Uživatelský manuál

1. Účel aplikace

Aplikace umožňuje bezpečně ukládat a spravovat hesla uložená na přípravku Arduino za pomocí textového rozhraní (CLI) napsané v jazyku Rust.

Arduino slouží pouze jako úložiště dat. Veškeré šifrování (AES-256), dešifrování a zobrazování hesel provádí hostitelský počítač nástrojem `vault-cli`.

2. Požadavky

Hardware

- **Arduino**
  - MKR Zero (vestavěný SD card slot)
  - Uno/Nano + externí SD card modul
- **SD karta**
  - minimálně 4GB, formát FAT32
- **USB kabel**
  - sériová komunikace Arduino PC

Software

- **Rust toolchain**
  - `rustc`, `cargo` (instalováno společně s nástrojem `rustup`)
- **Arduino IDE**
  - nebo Arduino CLI
- **Rust CLI aplikace**
  - `vault-cli v0.1.0`
- **Arduino firmware**
  - `main.ino`
- **Arduino knihovny**
  - `SD` pro práci s SD kartou

3. Ovládání – vault-cli

```
vault-cli v0.1.0
CLI application for an Arduino-based password vault

USAGE:
    vault-cli [OPTIONS] [COMMAND]

OPTIONS:
    -h, --help          Display this help message
    -i, --interactive   Start in interactive mode
    -v, --version       Display version information
```

```
COMMANDS:
    init                    Initialize an empty vault
    add     <srv> <usr> <pwd>  Add a new entry
    get     [srv] [usr]        Retrieve entries
    delete  <srv> <usr>        Delete an entry
    help                    Show this help information
    exit                    Exit interactive mode
```

Základní použití

1. Připojte Arduino s vloženou SD kartou k počítači přes USB
2. Spusťte `vault-cli`
    - Jednotlivé příkazy

    ```
    vault-cli init
    vault-cli add github alice pw123
    ```

    - Interaktiní mód (REPL)

    ```
    vault-cli --interactive
    > add github alice pw123
    ```

3. Inicializujte nové úložiště příkazem `init` a zvolte si hlavní heslo
4. Přidejte první záznam příkazem `add`

```
vault-cli add <service> <username> <password>
```

5. Odeberte záznam pomocí `get`

```
vault-cli get [service] [username]
```

1. Interaktiní režim ukončete příkazem `exit`

Ukázka interaktiního režimu

```
 Welcome to vault-cli v0.1.0
 Type 'help' for commands or 'exit' to quit

> init
Create master password: [hidden]
Vault initialized successfully
> add google.com lukas 123
Entry added successfully
> get google.com lukas
Found 1 entry

Entry #1
Service: google.com
```

```
Username: lukas
Password: 123

> delete google.com lukas
Entry deleted successfully
> get google.com lukas
No entries found
> exit
Goodbye!
```

4. Chybové hlášky

Chyba	Příčina	Řešení
"Vault not initialized"	Nebyl spuštěn příkaz `init`	Spusťte `vault-cli init`
"Entry already exists"	Záznam pro daný service a username existuje	Smažte starý záznam nebo použijte jiné klíče
SD error	SD karta není vložena nebo špatný formát	Vložte FAT32 kartu, ověřte modul a wiring

5. Poznámky

Silné hlavní heslo:

- Zvolte dlouhé (min. 12 znaků), náhodné heslo
- Ideálně nepoužívejte osobní informace ani jednoduchá slova

Bezpečné odpojení:

- Před fyzickým odpojením Arduino od počítače vždy ukončete CLI (exit), aby se všechna data zapsala
- Vyvarujte se odpojení při probíhajícím zápisu na SD kartu

Hardware dokumentace

Použité komponenty

- Arduino MKR Zero (včetně integrované microSD)
- SD karta

Zapojení

- USB kabel mezi MKR Zero a PC

Zpráva

Cílem projektu bylo vytvořit bezpečný trezor na hesla, který je možné kdykoliv jednoduše připojit k počítači a přečíst uložené záznamy, přičemž bez znalosti hlavního hesla nelze trezor dešifrovat.

Původně jsem zamýšlel hlavní šifrovací klíč derivovat pomocí Argon2 a uložit ho na kryptografický čip, ze kterého by šlo po zamčení pouze číst. Nakonec jsem ale zvolil jednodušší bezpečnostní model: klíč derivuji a držím v paměti jen po dobu běhu programu. Při ukončení programu je klíč přepsán (anulován), aby jej nebylo možné zpětně získat. Tento model si zároveň vystačí pouze s Arduinem a není potřeba připojovat externí čip.

Pro spolehlivé mazání klíče i citlivých dat používá program speciální paměťovou strukturu, která po opuštění rozsahu („out of scope“) přepíše příslušné místo v RAM nulami. Totéž platí pro načtený soubor s hesly, který je v paměti jen dočasně pro rychlé čtení a zápis. Abych minimalizoval riziko čtení paměti za běhu, plánuji v budoucnu přidat funkci mlock, která zamezí přístup do datové oblasti jiným procesům.

Dalším zásadním problémem byla struktura uložených dat. Existence jediného velké souboru na SD kartě znamená delší přenos přes sériovou linku. Na druhou stranu mnoho samostatných souborů komplikuje přiřazování šifrovaných bloků k jednotlivým záznamům. Nakonec jsem se rozhodl pro kompromis: jeden soubor, který se při startu přenese do počítače a po skončení běhu zpět na Arduino.

Komunikace po sériové lince byla také problematická. Výtstupy kryptografických funkcí jsou “syrová data”, a proto můj program využívá pro jejich přenos vlastní formát. Před každým datovým blokem je posílán header s typem bloku a délkou v bajtech. Tím odpadá potřeba dalšího kódování a obě strany vždy vědí, kolik dat mohou očekávat.

Celkově projekt hodnotím jako úspěšný. Rust se mi osvědčil, binárka je velmi rychlá, což je pro mě jednou z klíčových metrik. Pro větší objem hesel ale plánuju paralelizovat některé procesy (například během zadávání hlavního hesla spouštět na pozadí čtení a přípravu dešifrace).

Dále chci upravit formát uložených dat tak, aby uživatel mohl požádat o jediné heslo, aniž by se musely přenášet a dešifrovat kilobajty ostatních záznamů. Tento přístup navíc vyřeší současné riziko ztráty dat při neočekávaném ukončení (Ctrl-C). Každý záznam by se pak ukládal samostatně, takže při přerušení hrozí ztráta nejvýše jednoho hesla, nikoli celého souboru.

Poslední vylepšení, které bych chtěl implementovat se týká zálohování. Šifrované soubory by se zároveň ukládaly i na počítači. V případě ztráty nebo poškození SD karty tak bude možné obnovit nejnovější verzi trezoru a zamezit ztrátě uložených údajů.