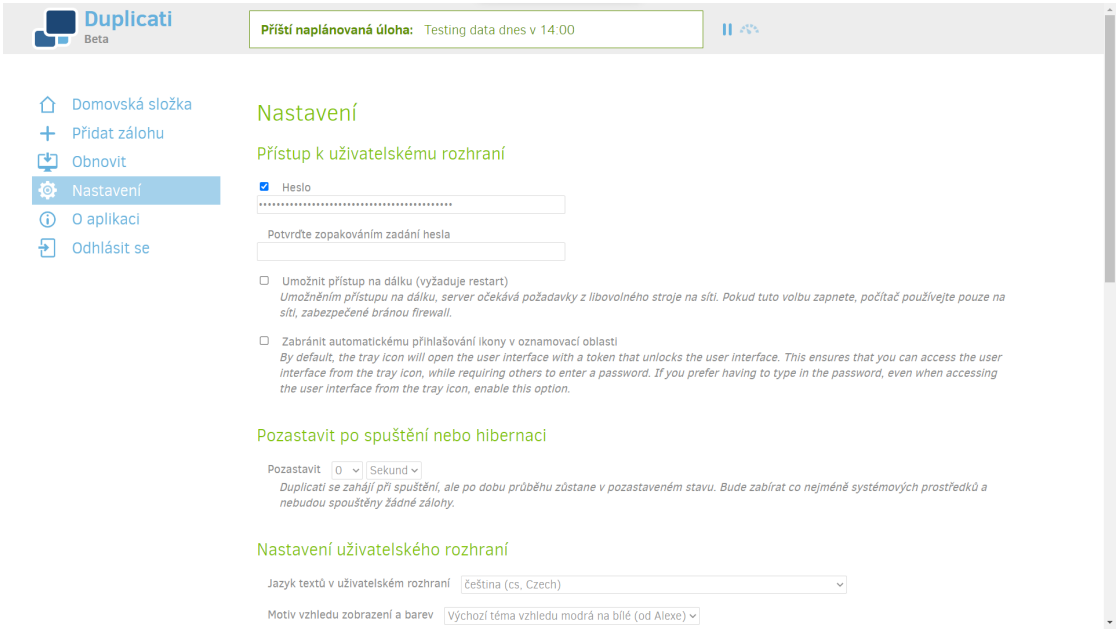
## 3.4 Zabezpečení přístupu

Přístup k nástroji přes webové rozhraní se může zabezpečit vstupním heslem, které není ve výchozím stavu nastavený. V nástroji existuje jen jeden vstupní profil. Smysl vstupního hesla je ochrana před neoprávněným přístupem. Nástroj umožňuje přihlásit se do webového rozhraní bez zadání hesla za pomoci ovládacího rozhraní ikony na liště pro uživatele s přístupem k zařízení. Tato sekce popisuje práci s tímto heslem, zpracování vstupu od uživatele, uložení informace o heslu a přístup pomocí ikony na liště.

### 3.4.1 Nastavení vstupního hesla

Výchozí URL adresa webového rozhraní nástroje Duplicati je `http://localhost:8200`. Webové rozhraní[13] je složeno ze statických stránek a z modulů, které se v nástroji označují jako témata. Uživatel si může na stránce `/theme.html` zvolit téma. V této verzi nástroje je jen jediný modul a to modul „NgAx“ – angularová aplikace. Kořenem této aplikace je stránka `/ngax/index.html` a koncové body, o kterých se bude v práci psát, se řetězí za symbol `#` následovně: základ URL + „`/ngax/index.html#`“ + koncový bod. Data se na server posílají přes REST API a jeho kořenem je `/api/v1`, za který se řetězí koncové body aplikačního rozhraní. Alternativně server navíc poskytuje CGI rozhraní pro některé funkcionality.

Nastavení vstupního hesla se provádí ve formuláři, který je v sekci „Nastavení“ (viz obrázek 3.1). Sekce nastavení se nachází v aplikaci „NgAx“ s koncovým bodem `/settings` a je implementovaná ve zdrojových souborech `settings.html` a `SystemSettingsController.js`.




■ **Obrázek 3.1** Duplicati – formulář pro nastavení vstupního hesla.

[13] Zdrojové soubory webového rozhraní jsou ve složce `webroot/`, `https://github.com/duplicati/duplicati/tree/v2.0.6.3-2.0.6.3_beta_2021-06-17/Duplicati/Server/webroot`. [16]

Součástí formuláře je zaškrtávací tlačítko pro vypnutí a zapnutí zabezpečení pomocí hesla, které je ve výchozím stavu vypnuté. Formulář také umožňuje vypnout automatické přihlášení pomocí rozhraní ikony na liště – ikona v oznamovací oblasti.

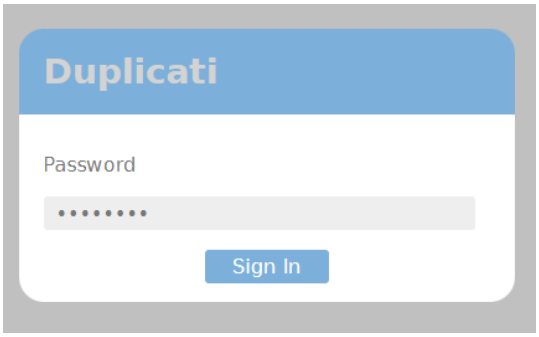
Při načtení stránky se nejprve pošle aplikačnímu rozhraní metodou *GET* dotaz na koncový bod `/serversettings`, aby se získalo stávající nastavení serveru. Součástí tohoto nastavení je hash vstupního hesla a jeho sůl, a dokonce heslo a hash hesla přístupu pomocí ikony na liště. Hash vstupního hesla se použije jako počáteční hodnota pole s heslem, a pokud je hash prázdný řetězec, zruší se označení zaškrtávacího tlačítka. Pokud je zapnuté zabezpečení vstupním heslem, tak se při uložení kontroluje, jestli pole s heslem není prázdné. Dále se kontroluje, jestli se pole s heslem shoduje s ověřovacím polem, který je pod ním. Pokud nějaká kontrola selže, objeví se upozornění a zruší se uložení formuláře. V případě když se nové heslo shoduje s hashem stávajícího hesla, nastavení hesla se neaplikuje a neobjeví se žádné upozornění.

Po kontrole se začne heslo zpracovávat. Vygeneruje se náhodný 256bitový řetězec, který slouží jako sůl. Heslo se dekóduje z UTF-8, poté se dekódované heslo a sůl zřetězí jako bitové pole a výsledek se zpracuje hashovcím algoritmem SHA-256. Sůl a výsledný hash se zakódují v kódování base64 a odešlou se na aplikační rozhraní metodou *PATCH* na koncový bod `/serversettings`. Funkce generující náhodný řetězec a algoritmus SHA-256 jsou použity z knihovny *crypto.js*[14].

### 3.4.2 Přihlášení pomocí vstupního hesla

Přihlášení do webového rozhraní je na statické stránce `/login.html`. Zdrojovým souborem kromě stránky `login.html` je soubor `login.js`. Přihlašovací formulář obsahuje jen pole pro heslo a potvrzovací tlačítko, jak znázorňuje obrázek 3.2. Při potvrzení formuláře se pošle dotaz s požadavkem na získání *nonce* metodou *POST* na CGI rozhraní na adresu `/login.cgi`. Z odpovědi se získá *nonce* a sůl vstupního hesla kódované v base64. Hodnota z pole s heslem, které je kódováno v UTF-8, se zřetězí se solí jako bitové pole a společně se zpracují hashovací funkcí SHA-256. Poté se obdobně vezme *nonce* a za něj se zřetězí výsledek a znovu se zahashuje stejnou funkcí. Výsledný hash se v base64 kódování odešle jako heslo na stejné rozhraní a také metodou *POST*. Použitá kryptografická hashovací funkce SHA-256 je implementována knihovnou *crypto.js*.

*Nonce* slouží k tomu, aby vygenerovaný hash měl jednorázové použití. Když si klient od serveru vyžádá *nonce*, server vytvoří náhodné číslo, číslo se odešle a očekává se jen jedna zpráva s hashem, který je derivovaný z *nonce* a hashe hesla. Pokud přijde více zpráv, tak se akceptuje jen první příchozí zpráva a ostatní zprávy jsou zamítnuty. Kdyby byl kanál odposloucháván, tak odposlechnutá zpráva útočníkovi neumožní se přihlásit. Nicméně jakmile se uživatel přihlásí, vyžádá se dotazem nastavení serveru, ve kterém se nachází hash a sůl hesla.




**Obrázek 3.2** Duplicati – přihlašovací formulář pro vstup do webového rozhraní.

[14] Knihovna *crypto.js* dostupná na `https://code.google.com/archive/p/crypto-js`.

### 3.4.3 Přihlášení pomocí ikony na liště

Po spuštění nástroje se na liště v oznamovací oblasti systému nachází ikona nástroje Duplicati. Ta slouží k ukazování stavu nástroje a má menu na jednoduché ovládání se třemi možnostmi (viz obrázek 2.1 z předchozí kapitoly). Jedna z těchto možností je *otevřít*, která umožňuje otevřít webové rozhraní. Ve výchozím stavu se automaticky přihlásí do webového rozhraní bez nutnosti zadat vstupní heslo. Pro přístup používá buď vlastní heslo, nebo vstupní heslo uživatele podle toho, jak byl program spuštěn.

Spustitelný soubor `Duplicati.GUI.TrayIcon.exe` je pro běžného uživatele hlavní vstupní bod nástroje. Tento program, pokud je zavolán standardně bez přepínače `--no-hosted-server`, spouští proces serveru, na kterém běží webové rozhraní. Z instance serveru si program načte hash uživatelského hesla, které používá k autentizaci. Pokud se program spustí s přepínačem `--read-config-from-db`, načte se konfigurace přímo z databáze včetně hesla, které je odlišné od hesla uživatele. V posledním případě může být heslo programu poskytnuto přímo od uživatele. Heslo se předá přepínačem `--webserver-password` a jedná se o uživatelské heslo, kterým by se uživatel běžně přihlásil. V případě splnění kombinace uvedených možností program se prokazuje posledním určeným způsobem.

Když uživatel zvolí možnost *otevřít* v menu ikony na liště, vytvoří program autentizované spojení se serverem, které potom předá webovému prohlížeči. Nejprve se sestrojí dotaz pro získání soli a nonce, do kterého se přidá hlavička `X-TrayIcon-PasswordSource` s hodnotou „database“ nebo „user“ podle toho, jestli se program prokazuje heslem z databáze respektive vstupním heslem uživatele. Dotaz se odešle metodou *POST* na CGI rozhraní `/login.cgi`. Solení hesla a aplikování nonce se provádí stejně jako v předchozí podsekci 3.4.2. Hashovací funkci SHA-256 poskytuje .NET knihovna *System.Security.Cryptography*. Solení hesla neproběhne, v případě když se program prokazuje hashem získaného z instance serveru.

Na závěr se vytvoří dotaz, do kterého se přidá stejná hlavička jako u prvního dotazu, přidá se cookie s hodnotou nonce a do těla se vloží výsledný hash. Dotaz se odešle stejným způsobem na stejné rozhraní jako ten první. Z odpovědi se přečte cookie s ověřovacím `session-auth` tokenem, který se předá prohlížeči. Prohlížeč se spustí s URL adresou sestrojenou následovně: základ URL + „/index.html?auth-token=“ + ověřovací token. V nastavení nástroje je možnost vypnout automatické přihlášení pomocí ikony na liště, a pokud je aktivní, přeskočí se autentizační proces a vytvoří se URL adresa bez parametru ověřovacího tokenu. To ale nezamezí, aby si program nahrál hesla do paměti. Heslo se načte při spuštění programu a je v paměti programu po celou dobu běhu.

### 3.4.4 Práce na serveru a ukládání informací o přístupu

V této části bude popsáno, jak nástroj pracuje na serveru s přístupovými údaji, které přijímá v dotazech, které byly popsány v předchozích podsekcích. Server zpracovává dotaz v několika krocích. Prvním krokem se ověřuje, jestli adresa hostitele je mezi povolenými adresami. Pokud se v povolených adresách vyskytuje symbol `*`, jsou povoleny všechny adresy. Povolenou adresou je vždy `localhost` a jeho aliasy. Následně v dalším kroku se spouští CGI skripty. Pokud zařízení, na kterém je Duplicati spuštěn, podporuje Synology DSM Authentication,[15] mohou se pomocí nastavení proměnné prostředí `SYNO_DSM_AUTH` použít vestavěné skripty pro ošetření `login.cgi` a `authenticate.cgi` dotazů, jinak server ošetří dotazy sám.

Pro dotaz na `/login.cgi` existují dva případy. V prvním případě se žádá o sůl a nonce a v druhém je pokus o přihlášení. Při požádání o sůl se vygeneruje náhodné 256bitové číslo, které bude sloužit jako nonce. Z instance nastavení, která je od spuštění v aktivní paměti serveru, se připraví vstupní heslo nebo v případě, že dotaz má hlavičku `X-TrayIcon-PasswordSource` s hodnotou „database“, se připraví heslo pro přístup z rozhraní ikony na liště. Nonce společně s heslem se

[15] Synology DSM dostupné na `https://www.synology.com/en-us/dsm`.

zpracují hashovací funkcí SHA-256 stejným způsobem, jak je vysvětleno v práci výše. Generování náhodných čísel a hashovací funkce pochází z knihovny *System.Security.Cryptography*.

Po zpracování se vytvoří nový záznam s nonce a vypočítaným hashem s dobou expirace deset minut. Když se obdrží dotaz na přihlášení, najde se záznam s hashem a zkontroluje se expirace a záznam se odstraní. Potom pokud je obdržený hash shodný jako ten v záznamu, vygeneruje se náhodné 256bitové číslo, které bude sloužit jako autentizační neboli ověřovací token s životností jedna hodina. Pokud je dotaz odeslán na `/logout.cgi`, jednoduše se vezme a odstraní používaný ověřovací token.

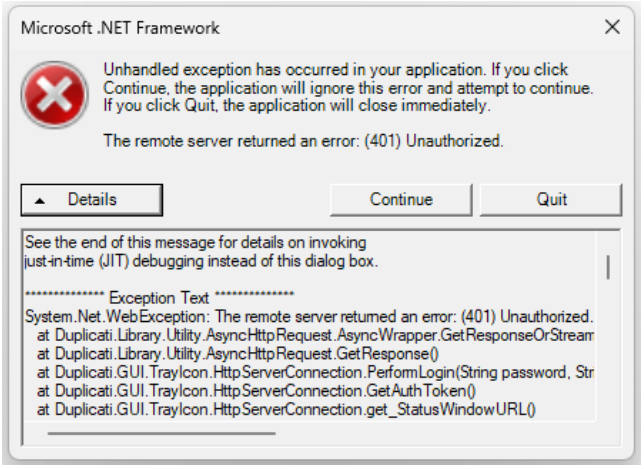
Dalším krokem je kontrola autentizace, které se provádí přes zjištění existence ověřovacího tokenu s platnou dobou životnosti. Kontrola ověření se provádí jen pro dotazy směřující na `/`, na `/index.html` a na REST API rozhraní `/api/v1/` s výjimkou `/api/v1/capthca/`. Pro REST API se navíc kontroluje XSRF token, aby se zablokovaly falešné požadavky z cizích zdrojů. V dalším kroku se zpracovávají dotazy REST API rozhraní. Jeden z koncových bodů `/serversettings`, který poskytuje nastavení serveru, ve kterém je obsažen hash vstupního hesla a sůl (i přístupové údaje přihlášení pomocí ikony na liště).

Metodou *GET* se vyvolá dotaz na databázi na tabulku *Options* a obdrží se všechny položky s nastavením serveru tedy s identifikátorem instance zálohy rovno $-2$ a navíc všechny položky, kde klíč nastavení začíná dvěma pomlčkami „`--`“. Hodnota nastavení s SSL certifikátem se zamaskuje tak, že pokud obsahuje nějaký certifikát, dosadí se hodnota „True“, jinak bude dosazena hodnota „False“. Připravené položky se odešlou jako odpověď.

Metodou *PATCH* se v těle dotazu obdrží nové nastavení, které se nijak nevaliduje, jen se odstraní záznam o SSL certifikátu. Nejprve se aktualizuje instance nastavení v aktivní paměti serveru, až potom se data nahrají do databáze. Jsou použity parametrizované SQL dotazy. Při změně hashe vstupního hesla se předpokládá, že se změnila i sůl a vygeneruje se nové heslo pro rozhraní ikony na liště. Pro heslo se zvolí GUID, globální unikátní identifikátor, a z něho a soli vstupního hesla se vygeneruje hash. Heslo a hash se následně uloží do databáze.

V dalších krocích procesu se mohou nastavit vlastní ovladače na zpracování dotazu, které jsou v této verzi ve výchozím nastavení prázdné. V posledním kroku se zjišťuje, jestli dotaz nežádá o soubor uložený v adresáři `webroot/`, který se nachází ve složce, kde je nástroj Duplicati nainstalovaný. Kontroluje se, zda v cestě URL adresy nejsou obsažené znaky „\“, „..“ a „:“, a pokud jsou obsaženy, dotaz je zamítnut. Pokud cesta začíná symboly „/“, zkrátí se, aby začínala jiným prvním symbolem. Potom je zkontrolovaná existence souboru a dotazovaný soubor se vrátí v odpovědi.

Při změně vstupních údajů se neoznámí program, který spravuje rozhraní ikony na liště. Když




**Obrázek 3.3** Chybová hláška – neošetřená výjimka vyvolaná rozhraním ikony na liště po změně hesla.

se uživatel pokusí zvolit možnost *otevřít* pro otevření webového rozhraní, program se pokusí vytvořit spojení neplatnými přihlašovacími údaji a vznikne neošetřená výjimka. Chybová hláška, která se zobrazí, je ukázána na obrázku 3.3.

## 3.5 Šifrovací klíč

Práce s šifrovacím klíčem neboli šifrovacím heslem začíná od vstupu od uživatele při vytváření nové zálohy. Ten může být pomocí webového rozhraní nebo rozhraní na příkazové řádce (CLI). Při analýze se nahlíželo do zdrojového kódu, aby se zjistil životní cyklus proměnných obsahujících klíč, a zjistilo se, na která místa se klíč ukládá.

### 3.5.1 Vstup z webového uživatelského rozhraní

Šifrovací klíč se zadává ve formuláři při vytváření nebo při úpravě zálohy zároveň s volbou šifrovacího modulu. Formulář je na obrázku 3.4. Tento formulář se objevuje na těchto koncových bodech angularové aplikace: `/add`, `/add-import` a `/edit/<backup-id>`, kde `<backup-id>` je identifikátor zálohy. Zdrojovými soubory formuláře jsou především `EditBackupController.js` a `addoredit.html`. Formulář navíc umožňuje vygenerovat heslo a zkontrolovat jeho sílu. Indikátor síly hesla má pět hodnot: „Nepoužitelné“, „Velmi slabé“, „Slabé“, „Silné“ a „Velmi silné“. Ohodnocení se nezjišťuje pro celé heslo, ale jen pro prvních 100 znaků hesla. Výpočet ohodnocení hesla zajišťuje knihovna *zxcvbn.js*[16].

Generováním hesla se volá funkce ze souboru `AppUtils.js`, kde je implementovaný vlastní algoritmus pro generování hesla. Nejprve se vygeneruje řetězec, který se skládá z náhodných symbolů vybraných z různých znakových sad a je složen následovně: 2 speciální znaky + 2 malá písmena + 2 velká písmena + 2 číslice + 5 až 10 znaků ze všech sad. Speciálními znaky jsou symboly z řetězce „`!@#$\%^&*()_+{}:"<>?[];',./`“. Poté se tento řetězec permutuje tak, že se pro každou pozici vybere náhodná pozice v řetězci. Pro náhodná čísla se používá funkce *Math.random*, která není kryptograficky bezpečná a neměla by se používat v ničem příbuzném s bezpečností [36].

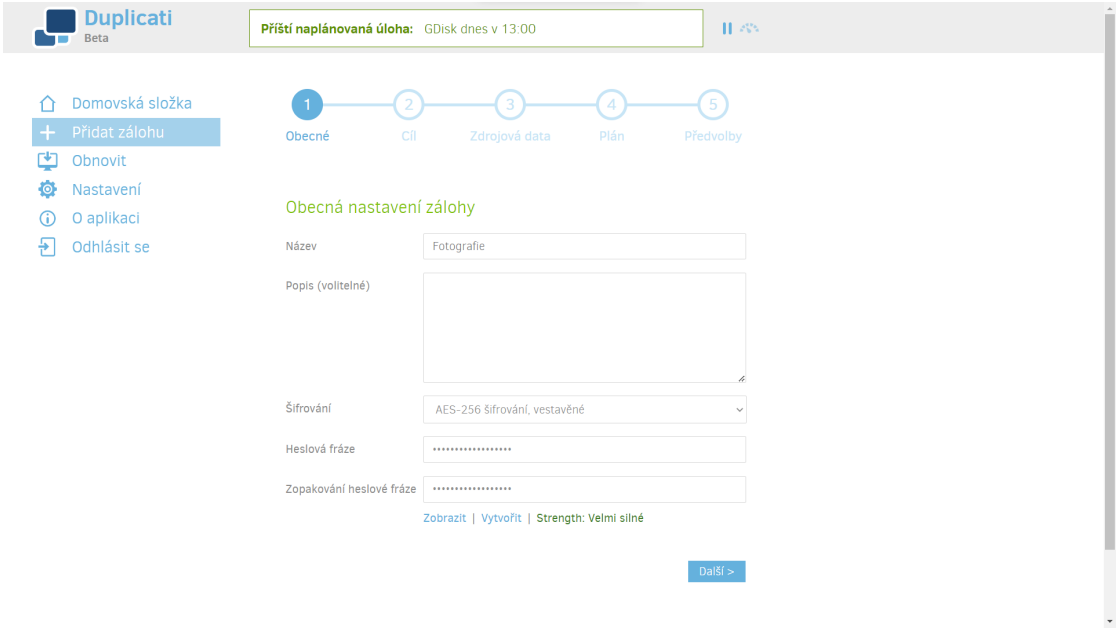
Při permutování se pro převod čísla s plovoucí čárkou používá funkce *parseInt*, která slouží k převodu řetězce na celočíselnou hodnotu. Číslo s plovoucí čárkou je tedy nejprve implicitně převedeno na řetězec, který poté funkce převede na celé číslo. Jelikož funkce ignoruje zbytek řetězce za desetinou tečkou, zdá se, že se číslo převedlo správně, ale nemusí tomu tak být. Například číslo 0,000 000 9 se převede na řetězec „9e−7“, který následně funkce *parseInt* převede na celé číslo 9. Toto chování jenom náhodou nezpůsobuje při permutování chybu, protože náhodně vybraná pozice není nikdy mimo meze, jelikož řetězec má vždy minimálně 13 znaků. Tento defekt nicméně může oslabit náhodnost hesla.

Pokud se upravuje existující instance zálohy, tedy pokud se přistupuje k formuláři z koncového bodu aplikace `/edit/<backup-id>`, nejprve se načtou stávající nastavení instance zálohy do formuláře. Data se získají z REST API metodou *GET* z koncového bodu `/backup/<backup-id>`. Mezi těmito daty je šifrovací klíč, který se použije jako počáteční hodnota pole s heslem i pole s ověřením. Další informací získanou tímto dotazem je i URL adresa vzdáleného úložiště s přihlašovacími údaji.

Při uložení formuláře se provede kontrola hesla a může se objevit upozornění pro uživatele s možností zrušit uložení formuláře. Kontroluje se, zda se pole s heslem shoduje s ověřovacím polem pod ním. Pokud se neshodují nebo je pole s heslem prázdné, uživatel je upozorněn a uložení selže. Jestli je heslo slabé, to znamená ohodnocení „Slabé“ nebo slabší, objeví se varovné upozornění, které nabádá uživatele heslo změnit. Obdobné více kritické hlášení se objeví, pokud se uživatel rozhodne zálohu nešifrovat. Jestliže bylo heslo vygenerováno, při uložení se zobrazí upozornění, aby si uživatel udělal bezpečnou kopii hesla.

[16] Knihovna *zxcvbn.js* dostupná na `https://github.com/dropbox/zxcvbn`.

Není doporučeno měnit heslo při úpravě existující instance zálohy. Duplicati neumí migrovat uloženou zálohu a měnit heslo nebo modul pro šifrování. Pokud se i tak uživatel rozhodl heslo změnit, je při uložení formuláře upozorněn, že změnou hesla může poškodit zálohu a pokud by chtěl heslo i tak změnit, ať provede novou zálohu s jiným heslem. Obdobné hlášení je vyvoláno při pokusu o změnu šifrovacího modulu. Uživatel může všechna upozornění přeskočit a dokončit uložení. Po uložení jsou data formuláře odeslána na REST API na koncový bod `/backups` metodou *POST*. V případě úpravy instance zálohy jsou odeslána na koncový bod `/backup/<backup-id>` metodou *PUT*.




**Obrázek 3.4** Duplicati – formulář pro zadání šifrovacího klíče.

### 3.5.2 Vstup z rozhraní příkazové řádky

Jsou celkem tři způsoby, jak zadat šifrovací heslo při použití rozhraní příkazové řádky. Pomocí přepínače, proměnné prostředí nebo ze standardního vstupu. Nejprve program načte šifrovací klíč z proměnné prostředí `PASSPHRASE`, a pokud je zavolán s přepínačem `--passphrase`, načte a přepíše heslo z argumentu tohoto přepínače. Přepínačem `--no-encryption` se indikuje, že záloha není šifrována a že žádný klíč není. Program, který spravuje rozhraní příkazové řádky, potom předá šifrovací klíč společně s dalšími nastaveními hlavní řídící jednotce nástroje, kterou je třída *Library.Main.Controller*. Tato jednotka umí vykonat všechny činnosti nástroje a je volána jak z CLI, tak z webového rozhraní.

Modul `console-password-input` slouží pro získání šifrovacího klíče ze standardního vstupu, jen když je to potřeba a zároveň když žádný klíč nebyl nástroji poskytnut. Tento modul je vypnutý, když se přistupuje z webového rozhraní. Modul přistupuje ke čtení ze standardního vstupu dvěma způsoby pomocí C# funkce *Console.ReadLine* a funkce *Console.ReadKey*. Funkce *ReadKey* čte stisknuté klávesy na terminálu a může jednoduše zamaskovat, které klávesy byly stisknuty. Na Windows se nahrazuje stisknutá klávesa symbolem `*` a na ostatních operačních systémech se neukazují žádné znaky. Při stisknutí klávesy *escape* se zruší zadávání hesla, klávesou *enter* se dokončí zadávání a nulové bajty jsou při zadávání ignorovány.

Funkce *ReadKey* nemusí vždy fungovat, proto se používá také funkce *ReadLine*, která čte řádku po řádce ze standardního vstupu. Šifrovací heslo je jedna řádka libovolných znaků. Přepína-

čem `--force-passphrase-from-stdin` se vynutí způsob načítání hesla pomocí funkce *ReadLine*. Pokud se vytváří záloha (podpříkaz `backup`), je vyžadované opětovné zadání hesla pro kontrolu. Pokud se hesla neshodují nebo pokud je heslo prázdné, selže zadávání a objeví se chybová hláška.

### 3.5.3 Uložení a použití klíče

Šifrovací heslo z rozhraní příkazové řádky je hlavní části nástroje poskytnuto interně, ale u webového rozhraní se musí nejprve zpracovat dotaz odeslaný na server. Dotazy, jak je popsáno v předešlé podsekci 3.5.1, jsou odesílány na REST API rozhraní a jsou na serveru zpracovávány v řetězci kroků, které jsou rozebrány v podsekci 3.4.4. Dotaz metodou *POST* odeslaný na koncový bod `/backups` vytvoří nový záznam o záloze. Dotaz nesmí mít prázdné tělo. Pokud je požadováno vytvořit dočasnou instanci zálohy, zapíše se záznam do aktivní paměti a GUID se přiřadí jako identifikátor, jinak se záznam zapíše do databáze.

Předtím než se zapíše, jsou data dotazu validována. Jméno, URL a cesta ke zdroji musejí být neprázdné, jméno nesmí už v databázi existovat a URL nesmí obsahovat deset po sobě jdoucích hvězdiček. Kontroluje se, aby parametry jako velikost bloku, velikost vzdáleného svazu nebo četnost zálohování nebyly mimo vyhrazené meze. Je ověřeno, aby buď existoval šifrovací klíč, nebo byla nastavena možnost symetrického šifrování modulu GPG, nebo bylo šifrování vypnuto.

Pokud není přiřazena žádná lokální databáze instanci zálohy, vytvoří se náhodné jméno pomocí .NET objektu *System.Random* s příponou „.sqlite“ a s umístěním v adresáři, které je nastavené pro ukládání databází. Nakonec se uloží instance zálohy do databáze serveru pomocí parametrizovaných dotazů. Do tabulky *Backup* se přidá záznam a získá se identifikátor. Také se smažou, upraví nebo se přidají související záznamy do tabulek *Source*, *Option*, *Filter*, *Metadata* a *Schedule*. Jako odpověď se odešle identifikátor instance zálohy.

Dotaz odeslaný metodou *GET* na koncový bod `/backup/<backup-id>` získá data o záloze s identifikátorem *<backup-id>*. Pokud identifikátor není číselná hodnota, pokusí se najít instanci zálohy mezi dočasnými zálohami, jinak se vytvoří parametrizovaný dotaz na databázi. Podle získané instance zálohy se najde záznam plánovače a jména a cesty zdrojových dat zálohy. Instanci a záznamy jsou odeslány jako odpověď.

Metodou *PUT* se upraví záznam s identifikátorem *<backup-id>*. Identifikátor ani tělo dotazu nesmí být prázdné a musí existovat záznam s tímto identifikátorem. Pokud identifikátor obsahuje pomlčku uprostřed nebo na konci, upraví se dočasná instance zálohy v aktivní paměti serveru, jinak jsou data dotazu validována a poté uložena do databáze. Validace probíhá stejně jako u dotazu na vytvoření záznamu o záloze a navíc nesmí být identifikátor záporné číslo, aby nenastala kolize s vyhrazenými identifikátory pro interní účely (čísla $-1$ a $-2$). Parametrickým dotazem se aktualizuje záznam v tabulce *Backup* a obdobně jako při vytváření se aktualizují související záznamy v dalších tabulkách.

Server pomocí záznamů ve své databázi může volat hlavní část nástroje, aby vykonávala operace nad instancemi záloh, jako jsou například vytvoření zálohy, obnovení dat ze zálohy, vytvoření a oprava lokální databáze a další. Analýza se bude věnovat jen těm částem, kde je práce se šifrovacími klíči. Při vytváření a při opravě lokální databáze se validují hodnoty nastavení zálohy. Pokud chybí sůl, vygeneruje se náhodné 256bitové číslo, převede se do hexadecimálního kódování a zřetězí se za řetězec „v1:“. Pro generování náhodného čísla se používá .NET třída *System.Random*, která nemá kryptografické vlastnosti.

Pomocí šifrovacího hesla a soli včetně řetězce „v1:“ se vytvoří hash. Hashování probíhá v iteracích, kde se vstupy z UTF-8 kódování převedou na binární pole. Za sůl se zřetězí heslo respektive výsledek předchozí iterace. Celkem se provede 1201 iterací a použitá hashovací funkce je funkce SHA-256 z knihovny *System.Security.Cryptography*. Pokud je vypnuté šifrování zálohy, přidá se na místo hashe hesla hodnota „no-encryption“. Pokud se vytvořený hash neshoduje s tím, který už v lokální databázi je a zároveň není zapnutá možnost `--allow-passphrase-change`, validace selže s chybou.

Při oznámení chyby vývojářům nástroje se pro identifikaci chyby odesílá lokální databáze.

Aby nedošlo k úniku citlivých dat, obfuskují se cesty k zálohovaným souborům. Názvy souborů se úplně odmažou a zachová se jen délky cest a podle typu oddělovačů lze zjistit, jestli cesty byly vytvořené ve Windows. Veškeré zprávy záznamů a logů, které obsahují „/“ nebo „:\“, se nahradí za řetězec „ERASED!“. Záznam s hashem hesla se také nahradí touto hodnotou. Protože je sůl přítomná, jen když je šifrování zapnuto, lze zjistit, že záloha je nešifrovaná. Sůl se zachová při odeslání databáze nezměněná. Lokální databáze je potom připravená k odeslání.

Nakonec se šifrovací klíče používají i při šifrování souborů se zálohou. Při šifrování pomocí modulu AES se šifrovací klíče společně s daty předají funkci *SharpAESCrypt.SharpAESCrypt*, která vrátí zašifrovaná data. Při použití modulu GPG se nejprve zavolá program *gpg* s potřebnými přepínači, předá se standardním vstupem šifrovací klíč a poté se standardním vstupem posílají data, která program vrací standardním výstupem zašifrovaná. Na Windows je program *gpg* volán s úplnou cestou k programu, který se nachází v umístění nástroje v podsložce `win-tools/`. Pokud je na systému nainstalovaný *gpg4win*, použije se úplná cesta k tomuto programu. Na ostatních operačních systémech je program *gpg* volán jen jménem programu. Dešifrování je implementováno obdobně.

Kapitola 4

# Vyhodnocení bezpečnostních zranitelností a jejich řešení

V předchozí kapitole byl nástroj Duplicati zanalyzován a byla podrobně popsána práce s citlivými údaji, kterými jsou šifrovací klíče a vstupní heslo. Tato kapitola zhodnotí použité kryptografické algoritmy a identifikuje bezpečnostní zranitelnosti v nástroji na základě provedené analýzy. Zranitelnosti budou ohodnoceny a bude jim přiřazené CVSS skóre.

## 4.1 Použitá kryptografie a autentizace

Pro šifrování používá nástroj Duplicati knihovny nebo programy třetích stran. Tato práce se zabývá jen kryptografií použitou nástrojem Duplicati, proto kryptografie šifrování nebude v práci popsaná. Kryptografie použitá v nástroji je zaměřená na práci s hesly nebo s klíči, proto budou v této podkapitole popsány kryptografické vlastnosti hesel a klíčů. Při vytváření uživatelského vstupního hesla nejsou kladeny žádné restrikce. Uživatel si může zvolit libovolné heslo nenulové délky ve znakové sadě UTF-8. Slabá a příliš krátká hesla jsou také akceptována bez jakéhokoliv upozornění. Podle specifikace NIST by heslo zvolené uživatelem mělo mít délku minimálně 8 znaků [37]. Nástroj neumožňuje uživateli zvolit hash původního hesla ve formátu base64 jako nové heslo.

Pro vytvoření otisku vstupního hesla je použitá metoda solení hesla a hashovací funkce SHA-256. Funkce SHA-256 je nevhodná pro derivaci hesla, protože je rychlá a není náročná na paměť. Pro správné použití by se měla derivační funkce volat v iteracích, zatímco nástroj funkci SHA-256 volá jen jednou. Pro derivaci hesla by měla být použita funkce, která má větší nároky na výpočet, jako například funkce PBKDF2 nebo funkce Balloon a funkce PBKDF2 by měla být volaná minimálně v 10 000 iteracích [37].

Při ověření přihlašovacího hesla se používá nonce. Nonce je v nástroji náhodně generován pomocí .NET kryptograficky bezpečného generátoru náhodných čísel *System.Security.Cryptography.RandomNumberGenerator*. Derivovaný hash vstupního hesla se přes komunikační kanál neposílá a zpracuje se nevratně hashovací funkcí SHA-256, kde se nonce vezme jako inicializační vektor. Stejnou operací se na serveru ověří přijatý hash se vstupním hashem hesla.

Pro alternativní přihlášení pomocí rozhraní ikony na liště je použit stejný proces ověření. Jako heslo se zvolí GUID, globální unikátní identifikátor, a použije se sůl vstupního hesla uživatele. .NET funkce *Guid.NewGuid* generuje náhodné identifikátory s entropií 122 bitů, ale není zaručeno, že se pro generování bude použit kryptograficky bezpečný generátor, proto není doporučeno použít GUID pro kryptografické účely [38].

Pro generování šifrovacích klíčů je implementovaná vlastní funkce. Generování se spoléhá

na pseudonáhodný generátor čísel JavaScriptové funkce *Math.random*. Záleží na implementaci prohlížeče, ale podle specifikace ES6 nemusí být zaručené kryptografické vlastnosti [39]. Při kompromitování stavu pseudonáhodného generátoru čísel může být vygenerovaná hodnota predikovatelná a to i zpětně. Generovaná hodnota může být za určitých podmínek manipulována. *Seed*[1] generátoru se může odvíjet z předvídatelné hodnoty. Vývojářská dokumentace Mozzila upozorňuje, že generátor není kryptograficky bezpečný a pro operace příbuzné s bezpečností by měla být použitá jiná a bezpečnější funkce například *window.crypto.getRandomValues* [36].

Implementace generování čísel vytváří dostatečně dlouhé klíče o délce minimálně 13 znaků. Entropie klíče je snížená, protože vygenerovaný klíč obsahuje alespoň 2 znaky z určitých znakových sad. Při vytváření klíče se volí náhodná délka, což nepřináší žádnou entropii do klíče, ale náhodně ovlivňuje sílu klíče. V implementaci je nevhodně použitá funkce *parseInt*, která může oslabit náhodnost klíče.

Před uložením klíče se kontroluje jeho síla knihovnou *zxcvbn.js*. Knihovna kontroluje heslo před známými frázemi a vzory, které jsou používány při prolamování hesel [40]. Nástroj neomezuje možnosti šifrovacího klíče, jen upozorní uživatele, pokud si chce zvolit slabé heslo. Šifrovacím klíčem mohou být libovolné znaky v UTF-8 kódování. Při vytváření klíče přes CLI může být uživatel omezen o znaky ESC, LF a NUL a není upozorněn na volbu slabého klíče.

Do lokální databáze se ukládá otisk šifrovacího klíče. Pro vytvoření otisku se používá sůl a hashovací funkce SHA-256 s 1201 iteracemi. Sůl je generovaná .NET pseudonáhodným generátorem čísel *System.Random*. Tento generátor čísel není kryptograficky bezpečný a neměl by se v oblasti bezpečnosti používat. Jak bylo už zmíněno hashovací funkce SHA-256 není vhodná pro derivaci hesla a 1201 iterací je v dnešní době příliš nízký požadavek na výpočetní sílu. Stejně jako u vytváření otisku vstupního hesla by se měla také použít funkce PBKDF2 s alespoň 10 000 iteracemi.

## 4.2 Nalezené zranitelnosti

Analýza objevila několik zranitelností a problémů. Tato část práce identifikuje zranitelnosti a popíše je. U zranitelností budou uvedena případná opravná řešení.

### 4.2.1 Zranitelností modulu AES Crypt

Šifrovací nástroj AES Crypt má několik známých zranitelností [23]. Jedna ze zranitelností je narušení integrity zašifrovaného souboru. Ve formátu souboru (viz analýza souborového formátu 3.2.1) je uložena velikost posledního šifrovaného bloku, ale není zaručena její pravost pomocí HMAC. Důkaz konceptu zranitelnosti provedl Christopher Wellons [28]. Je tedy možné nedetekovatelně měnit velikost šifrovaného souboru až o 16 bajtů. Nástroj Duplicati používá nástroj AES Crypt pomocí C# knihovny *SharpAESCrypt*. Soubory, které jsou v nástroji šifrované touto knihovnou, jsou vždy soubory typu ZIP, protože formát zálohy je vždy nejprve komprimován do archivu zvoleného komprimačního modulu a jediný funkční komprimační modul v této verzi nástroje je modul ZIP.

Využitím této zranitelnosti na nástroj se docílí změna velikosti archivu ZIP o maximálně 16 bajtů. Jak bylo popsáno při analýze v podsekci 3.2.2, na konci souboru typu ZIP se nachází zápatí centrálního adresáře, který je esenciální pro dekompresi uložených dat. Pokud se zvětší velikost archivu, nepoškodí se nic, protože za zápatím centrálního adresáře je prostor pro případný komentář. Pokud se naopak změnou velikosti posledního šifrovaného bloku zmenší velikost archivu, poškodí se zápatí centrálního adresáře a tím se poškodí celý archiv.

Nástroj Duplicati s takovým poškozením není schopen udělat nic a při práci se zálohou jen oznámí chybu o poškozeném souboru zálohy. Taková zranitelnost vede jen na odmítnutí služby

[1] *Seed* (česky *semínko*) je hodnota, kterou se inicializuje pseudonáhodný generátor čísel.

(DoS), která s předpokladem, že útočník má přístup k souboru, je sama sebou už dosažitelná například smazáním celého archivu. I když není tato zranitelnost zneužitelná z důvodu souborového formátu ZIP, může se jednat o příčinu chyb. Při rozšíření projektu nástroje Duplicati se může problém snadno zranitelností stát, protože Duplicati může implementovat nové komprimační moduly, kde taková změna velikosti souboru může mít fatální následky.

Další zranitelností nástroje AES Crypt je využití starého souborového formátu verze 0. V automatizovaném procesu, kde útočník má přístup ke čtení části dešifrovaného souboru, může být šifrovaný soubor manipulován tak, aby odšifroval sekundární inicializační vektor a sekundární šifrovací klíč. Útočník by potom mohl přečíst tento klíč a IV, který je postačující k otevření celého šifrovaného souboru [23]. Aplikování této zranitelnosti na nástroj Duplicati není možné, i když pravidelné plánované zálohování je automatizovaný proces, kde by útočník mohl mít potenciálně přístup ke čtení části obnovené zálohy. Útok selže znovu na tom, že se jedná o zřetězené souborové formáty a nástroj Duplicati po odšifrování sekundárního klíče a IV bude mít nevalidní soubor ZIP a operace selže.

Bezpečnostní audit nástroje AES Crypt ukazuje další zranitelnosti oslabující zabezpečení, které by měl nástroj poskytnout. „*Slabá derivace šifrovacího klíče pomocí rychlého hashovacího algoritmu SHA-256 s 8 192 iteracemi. Nadbytečný sekundární inicializační vektor a šifrovací klíč nepřináší žádné bezpečnostní výhody, ale naopak usnadňuje bruteforce útok. Hashování náhodného čísla s identifikátorem procesu, není užitečné ani neoslabuje náhodnost, ale je to zvláštní praktika. Nástroj má několik případů, kdy se chová nedefinovaně. Časově náročné ověření autentizační značky pomocí funkce* `memcmp`.“ [41]

### 4.2.2 Zneužitelnost dat v databázi serveru

Do databáze serveru se ukládají velmi citlivá data a zneužití těchto dat by mohlo mít fatální následky. Nástroj do databáze ukládá potřebné informace k tomu, aby mohl automatizovat veškerý proces vytváření záloh. Jsou zde také uloženy vstupní údaje do webového rozhraní, které může být ovládáno přes síť. Znalost uloženého otisku vstupního hesla stačí k ověření totožnosti pro vstup do webového rozhraní.

V databázi se uchovává klíč, kterým se prokazuje program rozhraní ikony na liště. Tento klíč má stejnou roli a stejnou působnost jako vstupní heslo uživatele. Přestože tento program ve většině případů spouští proces a vlastní instanci webového serveru nebo běží pod stejným uživatelem na stejném zařízení, ověřuje se se serverem přes síť za stejných podmínek jako uživatel. Ověření se může omezit jen na lokální rozhraní nebo interně v programu. Působnost tohoto hesla by potom byla nižší a tím by byla nižší i jeho zneužitelnost.

Kromě vstupních údajů jsou v databázi v prosté nešifrované podobě klíče pro šifrování záloh a přístupové údaje na vzdálené úložiště, kde se zálohy nachází. S těmito informacemi může odkudkoliv na síti měnit obsah záloh, pokud jsou zálohy uloženy na vzdáleném úložišti. Po opravě lokální databáze a následné zálohy může být skrz vzdálené úložiště čten obsah na disku, protože úplná cesta souborů je také uložena v záloze. Při obnově dat může být přepsáno libovolné místo na disku, kam má Duplicati práva zápisu.

Pokud navíc uživatel v nastavení „Umožnění přístupu na dálku“ povolil přístup všem zařízení (symbol `*`), získá se přístup do webového rozhraní, protože se znalostí otisku hesla je možné se autentizovat. S přístupem do webového rozhraní lze bez omezení vytvářet zálohy a obnovovat data, a tím libovolně modifikovat obsah na disku. Citlivé informace v databázi serveru mohou umožňovat vzdálené čtení a zápis do souborů se zálohou, čtení a zápis na vzdálené úložiště a čtení a zápis na disk, kam má Duplicati přístupová oprávnění.

Databáze je uložená v profilu uživatele, kam se může dostat jen uživatel nebo správce. Útočník s fyzickým přístupem může skenováním pevného disku zneužít informace v databázi. Na Windows je databáze chráněná slabou šifrou RC4, která zvyšuje náročnost útoku, ale neznemožňuje ho a se znalostí hesla, které je ve výchozím stavu známé, nepředstavuje téměř žádnou ochranu.

### 4.2.3 Nešifrovaná webová komunikace

Webové rozhraní, které nástroj poskytuje, ve výchozím nastavení komunikuje se serverem přes nezabezpečený protokol HTTP. Nástroj nicméně podporuje nastavení SSL certifikátu, ale to vyžaduje znalost tvorby certifikátů. Manuál ani nástroj důležitost tohoto nastavení nijak nevyzdvihují. Jediná zmínka v manuálu ohledně TSL/SSL zabezpečení je v sekci pro pokročilé uživatele v kapitole „Other Command Line Utilities“ mezi množinou přepínačů webového serveru [17].

Komunikace se serverem může být nepozorovaně odposlechnutá. Některé dotazy vracejí velmi citlivá data, jako jsou vstupní údaje, šifrovací klíče nebo přihlašovací údaje na vzdálené úložiště (viz příloha B). Server by neměl tyto údaje odesílat v běžných dotazech a při potřebě vyměnit tyto údaje by měly být zavedené ochranné postupy. Například při přihlášení se používá technika hashování pomocí nonce. Pokud je potřeba předat informaci nezměněně například při nastavení vstupního hesla, měla by se zpráva šifrovat například asymetrickou šifrou nebo by se měl zajistit bezpečný komunikační kanál.

Odposlechem je útočníkovi umožněno provádět stejné operace jako pomocí informací v databázi serveru, jak je popsáno v předchozí podsekci 4.2.2. Odposlech síťové komunikace je velmi jednoduchý útok a nevyžaduje žádné znalosti. Pokud uživatel změní nastavení „Umožnění přístupu na dálku“, bude komunikace vystavena na síťové rozhraní, odkud může být komunikace prostřednictvím nějakého DNS útoku přesměrovaná na vnější síť.

Na nezabezpečeném protokolu je možné provést i pokročilejší útoky, jako je Man-in-the-middle (MITM) útok. Protože má nástroj rozhraní příkazové řádky, poskytuje uživateli exportování instance používané serverem do formátu argumentů příkazu, které se přidávají za název spustitelného programu na příkazové řádce. Útočník může narušit komunikaci se serverem a podstrčit uživateli vhodně upravený řetězec argumentů, který neznalý uživatel spustí se svými oprávněními za účelem, aby spravoval instanci zálohy z rozhraní příkazové řádky.

Obdobně nástroj poskytuje exportování konfigurace instance zálohy a její stažení do počítače. Útočníkovi se může povést modifikovat stažený soubor tak, aby se ho neznalý uživatel mohl pokusit rozběhnout jako spustitelný program. Ve webovém rozhraní jsou externí odkazy, které odkazují na Facebook, na GitHub a na darovací Donate službu. Útočník může změnit tyto odkazy na falešné, phishingové stránky a ukrást přihlašovací údaje do těchto služeb. V případě Donate odkazu může podvržená stránka umět ukrást finanční částku nebo bankovní údaje.

### 4.2.4 Dekompresní bomba

Nástroj používá soubory typu ZIP komprimované pomocí algoritmu DEFLATE. S přístupem k souborům se zálohou, které mohou být například nešifrované, je možné vytvořit takzvanou DEFLATE bombu. Tato bomba spočívá v principu, že pokud jsou data při kompresi zmenšená, lze při dekompresi data zvětšit mnohonásobně a tím zahltit systém. Maximální faktor zvětšení, který je při dekompresi souboru ZIP možný, je 1100 % [42]. To znamená, že pro jeden napadený soubor zálohy s běžnou velikostí 50 MB se dekomprimuje na data o velikosti 550 MB.

Kromě DEFLATE bomby lze pomocí blokové architektury zálohy sestrojit daleko efektivnější bombu. Pokud je zálohovaný soubor složený z více bloků, bude odkazovat na blok s listem bloků, do kterých je soubor rozložen. Protože ve výchozím nastavení blok je velký 100 KiB a jeden odkaz má velikost 32 B, jeden blok s listem bloků může adresovat až 3200 bloků a tedy zastupovat soubory do velikosti 327,68 MB. Pokud je soubor větší než 327,68 MB, reprezentace odkazuje na více bloků s listem bloků. Soubor může odkazovat na jeden a ten samý blok.

Na složení bomby stačí tři bloky: jeden jako list bloků, druhý jako náhodná data a třetí pro metadata. Blok s metadaty může být úplně minimální s velikostí řádu nižších stovek bajtů. Blok s náhodnými daty by měl naopak být co největší tedy 100 KiB velký. Blok s listem bloků bude také maximálně velký a plný odkazů na blok s náhodnými daty. Spolu s `manifest` souborem budou tvořit malý *dblock* soubor zálohy. V souboru *dlist* se upraví soubor `filelist.json`, kde se nějaké souborové reprezentaci přidá do položky „blocklists“ několik čárkou oddělených odkazů

na blok s listem bloků.

Za každý přidaný odkaz na tento soubor vzroste výsledný soubor o 327,68 MB, kde jeden odkaz má celkem velikost 47 bajtů. Faktor zvětšení je tedy 697 191 489 % a základní velikost je přibližně 200 kB. Po kompresi ZIP bude nachystaná bomba ještě menší, protože se spousta dat duplikuje. V příloze A je znázorněn postup pro sestrojení dekompresní bomby. Bomba byla otestována a zranitelnost je proveditelná. Uživatelé by měli svoje zálohy šifrovat, aby nikdo nemohl modifikovat soubory se zálohou a bombu do nich nastražit. Při změně souborů se zálohou by měl nástroj varovat uživatele na toto riziko. Pro zabránění funkčnosti bomby, by se musel změnit formát zálohy. Nástroj by mohl limitovat maximální možnou velikost obnovených souborů, aby se předešlo riziku zahlcení systému.

## 4.3 Další nalezené chyby

Analýza odhalila několik dalších nedostatků nástroje Duplicati, které nejsou bezpečnostními zranitelnostmi a nepředstavují nutně hrozbu. Do této práce budou zahrnuty, protože v určitých případech mohou nějakou bezpečnostní zranitelnost skrývat nebo mohou být příčinou ke vzniku zranitelnosti.

### 4.3.1 Neošetřená výjimka při autentizaci

Po změně vstupního hesla se změní přístupové heslo rozhraní ikony na liště a změny se propíší do databáze. Protože program spravující rozhraní ikony není synchronizovaný s aktuálním stavem vstupních údajů v databázi nebo v instanci běžícího webového serveru a protože v případě jejich změny není nijak informovaný, existuje stav, kdy bude mít zastaralé vstupní údaje. Při pokusu o přihlášení s těmito údaji pochopitelně selže. Vyvolaná chyba není ale v programu ošetřena (viz obrázek 3.3) a výjimka může program ukončit a to může zapříčinit pád webového serveru. Webový server pracuje s citlivými informacemi a při jeho pádu může dojít ke ztrátě dat nebo k jejich úniku, pokud .NET *garbage collector* nebude schopný paměť správně vyčistit.

Odchycení výjimky se dá snadno spravit. Výjimka nicméně dokazuje, že program nakládá se vstupními údaji neefektivně a možná i nešetrně. Program načítá vstupní údaje hned při spuštění a udržuje si je v paměti programu po celou dobu běhu. To není nutně chybné, ale při kompromitaci operační paměti to může být závažné. Program by proto měl mít údaje v paměti jen po nezbytně dlouhou dobu a načítat je v případě potřeby. Tím by se i vyřešil problém se zastaráním údajů při jejich změně. Program neřeší bezpečné odmazání citlivých informací z paměti programu, ale *garbage collector* po ukončení paměť vyčistí automaticky.

### 4.3.2 Zvláštní chování webového rozhraní

V nastavení webového rozhraní se formulář chová zvláštně v několika případech. Nejviditelnější z toho je chování při uložení formuláře, kdy není žádná odezva uživateli, jestli uložení proběhlo správně. Všechny ostatní formuláře ve webovém rozhraní nástroje po uložení přesměrovaly uživatele na hlavní stránku, ale formulář v sekci nastavení to nedělá. Pro uživatele není žádná zpětná vazba ani v podobě nějakého vyskakovacího okna a uživatel se musí domnívat, že uložení proběhlo správně. Takové neintuitivní chování může způsobovat problémy.

Další problém v tomto formuláři je u zaškrtávacího tlačítka pro heslo. Pokud je tlačítko vypnuté a uživatel zadá nové heslo do pole pro heslo a ověření, po uložení se nové heslo nenastaví. Zaškrtávací tlačítko by se mělo buď samo zaškrtnou při zadání vstupu do pole hesla, nebo by nemělo být pole s heslem vůbec viditelné. Může dojít k nedorozumění, kdy se uživatel bude chybně domnívat, že heslo bylo nastaveno, a bude se mylně na zabezpečení spoléhat.

Dalším zvláštním bodem tohoto formuláře je, že nedovoluje uživateli změnit na heslo na takové, které se shoduje s hashem původního hesla ve formátu base64. Uživatel nebude při uložení

nijak upozorněn. Protože by nikdo nepředpokládal takovouto restrikci, uživatel se může mylně domnívat, že změnil kompromitované heslo, a mylně se spoléhat na bezpečnost nového hesla.

### 4.3.3 Modul 7z

Při používání a testování nástroje Duplicati byla použita většina základního nastavení nástroje včetně modulu 7z, který je jedním ze dvou možných komprimačních modulů nástroje. Uživatelský manuál informuje o existenci alternativního komprimačního modulu [17] a vývojářská dokumentace popisuje výhody tohoto modulu nad modulem ZIP [18]. V nástroji se tato možnost nastavuje přes pokročilé nastavení přepínačem `--compression-module=7z`.

Při spuštění není uživatel upozorněn, že 7z modul je pravděpodobně jen experimentální. Pokaždé když byla záloha s komprimačním modulem 7z spouštěna na testovacích datech, tak po zahájení zálohování se nástroj neočekávaně ukončil. Podle otevřeného problému na GitHub stránce to vypadá, že používaná knihovna *managed-lzma* způsobuje obrovské úniky paměti [27]. Pád a úniky paměti mohou zapříčinit bezpečnostní zranitelnost.

## 4.4 Shrnutí bezpečnosti

Zranitelnosti popsané v předešlé podkapitole budou rozděleny po jednotlivých celcích, budou určeny jejich vlastnosti a bude jim přiřazeno CVSS skóre ve verzi 1.3. Common Vulnerability Scoring System (CVSS) je metodika, která umožňuje vyjádřit charakteristiku zranitelnosti a zachytit její vážnost pomocí přiřazené číselné hodnoty [43]. Číselné hodnoty nabývají rozsahu od 0, nejméně závažné, po 10, nejzávažnější, a mohou být převedeny na slovní reprezentace: *„žádné“*, *„nízké“*, *„střední“*, *„vysoké“* a *„kritické“* [44]. Pro výpočet skóre bude použitý online kalkulátor na stránkách NIST[2].

Další vlastnost, kterou metodika CVSS kromě číselného skóre přináší, je takzvaný *vektorový řetězec* (anglicky *vector string*). *Vektorový řetězec* je textová reprezentace metrických hodnot, podle který se počítá skóre zranitelnosti [44]. V tomto řetězci jsou zakomponované charakteristické vlastnosti zranitelnosti, podle kterých mohou společnosti a organizace jednoduše rozpoznat jaká přibližná rizika zranitelnost představuje. Vektorový řetězec by měl být vždy součástí CVSS skóre, aby mělo skóre výpovědní hodnotu.

Tabulka 4.1 ukazuje, jaké metriky jsou ve vektorovém řetězci obsažené a do kterých skupin spadají. Metriky ze *základní* skupiny jsou povinné a vyjadřují základní charakteristiku každé zranitelnosti. *Dočasná* skupina obsahuje metriky vypovídající aktuální stav dočasných vlastností zranitelnosti. Ve skupině prostředí jsou metriky, které měří vlastnosti zranitelnosti v závislosti na prostředí, ve kterém se nachází.

Přehledová tabulka 4.2 na konci kapitoly ukazuje výsledné ohodnocení identifikovaných zranitelností. Metodika CVSS může ještě hodnotit zvlášť zneužitelnost a skóre dopadu, které jsou v tabulce uvedeny. Zneužitelnost se počítá z metrik AV, AC, PR a UI a skóre dopadu z metrik S, C, I a A. Také může být ohodnocena každá skupina zvlášť, ale protože většina zranitelností používá pouze základní skupinu, v přehledové tabulce uvedené nejsou.

### 4.4.1 Odposlech webové komunikace

Pro ilustraci bude popsán rozumně realistický scénář, který znázorňuje diagram 4.1. Uživatel Albert používá nástroj Duplicati na pravidelné zálohování dat ve svém počítači. Protože chce nástroj ovládat z webového rozhraní přes svůj mobil, umožnil vzdálený přístup a zodpovědně si nastavil silné vstupní heslo. Zálohovaná data si ukládá na vzdálené SFTP úložiště. Protože mezi zálohovanými daty jsou soubory Outlook, musí při zálohování použít VSS, která potřebuje

[2]Online CVSS kalkulátor dostupný na `https://nvd.nist.gov/vuln-metrics/cvss/v3-calculator`.

**Tabulka 4.1** Parametry pro vyhodnocení CVSS skóre, odvozeno [44].

| skupina | zkratka | název | popisek |
|---|---|---|---|
| základní (povinná) | AV | Attack Vector | Rozhraní ze kterého lze útok provést. |
| | AC | Attack Complexity | Náročnost provedení útoku. |
| | PR | Privileges Required | Požadované oprávnění. |
| | UI | User Interaction | Potřeba interakce s uživatelem. |
| | S | Scope | Rámec zásahu vzhledem k rámci útoku. |
| | C | Confidentiality | Dopad na důvěrnost. |
| | I | Integrity | Dopad na integritu. |
| | A | Availability | Dopad na dostupnost. |
| dočasná | E | Exploit Code Maturity | Vyspělost techniky zneužití. |
| | RL | Remediation Level | Úroveň stavu opravení. |
| | RC | Report Confidence | Znalost existence a podrobností. |
| prostředí | CR | Confidentiality Requirement | Požadavek na bezpečnost důvěrnosti. |
| | IR | Integrity Requirement | Požadavek na bezpečnost integrity. |
| | AR | Availability Requirement | Požadavek na bezpečnost dostupnosti. |
| | MAV | Modified Attack Vector | Upravení parametru AV v prostředí. |
| | MAC | Modified Attack Complexity | Upravení parametru AC v prostředí. |
| | MPR | Modified Privileges Required | Upravení parametru PR v prostředí. |
| | MUI | Modified User Interaction | Upravení parametru UI v prostředí. |
| | MS | Modified Scope | Upravení parametru S v prostředí. |
| | MC | Modified Confidentiality | Upravení parametru C v prostředí. |
| | MI | Modified Integrity | Upravení parametru I v prostředí. |
| | MA | Modified Availability | Upravení parametru A v prostředí. |

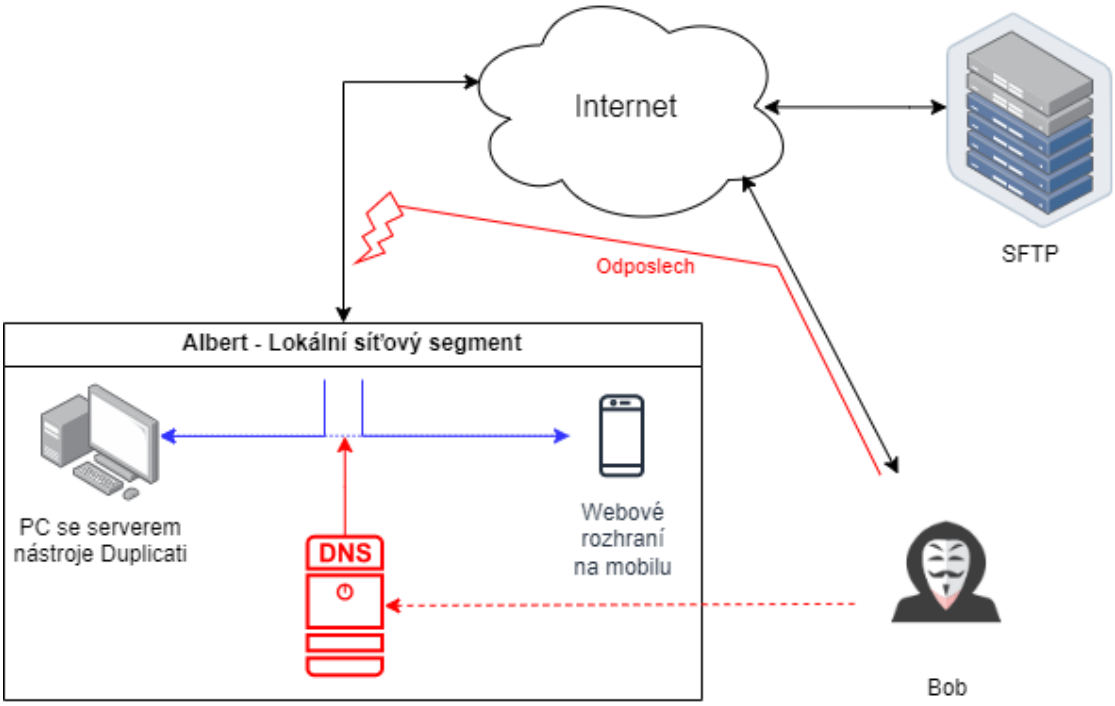
systémové oprávnění nástroje. Útočníkovi Bobovi se povedlo napadnout Albertův domácí síťový segment a je schopen odposlouchávat síťovou komunikaci Alberta.

Bob odposlechl odpověď několika HTTP dotazů z nástroje Duplicati. Protože je komunikace nešifrovaná, nebylo náročné zjistit poslaná citlivá data. Bob se dozvěděl Albertovy přihlašovací údaje na SFTP úložiště a šifrovací klíč zálohy. S údaji může číst a upravovat soubory zálohy a také všechny ostatní soubory, které Albert na úložišti má.

Odposlech je nenáročný útok a lze ho provést přes síť, proto vektor útoku je síť (AV:N) a složitost útoku je nenáročná (AC:L). Není potřeba žádného přihlášení, odposlouchávání jde provést bez jakéhokoliv přihlášení (PR:N) a uživatelské interakce (UI:N). Odposlechem útočník dokáže získat přihlašovací údaje na vzdálené úložiště, pomocí kterých může kdykoliv ze sítě na úložiště číst a zapisovat.

Zranitelná komponenta je webová komunikace nástroje Duplicati a poškozená komponenta je vzdálené úložiště, jde tedy o změnu rámce (S:C). Pokud navíc budou odposlechnuty šifrovací klíče, může útočník údaje použít k libovolné modifikaci a čtení dat na úložišti včetně souborů se zálohou (C:H/I:H). Útok nevyřadí dostupnost nástroje ani systému (A:N). Výsledný vektorový řetězec je „CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:N“, který odpovídá nejvyšší hodnotě CVSS skóre 10,0 – *kritické*.

Nastavení vzdáleného přístupu bez pokročilého nastavení TSL/SSL zabezpečení umožní tuto zranitelnost. Závažnost tohoto problému je kritická a mělo by na ní být více upozorněno jak v manuálu, tak v nástroji. Pro nastavení HTTPS by mohl být poskytnut lehčí způsob nastavení SSL certifikátů. Například by nástroj mohl vydávat sebou podepsané certifikáty, které by uživateli stačily nainstalovat do systému nebo které by mohly být instalovány při instalaci nástroje. Nápravou by bylo, kdyby nástroj nedovoloval nešifrovanou komunikaci v případě vystavení nástroje na síťové rozhraní.




**Obrázek 4.1** Odposlech[3]– modře značená komunikace mezi webovým rozhraním a nástrojem Duplicati je pomocí napadeného DNS serveru odkloněna na internet, kde ji odposlouchává útočník Bob.

### 4.4.2 Útok Man-in-the-middle

Pokud se povede útočníkovi zachytit komunikaci od webového serveru k uživateli, může útočník uživateli poslat modifikovaná data tak, aby nebylo poznat, že ke změně došlo. Může změnit například externí odkazy, tak aby odkazovaly na falešné útočníkovy domény, na kterých budou externí služby vypadat jako ty skutečné. Mezi externími odkazy je odkaz na službu s platbou pro darování finanční částky vývojářům nástroje. Pokud by uživatel zaplatil přes falešný odkaz, mohl by útočník částku odcizit. Alternativně mohou být z jiných podvržených externích služeb vyzrazeny přihlašovací údaje.

Stejně jako u zranitelnosti 4.4.1 vektor útoku je ze sítě (AV:N) a oprávnění není potřeba žádné (PR:N). Protože útočník musí komunikaci zachytit a modifikovat a potom také mít připravené vlastní domény s falešnými stránkami, jde o poměrně náročné provedení útoku (AC:H). Útočník potřebuje, aby uživatel interagoval s jeho falešnými stránkami (UI:R). Zranitelná komponenta je webová komunikace nástroje Duplicati a poškozenými komponentami jsou externí služby (S:C).

Výsledkem útoku je odcizení citlivých informací nebo financí (C:H). Dopad na integritu není žádný (I:N), ale útočník může zachytáváním komunikace částečně odmítnout službu nástroje (A:L). Vektorový řetězec zranitelnosti: „CVSS:3.1/AV:N/AC:H/PR:N/UI:R/S:C/C:H/I:N/A:L“. Výsledné CVSS skóre je 6,9 – *vysoké*.

Nastavení vzdáleného přístupu bez pokročilého nastavení TSL/SSL zabezpečení umožní tuto zranitelnost. Protože jsou podmínky zranitelnosti stejné jako u zranitelnosti 4.4.1, nápravné řešení bude také stejné. Tím je zakázat možnost vzdáleného přístupu, když nejsou nastaveny SSL certifikáty.

[3]Diagram byl vytvořen pomocí online nástroje `https://app.diagrams.net`.

### 4.4.3 Citlivé informace v databázi v čitelné podobě

Pro zneužití informací v databázi serveru musí mít útočník lokální přístup (AV:L) a práva administrátora (PR:H). Data jsou uložená v čitelné verzi a databáze se dá snadno odemknout, pokud je klíč ve výchozím stavu (AC:L). Není potřeba, aby uživatel vykonal nějakou činnost (UI:N). Zranitelná komponenta je databáze serveru a poškozená komponenta je vzdálené úložiště a šifrovaná záloha (S:C). Útočník může se získanými přístupovými údaji číst zálohu a data vzdáleného úložiště (C:H) a může je libovolně upravovat (I:H). Při obnově dat ze zálohy může poškodit systém uživatele (A:L). Hodnocení je 7,4 – *vysoké* a vektorový řetězec: „CVSS:3.1/AV:L/AC:H/PR:H/UI:N/S:C/C:H/I:H/A:L“.

Proto je důležitá obezřetná práce s informacemi uloženými v databázi. Nástroj by měl obeznámit uživatele s rizikem a měl by upozornit, že správce systému má k těmto informacím přístup. Řešením nad rámec tohoto nástroje by bylo externí hardwarové zařízení, která by poskytovalo přístupový klíč do zamčené databáze a databáze by musela být šifrovaná bezpečnou šifrou například AES-256.

### 4.4.4 Povolení použít slabé vstupní heslo

Nástroj povoluje uživateli zvolit příliš slabé vstupní heslo. Heslo může být i jednoznakové. Útok na zabezpečení ochráněné slabým heslem není náročné, stačí udělat nějaký bruteforce útok (AC:L). Ověřovací služba navíc nijak nezamezuje časté zkoušení hesla. Na webové rozhraní se lze připojit přes síť (AV:N) bez potřeby jakéhokoliv oprávnění (PR:N). Napadená komponenta je webové rozhraní a poškozená komponenta je lokální úložiště a vzdálená úložiště (S:C).

Úspěšným útokem se docílí získání přístupu do nástroje, kde jsou citlivé informace, se kterými se lze připojit na vzdálené úložiště, kde útočník může číst a modifikovat soubory se zálohou. Ovládáním si lze vyžádat zálohování a obnovení dat a pomocí toho číst a zapisovat na lokálním úložišti, kde je nástroj Duplicati je spuštěn (C:H/I:H/A:H). Hodnocení zranitelnosti je nejvyšší 10,0 – *kritické* a vektorový řetězec: „CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:H“.

Nástroj by měl zakázat možnost uživateli vytvořit heslo kratší než 8 znaků. Navíc by měl ověřovat, jestli heslo není známou frází nebo se v něm neobjevují nějaké vzory. V ověřovacím procesu se používá rychlá derivace hesla pomocí SHA-256 bez iterací. Měla by se zvýšit náročnost derivace hesla pomocí složitějšího algoritmu například PBKDF2 s 10 000 iteracemi. Server by také mohl po velkém počtu pokusů začít omezovat ověřovací proces.

### 4.4.5 Oslabení klíče se znalostí generující funkce

Při generování klíče není použitý kryptografický generátor náhodných čísel a útočník může zneužít například znalosti přibližného *seedu* generátoru. V lokální databázi je uložený nedostatečně silně derivovaný otisk šifrovacího klíče. Útočník může udělat bruteforce útok s odhadnutými vstupy podle znalosti generátoru. Pro přečtení lokální databáze je potřeba lokální přístup (AV:L) a oprávnění správce (PR:H). Provedení útoku potřebuje znalost generující funkce (AC:H). Útočník nemusí nijak interagovat s uživatelem (UI:N). Napadená a poškozená komponenta je otisk šifrovacího klíče a tedy klíč (S:U). Výsledkem je odhalení klíče, který je citlivou informací (C:H), a dopad na integritu a dostupnost není (I:N/A:N). CVSS skóre je 4,1 – *střední* a vektorový řetězec: „CVSS:3.1/AV:L/AC:H/PR:H/UI:N/S:U/C:H/I:N/A:N“.

Proces generování klíče by se měl spoléhat na kryptograficky bezpečný generátor náhodných čísel. Nevhodně použitá funkce *parseInt* by se měla nahradit za správnou funkci *Math.floor*. Je zbytečné, aby klíč měl náhodnou délku a v klíči by nemusely být požadavky typu 2 symboly od každé znakové sady. Vygenerované heslo by se i tak mělo zkontrolovat, jestli neobsahuje známé fráze nebo nějaké vzory. Otisk klíče uložený v lokální databázi by měl být náročněji derivovaný například algoritmem PBKDF2 s 10 000 iteracemi.

### 4.4.6 Dekompresní bomba

Pokud má útočník možnost upravovat nešifrované soubory se zálohou, může do souborů nastražit dekompresní bombu. Musí mít tedy lokální přístup k souborům se zálohou (AV:L) a musí mít oprávnění správce (PR:H), aby mohl zapisovat do úložiště uživatele. Vytvořit bombu není náročné (AC:L), jak ukazuje příloha A. Pro spuštění bomby je nutné, aby uživatel udělal obnovu dat z napadené zálohy a aby přijal změny v záloze (UI:R). Napadená komponenta jsou soubory se zálohou a ovlivněná komponenta je systém, kam se bomba obnoví (S:C).

Použitím bomby nejsou vyzrazené citlivé informace (C:N). Bomba může přepsat libovolné soubory, kam má nástroj oprávnění (I:H), a může shodit napadený systém (A:H). Součástí práce bylo otestování této zranitelnosti a útok je proveditelný a funkční (E:F). Zranitelnost lze řetězit s některými výše popsanými zranitelnostmi. Hodnocení je 7,2 – *vysoké* a vektorový řetězec: „CVSS:3.1/AV:L/AC:L/PR:H/UI:R/S:C/C:N/I:H/A:H/E:F/RL:X/RC:X“. Zneužitelnost je 1,1, podskóre dopadu je 5,8 a hodnocení dočasné skupiny je 7,2.

Pokud vzdálená záloha neodpovídá lokální databázi, měl by nástroj varovat uživatele na toto riziko. Dekompresní bomba je součástí vlastností blokové architektury zálohy. Aby se znemožnil koncept této zranitelnosti musela by se změnit architektura. Nástroj by mohl mít horní limit velikosti obnovených dat, aby dekompresní bomba nemohla zahltit systém.

**Tabulka 4.2** Přehledová tabulka s CVSS hodnocením zranitelností.

| sekce | zranitelnost | CVSS skóre | | | |
|---|---|---|---|---|---|
| | | zneužitelnost | dopad | **celkové** | **slovní** |
| 4.4.1 | Odposlech webové komunikace | 3,9 | 5,8 | **10,0** | *kritické* |
| 4.4.2 | Útok Man-in-the-middle | 1,6 | 4,7 | **6,9** | *střední* |
| 4.4.3 | Citlivé informace v databázi v čitelné podobě | 0,8 | 6,0 | **7,4** | *vysoké* |
| 4.4.4 | Povolení použít slabé vstupní heslo | 3,9 | 6,0 | **10,0** | *kritické* |
| 4.4.5 | Oslabení klíče se znalostí generující funkce | 0,5 | 3,6 | **4,1** | *střední* |
| 4.4.6 | Dekompresní bomba | 1,1 | 5,8 | **7,2**[4] | *vysoké* |

[4]Celkové skóre zranitelnosti „Dekompresní bomba“ se počítá i s dočasnou skupinou, jejíž podskóre je 7,2.

# Kapitola 5
# Závěr

Cílem této bakalářské práce bylo provedení bezpečnostní analýzy zálohovacího nástroje Duplicati. Práce se v první kapitole věnovala problematice zálohování z teoretického úhlu pohledu. Ve druhé kapitole seznámila s nástrojem Duplicati, jeho funkcionalitami a vnitřními procesy tvorby zálohy nástrojem. Hlavní částí práce byla analýza nástroje v oblasti práce s citlivými údaji a detailní prozkoumání souborových formátů, metadat a databází. Byl popsán životní cyklus klíčů a hesel od zadání uživatelem po uložení a následné znovupoužití. Práce se věnovala použití kryptografie. Byly nalezeny a popsány bezpečnostní zranitelnosti a jejich dopad na nástroj Duplicati. Při popisu byla navržena opravná řešení. Zranitelnosti byly ohodnoceny metodikou CVSS, jak znázorňuje přehledová tabulka 4.2.

Na základě ohodnocení identifikovaných bezpečnostních zranitelností lze směřovat vývoj nástroje ke zlepšení jeho bezpečnosti. Pro nápravu mohou být implementována zmíněné opravné řešení. Výsledky práce byly oznámeny vývojářům nástroje. Pro nové vývojáře a pokročilé uživatele může práce sloužit k detailnímu seznámení s nástrojem ve zkoumaných částech. Práce slouží ke zvýšení povědomí o bezpečnosti a upozorňuje, že některé použití nástroje může vést k zásadnímu oslabení bezpečnosti s fatálními následky. Nejzásadnější zranitelnost je zneužitelná v případě použití nástroje na síťovém rozhraní s nešifrovanou webovou komunikací. Další kritickou zranitelností nástroje je povolení použít příliš slabé vstupní heslo.

Bezpečnostní analýza byla provedena jen na vybraných klíčových bodech nástroje. Rozšířením této práce může být provedení analýzy dalších částí a funkcionalit. Těmi může být analýza zabezpečení oficiálního webu, instalátoru nebo automatického stahování aktualizací. Prozkoumání bezpečnosti při komunikaci se vzdálenými úložišti a práce s OAuth tokeny. Zajímavá by mohla být například práce v oblasti vlastní architektury zálohy na téma kolize identifikačních hashů bloků a jejich exploitace.

# Příloha A

# Příprava dekompresní bomby

Následující bash skript vytvoří dekompresní bombu, která se po zálohování rozbalí na soubor o velikosti 10 GB. Velikost je nastavitelná. Pro vytvoření bloku s obsahem se může namísto `/dev/random` použít `/dev/zero`, potom bude blok s obsahem jednoduše komprimovatelný a výsledný ZIP bude ještě menší. Poznamenejte, že cesta zálohovaného souboru je `/bomb` a že soubor je tedy v kořenovém adresáři Unix systému, kam nástroj nemusí mít přístup.

Celkový kontrolní hash expandovaného souboru následující skript nepočítá. Bomba se dokáže expandovat i přes to, že kontrolní hash nebude sedět, jen nástroj zahlásí upozornění. Pokud by bylo potřeba se upozornění vyhnout, lze si celkový hash souboru napočítat. Alternativně se bomba dá sestrojit pomocí vhodně vybraného velkého souboru a následným zálohováním nástrojem Duplikati, který velký soubor výrazně zmenší do souborů zálohy a ty použít jako bombu.

**Ukázka A.1** Jednoduchý skript, který připravý dekompresní bombu

```
#!/usr/bin/bash


# size je velikost bomby v GB
size=10
n=$(( $size * 1000000000 / 3200 / 102400 ))
size=$(( ($n + 1) * 3200 * 102400 ))

# Blok s náhodnými daty
dd if=/dev/random of=block count=200
sha256sum -b block | xxd -r -p > sha256.bin
block_content=$(base64 sha256.bin | tr '/+' '_-')
mv block $block_content

# Blok s listem bloku
cat $(printf ' sha256.bin%.0s' {1..3200}) > block
rm sha256.bin
listhash=$(sha256sum -b block | xxd -r -p | base64)
block_list=$(echo $listhash | tr '/+' '_-')
mv block $block_list

# Blok s metadaty
printf '\xEF\xBB\xBF{"win-ext:accessrules":"[]","CoreAttributes":'\
'"Archive","CoreLastWritetime":"638132723285693448",'\
'"CoreCreatetime":"638132723086470960"}' > block
```

```
metahash=$(sha256sum -b block | xxd -r -p | base64)
block_meta=$(echo $metahash | tr '/+' '_-')
mv block $block_meta

# Soubor manifest
printf '\xEF\xBB\xBF{"Version":2,"Created":"20230101T000000Z",'\
'"Encoding":"utf8","Blocksize":102400,"BlockHash":"SHA256",'\
'"FileHash":"SHA256","AppVersion":"2.0.6.3"}' > manifest

# Soubor filelist.json
x=$(printf '\xEF\xBB\xBF[{"type":"File","path":"/bomb",'\
'"hash":"%%s","size":%%s,"time":"20230101T000000Z",'\
'"metahash":"%%s","metasize":137,"blocklists":["%%s"%%s]}]')
list=$(printf ',"'$listhash'"%.0s' $(seq 1 $n))
totalhash='44AWKbgzQvkM60GLE9L5LMSAbQQihYNrPS6Dora4Cn0='
printf $x $totalhash $size $metahash $listhash $list > filelist.json

# Soubor fileset
printf '\xEF\xBB\xBF{"IsFullBackup":true}' > fileset

# dlist a dblock archivy
dblock='duplicati-b00000000000000000000000000000000.dblock.zip'
zip $dblock $block_content $block_list $block_meta manifest
zip duplicati-20230101T120000Z.dlist.zip filelist.json fileset manifest
rm $block_content $block_list $block_meta filelist.json fileset manifest
```

# Příloha B

# Pozorování síťové komunikace

Práce prozkoumala zranitelnost nešifrované webové komunikace nástroje Duplicati. Pozorování bylo provedeno v nástroji Wireshark. Výsledkem pozorování je následující přehledová tabulka B.1, která znázorňuje, který dotaz obsahuje které citlivé informace. Citlivé informace byly odposlechnuty ve formátu JSON v odpovědích REST API rozhraní.

**Tabulka B.1** Přehledová tabulka s dotazy, které obsahují citlivé informace.

| URI dotazu | citlivá informace |
|---|---|
| `/api/v1/serversettings` | hash hesla uživatele,<br>heslo ikony na liště,<br>hash hesla ikony na liště |
| `/api/v1/backups` | URL vzdáleného úložiště s přihlaš. údaji |
| `/api/v1/backup/1`[1] | URL vzdáleného úložiště s přihlaš. údaji,<br>šifrovací klíč |
| `/api/v1/backup/1/export?cmdline=true`<br>`&export-passwords=true` | URL vzdáleného úložiště s přihlaš. údaji,<br>šifrovací klíč |
| `/api/v1/backup/1/export?argsonly=true`<br>`&export-passwords=true` | URL vzdáleného úložiště s přihlaš. údaji,<br>šifrovací klíč |

[1] Hodnota `1` je identifikátor instance zálohy, citlivé informace se vztahují k této instanci zálohy.

# Bibliografie


1. DTI/PRICE WATERHOUSE COOPERS. *Information Security Breaches Survey 2004*. 2004. Tech. zpr. Převzato z článku; CAMPBELL, Mark. *What Are the Consequences of Data Loss?*. 2023. [cit. 2023-04-11]. Dostupné z: https://www.unitrends.com/blog/what-are-the-consequences-of-data-loss.
2. BAMBUCH, Michal. *Bezpečnostní analýza Drive Snapshot*. Praha, 2021. Dipl. pr. České vysoké učení technické, Fakulta informačních technologií.
3. G., Denis. [online]. 2020-03-26. [cit. 2023-04-23]. Dostupné z: https://www.msp360.com/resources/blog/backup-vs-archive/.
4. LIU, Jean. *3-Minute Guide on Backup vs. Sync* [online]. Přel. VEČERNÍK, Radek. 2023-02-22. [cit. 2023-04-23]. Dostupné z: https://www.easeus.com/backup-recovery/backup-vs-sync.html.
5. WALLEN, Dave. *Types of Backup: Full, Differential, and Incremental Backup* [online]. Spanning Cloud Apps, LLC, 2020-03-31 [cit. 2023-04-11]. Dostupné z: https://spanning.com/blog/types-of-backup-understanding-full-differential-incremental-backup/.
6. SIRON, Eric. *The Backup Bible*. Altaro Software, 2020. Dostupné také z: https://www.altaro.com/ebook/backup-bible.php.
7. TIP#757: Co je to verzování a jak vám v tom pomohou cloudová úložiště? *365tipu* [online]. 2017 [cit. 2023-04-12]. Dostupné z: https://365tipu.cz/2017/04/04/tip757-co-je-to-verzovani-a-jak-vam-v-tom-pomohou-cloudova-uloziste/.
8. *How IT Works: Data Backup Rotation Schemes* [online]. Mindsight, 2017-02-07 [cit. 2023-04-12]. Dostupné z: https://gomindsight.com/insights/blog/data-backup-schemes/.
9. SMITH, David M.; WILLIAMS, Michael L. *Data Loss and Hard Drive Failure: Understanding the Causes and Costs* [online]. ACE Data Recovery Engineering Inc., 2013 [cit. 2023-04-12]. Dostupné z: https://www.deepspar.com/wp-data-loss.html.
10. FASTNEURON INC. *Why you Shouldn't Use Optical Media for Backups* [online]. 2023. [cit. 2023-04-12]. Dostupné z: https://backupchain.com/why-you-shouldnt-use-optical-media-for-Backup.html.
11. SEAGATE TECHNOLOGY LLC. *What is NAS (Network Attached Storage) and Why is NAS Important for Small Businesses?* [online]. 2023. [cit. 2023-04-12]. Dostupné z: https://www.seagate.com/blog/what-is-nas-master-ti/.
12. MICROSOFT. *Co je cloudové úložiště?* [online]. 2023. [cit. 2023-04-12]. Dostupné z: https://azure.microsoft.com/cs-cz/resources/cloud-computing-dictionary/what-is-cloud-storage.

13. G2.COM, INC. *Best Backup Software* [online]. 2023. [cit. 2023-04-12]. Dostupné z: https://www.g2.com/categories/backup.
14. CEDRIC. *Top 10 Best Free Open Source Backup Software | 2023 Selections* [online]. EaseUS, 2023-02-22 [cit. 2023-04-12]. Dostupné z: https://www.easeus.com/backup-recovery/open-source-backup-software.html.
15. THE DUPLICATI TEAM. *Duplicati 2.0* [online]. 2023. [cit. 2023-04-12]. Dostupné z: https://www.duplicati.com.
16. *Duplicati* [online]. 2021-06-17. 2.0.6.3-beta [cit. 2023-04-12]. Zdroj. kód. GitHub. Dostupné z: https://github.com/duplicati/duplicati/tree/v2.0.6.3-2.0.6.3_beta_2021-06-17.
17. THE DUPLICATI TEAM. *Duplicati User's Manual* [online]. 2023-02-16. [cit. 2023-04-12]. Dostupné z: https://duplicati.readthedocs.io/en/latest/.
18. *Developer documentation* [online]. 2019-12-25. [cit. 2023-04-16]. Dostupné z: https://github.com/duplicati/duplicati/wiki/Developer-documentation.
19. THE DUPLICATI TEAM. *Different ways to make a Duplicati backup* [online]. 2016-12-26. [cit. 2023-04-21]. Dostupné z: https://www.duplicati.com/articles/Way-To-Make-A-Backup/.
20. SKOVHEDE, Kenneth. *A block-based storage model for remote online backups in a trust-no-one environment* [online]. 2014-02. [cit. 2023-04-17]. Dostupné z: https://www.duplicati.com/assets/Block-basedstorageformat.pdf.
21. IBM. *What is Block Storage?* [online]. 2023. [cit. 2023-04-16]. Dostupné z: https://www.ibm.com/topics/block-storage.
22. DWORKIN, Morris; BARKER, Elaine; NECHVATAL, James; FOTI, James; BASSHAM, Lawrence; ROBACK, E.; DRAY, James. *Advanced Encryption Standard (AES)*. Federal Inf. Process. Stds. (NIST FIPS), National Institute of Standards a Technology, Gaithersburg, MD, 2001. Dostupné z DOI: https://doi.org/10.6028/NIST.FIPS.197.
23. PACKETIZER, INC. *AES Crypt Wish List* [online]. 2023. [cit. 2023-04-19]. Dostupné z: https://www.aescrypt.com/wishlist.html.
24. ZIMMERMANN, Philip R. *Why I Wrote PGP* [online]. Boulder, Colorado, 1999 [cit. 2023-04-12]. Dostupné z: https://www.philzimmermann.com/EN/essays/WhyIWrotePGP.html.
25. THE GNUPG PROJECT. *The GNU Privacy Guard* [online]. 2023-04-28. [cit. 2023-05-03]. Dostupné z: https://gnupg.org/index.html.
26. FINNEY, Hal; DONNERHACKE, Lutz; CALLAS, Jon; THAYER, Rodney L.; SHAW, David. *OpenPGP Message Format* [RFC 4880]. RFC Editor, 2007. Request for Comments, č. 4880. Dostupné z DOI: 10.17487/RFC4880.
27. MNAIMAN, Max; THE DUPLICATI TEAM. *Terrible memory leak in 7Z compression #1849* [online]. 2021-05-08. [cit. 2023-05-02]. Dostupné z: https://github.com/duplicati/duplicati/issues/1849.
28. WELLONS, Christopher. *Unauthenticated header data is trusted, making the plaintext malleable (security issue)* [online]. 2019-03-31. [cit. 2023-04-19]. Dostupné z: https://github.com/paulej/AESCrypt/issues/23.
29. PACKETIZER, INC. *AES File Format* [online]. 2023. [cit. 2023-04-19]. Dostupné z: https://www.aescrypt.com/aes_file_format.html.
30. BUCHHOLZ, Florian. *The structure of a PKZip file* [online]. 2006. [cit. 2023-04-19]. Dostupné z: https://users.cs.jmu.edu/buchhofp/forensics/formats/pkzip-printable.html.

31. PKWARE INC. *APPNOTE.TXT - .ZIP File Format Specification* [online]. 2022-11-01. Ver. 6.3.10 [cit. 2023-04-20]. Dostupné z: https://pkware.cachefly.net/webdocs/casestudies/APPNOTE.TXT.
32. ASHCRAFT, Alvin; SHARKEY, Kent; SATRAN, Michael. *Change Journals* [online]. Microsoft, 2021-01-07 [cit. 2023-04-21]. Dostupné z: https://learn.microsoft.com/en-us/windows/win32/fileio/change-journals?view=net-8.0.
33. ULRICH TELLE. *System.Data.SQLite: RC4* [online]. 2022. [cit. 2023-04-24]. Dostupné z: https://utelle.github.io/SQLite3MultipleCiphers/docs/ciphers/cipher_sds_rc4/.
34. POPOV, Andrei. *Prohibiting RC4 Cipher Suites* [RFC 7465]. RFC Editor, 2015. Request for Comments, č. 7465. Dostupné z DOI: 10.17487/RFC7465.
35. SURLYHACKER. *Which tool can open encrypted DB* [online]. 2022-05-01. [cit. 2023-04-24]. Dostupné z: https://forum.duplicati.com/t/which-tool-can-open-encrypted-db/2372/17.
36. MDN CONTRIBUTORS. *Math.random()* [online]. Mozilla Develloper Network, 2023-03-28 [cit. 2023-04-25]. Dostupné z: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Math/random.
37. GRASSI, Paul; FENTON, James; NEWTON, Elaine; PERLNER, Ray; REGENSCHEID, Andrew; BURR, William; RICHER, Justin; LEFKOVITZ, Naomi; DANKER, Jamie; CHONG, Yee-Yin; GREENE, Kristen; THEOFANOS, Mary. *Digital Identity Guidelines: Authentication and Lifecycle Management [includes updates as of 03-02- 2020]*. Special Publication (NIST SP), National Institute of Standards a Technology, Gaithersburg, MD, 2020. Dostupné z DOI: https://doi.org/10.6028/NIST.SP.800-63b.
38. MICROSOFT. *Guid.NewGuid Method* [online]. 2023. [cit. 2023-05-04]. Dostupné z: https://learn.microsoft.com/en-us/dotnet/api/system.guid.newguid?view=net-8.0.
39. ECMA INTERNATIONAL. *ECMAScript® 2015 Language Specification* [online]. Geneva, 2015 [cit. 2023-05-04]. Standard ECMA-262, č. 6. Dostupné z: https://262.ecma-international.org/6.0/#sec-math.random.
40. WHEELER, Dan; DROPBOX, INC. *zxcvbn* [online]. 2017-02-07. [cit. 2023-04-12]. Dostupné z: https://github.com/dropbox/zxcvbn.
41. WELLONS, Christopher. *AES Crypt security audit (1 serious issue found)* [online]. Přel. VEČERNÍK, Radek. 2019-03-31. [cit. 2023-05-02]. Dostupné z: https://www.reddit.com/r/privacytoolsIO/comments/b7riov/aes_crypt_security_audit_1_serious_issue_found/.
42. ROELOFS, Greg; GAILLY, Jean-loup; ADLER, Mark. *zlib Technical Details* [online]. 2022-01-08. [cit. 2023-05-05]. Dostupné z: http://www.zlib.net/zlib_tech.html.
43. FORUM OF INCIDENT RESPONSE AND SECURITY TEAMS. *Common Vulnerability Scoring System SIG* [online]. 2023. [cit. 2023-05-06]. Dostupné z: https://www.first.org/cvss/.
44. FORUM OF INCIDENT RESPONSE AND SECURITY TEAMS. *Common Vulnerability Scoring System SIG* [online]. 2023. [cit. 2023-05-06]. Dostupné z: https://www.first.org/cvss/v3.1/specification-document.

# Obsah přiloženého média

- `README.md` ..... stručný popis obsahu média
- `examples` ..... adresář s ukázkami
- `text` ..... text práce
  - `src` ..... zdrojová forma práce ve formátu LaTeX
  - `thesis.pdf` ..... text práce ve formátu PDF