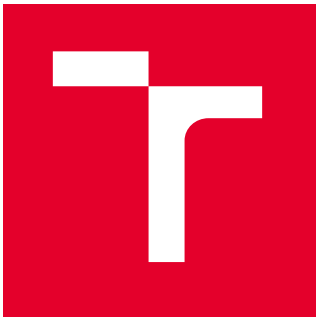


VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

ÚSTAV TELEKOMUNIKACÍ

DEPARTMENT OF TELECOMMUNICATIONS

ŠIFROVÁNÍ DAT NA USB DISKU

DATA ENCRYPTION ON USB FLASH DRIVE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **Petr Bráblík**
AUTHOR

VEDOUCÍ PRÁCE — **prof. Ing. Dan Komosný, Ph.D.**
SUPERVISOR

BRNO 2024

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

Bakalářská práce

bakalářský studijní program **Informační bezpečnost**

Ústav telekomunikací

Student: Petr Bráblík ***ID:*** 240592

Ročník: 3 ***Akademický rok:*** 2023/24

NÁZEV TÉMATU:

Šifrování dat na USB disku

POKYNY PRO VYPRACOVÁNÍ:

Vytvořte systém pro automatické šifrování dat na zvoleném oddílu USB disku. Systém bude data zpřístupňovat na základě identifikace počítače, do kterého se USB disk připojí. Vytvořte aplikaci v jazyce Python pro editaci seznamu povolených počítačů a zobrazení historie připojení USB disku k počítačům. Aplikace bude součástí operačního systému pro seniory. Výsledky práce publikujte na repozitáři GitHub pod licencí MIT.

DOPORUČENÁ LITERATURA:

Podle pokynů vedoucího práce

Termín zadání: 5.2.2024 ***Termín odevzdání:*** 28.5.2024

Vedoucí práce: prof. Ing. Dan Komosný, Ph.D.

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

ABSTRAKT

Bakalářská práce se zabývá návrhem a implementací bezpečnostního programu pro šifrování a automatické dešifrování dat na USB paměťovém disku. Systém je navržený tak, aby ulehčil používání operačního systému na USB paměťovém disku pro seniory ve věkové skupině 90 let a více. Cílem je nahrát „živou“ verzi operačního systému, která se může přímo spouštět z USB paměťového disku. Implementace skupiny skriptů zajistí automatické dešifrování dat podle MAC adresy počítače, ve kterém se USB paměťový disk nachází. Dále práce zahrnuje grafickou aplikaci pro úpravu povolených MAC adres na automatické dešifrování. Aplikace umožňuje přidat povolené MAC adresy, zobrazit historii počítačů, ve kterých se USB paměťový disk nacházel a zálohovat uživatelská data na další disk s operačním systémem pro seniory. Výsledky práce jsou publikovány na platformě GitHub.

KLÍČOVÁ SLOVA

operační systém, senior, USB disk, šifrování dat, UUID, Python, MAC adresa, Argon2, AES, Ubuntu

ABSTRACT

The bachelor thesis deals with the design and implementation of a security program for encryption and automatic decryption of data on a USB memory stick. The system is designed to facilitate the use of the operating system on a USB memory stick for seniors in the age group of 90 years and above. The goal is to load a „live“ version of the operating system that can be directly run from the USB memory stick. The implementation of a group of scripts will ensure automatic decryption of data according to the MAC address of the computer in which the USB memory stick is located. Furthermore, the work includes a graphical application for modifying the allowed MAC addresses for automatic decryption. The application allows to add enabled MAC addresses, view the history of computers in which the USB memory drive was located, and back up user data to another drive running the legacy operating system. The results of the work are published on GitHub.

KEYWORDS

operating system, senior, USB drive, data encryption, UUID, Python, MAC adress, Argon2, AES, Ubuntu

BRÁBLÍK, Petr. *Šifrování dat na USB disku*. Bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2024. Vedoucí práce: prof. Ing. Dan Komosný, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Petr Bráblík
VUT ID autora:	240592
Typ práce:	Bakalářská práce
Akademický rok:	2023/24
Téma závěrečné práce:	Šifrování dat na USB disku

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

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

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Mé poděkování si zaslouží vedoucí semestrální práce pan prof. Ing. Dan Komosný Ph.D. za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Obsah

Seznam výpisů a obrázků

Úvod

Práce s počítačem může být pro seniora velmi náročná, a proto vznikl projekt s názvem ForSenior-OS, který má seniorovi práci zjednodušit. ForSenior-OS se bude nacházet na USB paměťovém disku, čímž vznikne potřeba zabezpečit data proti krádeži nebo ztracení. Proto také byl vytvořen zabezpečovací program, který umožní asistentovi seniora vybrat povolené počítače podle MAC adresy a bude automaticky dešifrovat data na USB paměťovém disku. Díky tomuto si nebude muset senior pamatovat heslo a zamezí se kompromitování dat na disku. Systém je tak přizpůsoben pro užití staršími lidmi.

V první kapitole je obecně popsán operační systém GNU/Linux a jeho distribuce, datový formát ISO a principy „živého“ operačního systému. Druhá kapitola se zaměřuje na identifikátory počítače – MAC adresu nebo identifikátor UUID a jejich roli v implementaci. Třetí kapitola popisuje zabezpečovací prvky jako je kryptografický formát diskových oddílů LUKS, algoritmus na vytváření otisků Argon2 a algoritmus symetrické kryptografie AES. Ve čtvrté kapitole je rozebrán startovací proces počítače a jednotkové soubory potřebné pro spuštění služeb.

Praktická část začíná šestou kapitolou. Detailně popisuje návrh a obecnou představu programu Ssec a jeho funkcí. Sedmá kapitola uvádí implementaci programu, vysvětluje manuální a automatické dešifrování, úpravu konfigurace a práci s konfiguračním souborem.

Kapitola je ukončena analýzou bezpečnosti programu Ssec. Je zmíněn obecný popis útoků, se kterými se program může setkat. Je popsána účinnost implementovaných bezpečnostních prvků a designových rozhodnutí v kontrastu s naivní implementací některých funkcí.

1 Operační systémy

V této kapitole je popsán operační systém GNU/Linux a dvě z jeho mnoha distribucí: Fedora a Ubuntu. Distribuce jsou operační systémy zakládající se na Linuxu, ale lišící se softwarovým vybavením. Dále je popsáno, co znamená „živý“ systém a jsou popsány vlastnosti souboru s příponou `.iso`, jako je například datová perzistence.

1.1 Operační systém GNU/Linux

GNU/Linux je systém s otevřeným kódem, který se skládá se ze dvou částí: jádra operačního systému Linux a operačního systému GNU a jeho nástrojů. Linux je jádro systému, který přiděluje zdroje ostatním programům, které se spouští. Jádro je nezbytnou součástí operačního systému, ale samo o sobě je nepoužitelné; může fungovat pouze v kontextu kompletního operačního systému. Operační systém GNU slouží jako komunikační prostředek mezi hardwarem a softwarem. Linux se obvykle používá v kombinaci s operačním systémem GNU: celý systém je v tedy Linux s přidaným GNU neboli GNU/Linux. [1] [2]

Zakladatel Linuxu se jmenuje Linus Torvalds. On také vymyslel zkratku GNU, což je rekurzivní zkratka (GNU is Not Unix) a název projektu podporujícího vývoj svobodného softwaru. GNU/Linux je tedy správné označení pro svobodný operační systém založený na linuxovém jádru a tvořený svobodnými programy s otevřeným zdrojovým kódem. Běžně se lze setkat s jeho označením jenom jako Linux. [3]

Velkou výhoda Linuxu je možnost vývoje a přizpůsobení prostředí. Existují Linuxové distribuce, což jsou varianty operačního systému Linux, které kombinují jádro Linuxu s různými softwarovými balíčky a konfiguracemi. Distribuce se mohou lišit například uživatelským prostředím. V této práci je například použitá „živá“ verze linuxového operačního systému, vhodná pro použití na USB paměťovém disku. Podrobněji bude „živý“ systém popsán dále v této kapitole. [3]

Distribuce Fedora

Fedora je kompletní operační systém (tzv. distribuce Linuxu), která vznikla jako nekomerční odnož Red Hat Linuxu (RHEL). Vyvíjí ho komunita vývojářů za podpory firmy Red Hat. Nejznámější a nejpoužívanější verze Fedory verze se nazývá „Workstation“, která je zaměřená na použití na osobních počítačích. Tuto distribuci není doporučeno používat zvláště kvůli zastaralému nástroji LiveCD Tools popsanému v sekci Použité nástroje. [4]

Distribuce Ubuntu

Ubuntu je multifunkční distribuce GNU/Linux. Jeho design umožňuje provoz na široké škále zařízení a v různých aplikacích. Dále design nabízí jednotný zážitek bez ohledu na použití. V práci je použita právě tato distribuce, protože je na ní možné splnit všechny podmínky vypsané v kapitole Příprava prostředí. [5]

1.2 Živý operační systém

Díky tomu, že všechny obrazy operačních systémů Fedora a Ubuntu splňují požadavky standardu ISO-9660, je možné je zapsat na USB paměťový disk.

Je tedy možné vytvořit „živou“ neboli „live“ verzi operačního systému, která vznikne tím, že se soubor s příponou `.iso` operačního systému (v případě této práce se jedná o operační systém Ubuntu) zapíše na USB paměťový disk a vznikne tím startovací disk. Ze startovacího disku je možné spustit operační systém, aniž by se použil pevný disk počítače, ve kterém se USB paměťový disk nachází. [6]

Soubory s příponou .iso

Soubor s příponou `.iso` (dále ISO obraz) je nekomprimovaný obraz diskového archivu, který reprezentuje obsah celého datového nosiče optického disku, jako je například CD nebo DVD. Na základě standardu ISO-9660 formát souboru obrazu ISO obsahuje data disku spolu s informacemi o souborovém systému. [7]

Schopnost ISO souborů obsahovat přesnou binární kopii obsahu je činí ideálním typem souboru k ukládání spustitelných dat. Většinou jsou ISO soubory vypalovány na USB/CD/DVD pro spuštění programu pro instalaci operačního systému. V případě této práce se obraz ISO zkopíruje na první diskový oddíl USB paměťovém disku a bude sloužit jako plnohodnotný operační systém. [7]

Perzistence dat

ISO obrazy mají vlastnost, že se po restartování operačního systému vrátí do původního stavu. Data dříve zapsaná na USB paměťový disk na něm nezůstanou, jsou tedy neperzistentní. Perzistence obecně znamená „pokračování účinku poté, co je odstraněna jeho příčina“. V kontextu uložení dat v počítačovém systému znamená, že data zůstanou nezměněná i po ukončení procesu, kterým byla vytvořena. Tato vlastnost zajišťuje, že i při vypnutí nebo restartování systému jsou data stále dostupná a uchována na daném médiu. V této práci je zajištěna vlastnost perzistence dat na USB paměťovém disku, který je použit pro zavedení programu pro rozpoznání identifikátoru počítače. [8]

2 Identifikace povoleného počítače

V této kapitole jsou popsány dvě možnosti způsobů, jak identifikovat počítač. Jsou popsány vlastnosti MAC adresy a UUID a dále porovnána praktičnost jejich použití.

2.1 MAC adresa

Adresa MAC (Media Access Control) představuje jeden z klíčových prvků v síti. MAC adresy umožňují komunikaci mezi hardwarovými zařízeními. Pracují na druhé vrstvě ISO/OSI internetového modelu. MAC adresy jsou pevně spojeny s konkrétními zařízeními, přičemž jsou přiřazeny výrobcem. Můžete se setkat s více možnými názvy MAC adres:

- adresa síťového hardwaru,
- adresa vypálení (BIA),
- fyzická adresa,
- ethernetová hardwarová adresa (EHA). [9]

Připojení přes Wi-Fi, Bluetooth a Ethernet využívá MAC adresy. MAC adresy jsou unikátní identifikátory a jsou přiřazeny každé síťové kartě. Síťové karty se pak nachází v každém zařízení v síti, což umožňuje komunikaci a směrování dat přímo k daným zařízením. Díky tomuto principu je pak možné se připojit k internetu. Síťová karta je také známá jako Network Interface Controller (NIC). [9]

MAC adresy mají formát 12místného hexadecimálního čísla. Číslice jsou pak po dvojicích odděleny dvojtečkou nebo pomlčkou. Příklady MAC adres ve správném formátu jsou tedy `2C:54:91:88:C9:E3` nebo `2c-54-91-88-c9-e3`. [9]

Někteří známí výrobci síťových adaptérů nebo síťových karet, jako například Dell, Belkin, Nortel a Cisco umísťují do MAC adresy speciální číselnou sekvenci nazývanou OUI (Organization Unique Identifier), která je identifikuje jako konkrétního výrobce. OUI většinou bývají první tři oktety na začátku adresy. [10]

2.2 Identifikátor UUID

Identifikátor UUID (Universally Unique Identifier), někdy označované jako „GUID“ (Globally Unique Identifier), jsou 128bitové hodnoty, které jsou reprezentovány jako 36znakový řetězec ve formátu `123e4567-e89b-12d3-a456-426614174000` (pět hexadecimálních řetězců oddělených pomlčkami). Existuje několik verzí UUID odlišnými vlastnostmi. Standardy pro UUID jsou formálně definovány v RFC 4122, zveřejněném v roce 2005. [11]

UUDI verze 1 je založena na aktuálním čase a MAC adrese počítače („uzlu“). Prvních 24 číslic je generováno pomocí počtu nanosekund, které uběhly od 15. října

1582 o půlnoci UTC. Posledních 12 hexadecimálních číslic řetězce UUID reprezentuje MAC adresu uzlu. V některých implementacích se namísto skutečné MAC adresy uzlu používá náhodná MAC adresa. [13]

Verze UUID 4 je plně náhodná, kromě jedné číslice 4 (v pořadí 13. číslice), která značí verzi. Existuje přibližně $5,3 \cdot 10^{36}$ možných UUID. Toto číslo znamená, že i při generování 1 miliardy UUID za sekundu po dobu 85 let je šance na vytvoření duplikátu pouze 50 %. [12]

Díky extrémně nízké pravděpodobnosti duplicity lze považovat každé UUID za jedinečné, aniž by bylo nutné koordinovat nebo registrovat prostřednictvím centrálního orgánu. Rozdílné počítačové systémy mohou generovat UUID současně a nezávisle a přesto zaručují jedinečnost. UUID lze generovat lokálně. Většina programovacích jazyků poskytuje mechanismy pro generování UUID, což usnadňuje kompatibilitu mezi různými systémy. [11] [12]

Pro jednodušší a pohodlnější používání pro asistenta seniora je v této práci použita MAC adresa jako identifikátor počítače, avšak UUID je použito jako kryptografický klíč pro zašifrování uživatelského hesla. Podle organizace OWASP může být identifikátor UUID dostatečně náhodný jako kryptografický klíč, záleží ale na verzi a konkrétní implementaci. [14]

3 Zabezpečení

V této kapitole jsou obecně popsány zabezpečovací prvky systému. Nejdříve je popsán formát LUKS, pomocí kterého je zašifrovaný diskový oddíl s uživatelskými daty. Dále jsou popsány obecně otisky (haše) a symetrická kryptografie. V práci jsou použity dva algoritmy: algoritmus Argon2 a algoritmus AES. Argon2 se používá na vytváření otisků a v práci je použit pro vytvoření otisků z MAC adres. Dále je zmíněn algoritmus AES, který je použitý pro zašifrování uživatelského hesla.

3.1 Formát LUKS

Formát LUKS (Linux Unified Key Setup) je standard pro šifrování disků v operačním systému Linux. Poskytuje způsob šifrování dat nezávislý na disku a jednotný mechanismus pro šifrování diskových oddílů. LUKS byl vytvořen tak, aby byl kompatibilní s různými distribucemi Linuxu.

Formát LUKS umožňuje vytvářet šifrované oddíly na pevných discích nebo jiných úložištích, na kterých jsou uložená citlivá data. Formát LUKS umožňuje správu šifrování pomocí hesel nebo klíčů a poskytuje ochranu před neoprávněným přístupem k datům na disku, což je užitečné v případě ztráty nebo odcizení zařízení. Po vytvoření šifrovaného oddílu může uživatel přistupovat k datům na disku pouze po úspěšném ověření hesla nebo předložení klíče. [15]

Clemens Fruhwirth v roce 2004 vytvořil specifikaci LUKS pro šifrování disků. Specifikace je převážně určena pro Linux. Jedná se o dobře známou, bezpečnou a vysoce výkonnou metodu šifrování disků založenou na vylepšené verzi nástroje `cryptsetup`, který využívá `dm-crypt` v pozadí pro šifrování disků. [15]

Formát LUKS umisťuje metadata před zašifrovaná data a na diskovém oddílu vytváří záhlaví (header). Metadata ukládají informace o šifrovacím algoritmu, délce klíče, metodě řetězení bloků atd. Parametry pak není nutné držet v paměti, což formát LUKS činí vhodným pro použití například na USB paměťových discích. Formát LUKS používá hlavní klíč, který je zašifrován pomocí otisku (haše) hesla. Tím je možné změnit heslo a lze použít více hesel. Po úspěšném ověření hesla nebo je klíč načten a použit pro dešifrování dat na disku. Nástroj `cryptsetup` pak umí pracovat s takto zašifrovanými bloky dat. Formát LUKS je výchozím a doporučeným formátem pro šifrované diskové oddíly. [15] [16]

3.2 Funkce na vytvoření otisku

Obecně hašovací funkce, neboli funkce na otisk, převádějí text nebo jiná data na řetězce stejné délky. Výstup hašovací funkce se nazývá haš nebo otisk. Různé vstupy

jsou obvykle převáděny na různé výstupy, ale někdy může dojít ke kolizi. Kolize nastává tehdy, kdy dva různé vstupy do funkce mají stejný otisk. [17] [18]

Kryptografická funkce na otisky je matematická operace, která převede libovolně dlouhý vstupní řetězec dat na řetězec pevné délky. Kryptografické funkce na otisk jsou známy tím, že jsou odolné proti kolizím a nevratné. Důležitou vlastností funkcí je, že drobná změna ve vstupních datech vytvoří velmi odlišný otisk. Další důležitou vlastností je velmi obtížné získat původní data z otisku. [18]

Funkce na otisk se často používá pro ověření integrity dat, například když je potřeba zkontrolovat, zda se soubor během přenosu nepoškodil nebo nebyl změněn. Dále se používají pro vytváření otisků hesel v databázích, aby nebylo možné je přímo číst. V této práci jsou použity pro vytvoření otisků MAC adres z důvodu, aby nebyly uloženy jako prostý text.

3.2.1 Implementace otisků v programování

„V programování jsou funkce na otisk používány při implementaci datové struktury hašovací tabulky (asociační pole), která mapuje hodnoty určitého vstupního typu na hodnoty jiného typu, například mapování názvu produktu (text) na cenu produktu (desetinné číslo).

Naivní funkce na otisk může být například součet bajtů vstupních dat. To ale způsobuje mnoho kolizí, například hello a ehllo budou mít stejný otisk.

V kryptografii transformují funkce na otisk vstupní data libovolné velikosti (například textovou zprávu) na výsledek pevné velikosti (například 256 bitů), který se nazývá otisk (hašový kód, otisk zprávy nebo jednoduše haš). Příklady takových funkcí jsou SHA-256 a SHA3-256, které přeměňují libovolný vstup na 256bitový výstup.“ [18]

3.2.2 Algoritmus Argon2

Argon2 je moderní algoritmus pro odvozování klíčů a vytváření otisků, který je navržen speciálně pro ukládání hesel. Je možné si zvolit, jak dlouho trvá vytvoření otisku hesla a kolik paměti je vyžadováno. V září 2021 byl Argon2 standardizován organizací IETF v RFC 9106. [19]

Návrh algoritmu Argon2 vychází z toho, že hesla jsou typicky slabým článkem v zabezpečení systémů. Argon2 je navržen tak, aby byl odolný vůči různým typům útoků, například útoku hrubou silou (brute-force) nebo slovníkových útoků.

Argon2 existuje ve třech variantách: Argon2d, Argon2i a Argon2id. Síla varianty Argon2d spočívá v odolnosti proti kompromisu čas-paměť, zatímco varianta Argon2i se zaměřuje na odolnost proti útokům na postranní kanály. Podle toho bylo původně považováno Argon2i za správnou volbu pro vytváření otisků hesel a odvozování klíčů

z hesla. V praxi se však ukázalo, že kombinace d a i – která kombinuje jejich silné vlastnosti – je lepší volbou. A tak vznikl Argon2id, který je nyní považován za hlavní a jedinou variantu, kterou je podle RFC 9106 povinné implementovat pro splnění standardu. [20]

3.3 Symetrická kryptografie

Symetrická kryptografie je typ šifrování u kterého je použit stejný klíč pro šifrování a dešifrování dat. To znamená, že odesílatel a příjemce musí mít přístup ke stejnému tajnému klíči, vyžaduje tedy bezpečný způsob distribuce klíčů mezi komunikujícími stranami. Algoritmus AES (Advanced Encryption Standard) je jedním z nejpoužívanějších symetrických šifrovacích algoritmů. Symetrické šifrovací algoritmy mohou být rozděleny do dvou typů:[21]

- Blokové algoritmy: Blokové algoritmy rozdělí data do bloků a každý blok je pak zašifrován pomocí klíče. Klíč může být pro každý blok stejný, ale není to doporučené, protože je pak možné vyčíst data i ze šifrového textu. Každý blok má pevnou velikost a systém udržuje data o každém bloku v paměti, dokud není každý blok zašifrován.
- Proudové algoritmy: Na rozdíl od blokových algoritmů se u proudových algoritmů každý znak zašifruje tehdy, kdy vstoupí do funkce. To je považováno za bezpečnější ve srovnání s blokovými algoritmy, protože systém neukládá data do paměti.[21]

3.3.1 Algoritmus AES

Algoritmus AES (Advanced Encryption Standard) je blokový symetrický šifrovací algoritmus, který byl vytvořen jako náhrada za algoritmus DES (Data Encryption Standard). AES má tři varianty, které se liší délkou klíče. AES je široce používán pro zabezpečení důvěrnosti dat. Je součástí protokolu SSL/TLS a je jím zašifrována většina internetové komunikace. Bloková šifra rozdělí zprávu do bloků, které potom permutuje sekvencí bitů nazývaných klíč. Blok je sekvence bitů pevné délky.

Blok je sekvence 128 bitů; data vstupu a výstupu pro blokové šifry AES jsou bloky. Klíč je sekvence bitů, která jako další vstupuje do bloků šifry AES. Klíč je obvykle nastaven předem a udržován během mnoha volání blokové šifry. [22]

Historie algoritmu AES

V roce 1997 NIST zahájil vývoj standardu pokročilého šifrování (AES) a vyzval veřejnost, aby předložila kandidátní algoritmy pro blokové šifry. Blokové šifry jsou základem mnoha kryptografických služeb, zejména těch, které zajišťují důvěrnost

dat. V roce 2000 NIST oznámil výběr Rijndael pro AES. Tento standard specifikuje tři instance Rijndaelu: AES-128, AES-192 a AES-256, kde přípona označuje délku klíče. Velikost bloku (tj. délka vstupů a výstupů dat) je v každém případě 128 bitů. Rijndael podporuje další velikosti bloků a délky klíčů, které nejsou v tomto standardu přijaty. [22]

Módy šifry AES

Režim blokové šifry, zkráceně režim, je algoritmus, který využívá symetrickou blokovou šifrovací metodu s cílem poskytnout informační službu jako je důvěrnost nebo autentizace. [23]

V současnosti NIST schválil čtrnáct režimů blokových šifer v řadě zvláštních publikací. Mezi aktuální režimy patří osm režimů důvěrnosti (ECB, CBC, OFB, CFB, CTR, XTS-AES, FF1 a FF3), jeden režim autentizace (CMAC) a pět kombinovaných režimů pro důvěrnost a autentizaci (CCM, GCM, KW, KWP a TKW). V práci použit mód CBC, autentizace totiž není pro toto použití nutná. [23]

4 Startovací proces počítače

V této kapitole bude popsán startovací proces počítače společně s jednotkovými soubory služby Systemd. Jednotkové soubory jsou použité pro spouštění skriptů při startovacím procesu počítače.

4.1 Program BIOS

Při startu počítače hledá procesor v systémové paměti program BIOS (Basic Input/Output System), který následně spustí. Program BIOS nejenom řídí první fázi procesu spouštění, ale také poskytuje nejnižší úroveň rozhraní pro periferní zařízení. Z tohoto důvodu je zapsán do trvalé paměti pouze pro čtení. Kromě prostředí BIOS je například možno se setkat s prostředím Extensible Firmware Interface (EFI).[24]

Po načtení programu BIOS provede testy systému, prozkoumá a ověří zařízení pomocí kterého lze načíst a spustit operační systém. Prověří všechny přítomné pevné disky. Pořadí prohledávaných jednotek se obvykle určuje při spouštění podle nastavení v systému BIOS. BIOS poté nahrává do paměti program umístěný v prvním sektoru tohoto zařízení, známý jako MBR (Master Boot Record). MBR a obsahuje strojový kód pro spuštění počítače spolu s tabulkou oddílů. Po načtení zaváděcího programu do paměti předá BIOS kontrolu nad procesem spouštění tomuto programu.[24] [25]

4.2 Zavaděč GRUB

Zavaděč GRUB (GRand Unified Bootloader) je zavaděč, jenž je rozdělen do dvou fází. První fáze obsahuje binární strojový kód na MBR, jehož jediným úkolem je nalézt zavaděč druhé fáze a načíst ho do paměti. GRUB umožňuje čtení souborových systémů a načítání konfiguračního souboru **`/boot/grub/grub.conf`** při spouštění.[24]

Po načtení zavaděče druhé fáze do paměti GRUB zobrazí uživateli grafickou obrazovku s různými operačními systémy, které BIOS našel při průzkumu zařízení. Uživatel má pak možnost vybrat, který operační systém chce načíst. [24]

Jakmile zavaděč druhé fáze určí, které jádro má být načteno, najde odpovídající binární soubor jádra v adresáři **`/boot/`**. Binární soubor jádra má název ve formátu **`/boot/vmlinuz-<verze-kernelu>`** (**`<verze-kernelu>`** odpovídá verzi jádra nastavené ve zavaděči).[24]

Častokrát je zavedena podpora udržování více verzí jádra, takže pokud dojde k problému s nejnovějším jádrem, lze zavést verzi starší. Ve výchozím nastavení

poskytuje GRUB nabídku nainstalovaných jader před spuštěním, včetně možnosti záchrany a, je-li nakonfigurována, možnosti obnovy.[25]

Poté, co nalezne jádro operačního systému, předá jádro do paměti RAM a přenechá řízení počítače jádru.

4.3 Jádro operačního systému

Po načtení jádra probíhá inicializace a konfigurace paměti počítače a nastavení různého hardware připojeného k systému, včetně všech procesorů, `I/O` (vstupních a výstupních) subsystémů a úložných zařízení.[24]

Jádro následně vytvoří kořenové zařízení, připojí kořenový oddíl pouze pro čtení a uvolní veškerou nepoužívanou paměť. V této fázi je jádro plně načteno do paměti a je funkční. Zatím ale nejsou spuštěné žádné uživatelské aplikace, které by umožňovaly vstup do systému. Pro nastavení uživatelského prostředí jádro spustí program **`/sbin/init`** nebo Systemd.[24]

4.4 Systemd

Systemd je správce systému a služeb pro operační systémy Linux a je zodpovědný za uvedení operačního systému do stavu, kdy ho může uživatel používat. Systemd je náhrada za starší program Init (nebo SystemV), díky čemuž umožňuje rozsáhlejší počet funkcí, například připojování souborových systémů a správu systémových služeb.[24] [25]

Systemd obvykle není vyvoláván přímo uživatelem; spíše je instalován jako symbolický odkaz **`/sbin/init`** a spouští se během časné fáze spouštění. Instance správce uživatelů jsou automaticky spuštěny prostřednictvím služby **`user@.service(5)`**. [25]

Při spouštění Systemd připojí souborové systémy definované v **`/etc/fstab`**, včetně všech dočasných souborů nebo oddílů. V tomto okamžiku může přistupovat ke konfiguračním souborům umístěným v **`/etc`**, včetně svých vlastních. V adresáři **`/etc/systemd/system/default.target`** se nachází **`default.target`**, což je pouze symbolický odkaz na skutečný cílový soubor. Tento soubor určuje úroveň běhu počítače. Například **`graphical.target`** je ekvivalentní úrovni běhu 5 ve starším SystemV init. [25]

Systemd je kontroverzní z několika důvodů: je to náhrada za něco, o čem si mnoho uživatelů Linuxu nemyslí, že je třeba nahradit a série špatných rozhodnutí vývojářů Systemd si nezískala pozitivní ohlas linuxové komunity. Linus Torvalds například zakázal developerovi Systemd Kay Sieversovi vývoj linuxového jádra.[27]

I přes svoji kontroverzi je Systemd hojně používaný. V této práci jsou použity jednotkové soubory programu Systemd.

4.4.1 Jednotkové soubory

Jednotkové soubory (Unit Files) obsahují konfigurační direktivy, popisují a definují chování dané služby. Hlavními adresáři, kde jsou jednotkové soubory uloženy, jsou **`/etc/systemd/system/`**, který je určený pro jednotkové soubory vytvořené správcem systému.[26]

Názvy souborů jednotek mají obvykle následující formát: **`<název_jednotky>.<typ_přípony>`**. Kde **`<název_jednotky>`** představuje název dané služby a **`<typ_přípony>`** říká, o jaký typ jednotky se jedná. Například v operačním systému můžete najít službu **`sshd.service`** nebo jednotku **`sshd.socket`**.[26]

Správce služeb Systemd může také vytvářet adresáře **`sshd.service.wants/`** a **`sshd.service.requires/`**. Tyto adresáře obsahují symbolické odkazy na jednotkové soubory, které jsou závislé na službě **`sshd`**. Symbolické odkazy jsou vytvářeny automaticky buď během instalace na základě možností jednotkového souboru **`[Install]`** nebo za běhu na základě voleb **`[Unit]`**. Tyto adresáře a symbolické odkazy je možné vytvořit i ručně.[26]

4.4.2 Struktura jednotkových souborů

Soubory jednotek se obvykle skládají ze tří následujících částí:

- Sekce **`[Unit]`** - zde se nachází obecné možnosti, které nejsou závislé na typu služby. Tyto možnosti poskytují popis služby, určují chování služby a nastavují závislosti na jiných službách.
- Sekce **`[Unit Type]`** - Obsahuje direktivy specifické pro daný typ. Například soubory servisních služeb (**`.service`**) mají tuto sekci pojmenovanou **`[Service]`**.
- Sekce **`[Install]`** - Obsahuje informace o instalaci služby používané nástroje **`systemctl`**, například příkazy povolení a zakázání. [26]

4.5 Použité jednotkové soubory

Každá sekce má své parametry, které budou vysvětleny na dvou jednotkových souborech použitých v práci: **ssec.service** a **ssec.desktop**.

Jednotkový soubor **ssec.service** zajišťuje (výpis 4.1), aby se skript **ssec.py** spouštěl při startovacím procesu počítače. Parametr **before** určuje chvíli, ve které se služba spustí. Hodnota parametru **before** je **multi-user.target**, což je chvíle startovacího procesu počítače, jakmile se načetly všechny služby potřebné pro běh operačního systému a operační systém je připravený zobrazit grafické uživatelské prostředí. Je to uděláno tímto způsobem vzhledem k tomu, že program Ssec není nutný pro běh operačního systému. Služba čeká se na načtení všech kritických služeb.

Důležitý parametr je také **type**, který určuje, zda služba bude spouštět i podprocesy. Byl zvolen typ **oneshot**, který dovoluje spouštění podprocesů, které jsou použité k provedení příkazů v příkazové řádce.

Parametr **ExecStart** ukazuje, který program se má spustit. Nejdříve odkazuje na adresář, kde je uložený Python, který se musí nejdříve spustit. Poté odkazuje na skript **ssec.py**, který se má spouštět při startovacím procesu počítače.

```
1  [Unit]
2  Description=Automated LUKS decryption
3  Before=multi-user.target
4
5  [Service]
6  Type=oneshot
7  ExecStart=/usr/bin/python3 /ssec/ssec.py
8
9  [Install]
10 WantedBy=multi-user.target
```

Výpis 4.1: Ukázka jednotkového souboru ssec.service

Soubory s příponou **.desktop** se také označují za jednotkové soubory, avšak mají rozdílný formát a rozdílné parametry. Ukázkou může být soubor **ssec.desktop**. Soubor **ssec.desktop** (výpis 4.2) zajišťuje spouštění skriptu **ssec_startup.py** při načtení plochy a grafického uživatelského prostředí. Nachází se zde parametr **Exec**, který je podobný parametru **ExecStart** ze služby ssec.service. Musí se zde také určit, který skript se má spustit. Jednotkový soubor nejdříve určí, že se má spustit Python a poté pomocí Pythonu skript **ssec_startup.py**.

Parametry **Type** a **Terminal** určují o jaký typ programu se jedná. Bylo zadáno, že typ je **application** a parametr **terminal** byl zadán jako **false**. Tímto bylo řečeno, že se jedná o grafickou aplikaci, která se nemá spouštět v terminálu.

```
[Desktop Entry]
Type=application
Name=Ssec login window
Exec=python /bin/ssec_startup.py
Terminal=false
Type=Application
Categories=

X-GNOME-Autostart-enabled=true
X-GNOME-Autostart-Delay=0
```

Výpis 4.2: Ukázka jednotkového souboru ssec.desktop

5 Použité nástroje

V této kapitole jsou popsány standardní a nestandardní knihovny jazyku Python. Celá práce je napsána v jazyce Python s pomocí těchto modulů. Dále jsou popsány vlastnosti dvou nástrojů pro vytvoření „živého“ operačního systému.

5.1 Programové moduly jazyku Python

Python je vysokoúrovňový programovací jazyk, který se vyznačuje jednoduchou syntaxí. Je interpretovaný, což znamená, že kód se vykonává řádek po řádku a není nutné ho předem kompilovat. Python má jednoduchý přístup k objektově orientovanému programování a jeho interpretovaná povaha umožňuje skriptování a rychlý vývoj aplikací v mnoha oblastech a platformách. Interpret Pythonu lze snadno rozšiřovat o nové funkce a datové typy implementované jinými programovacími jazyky. Tím, že je Python volně dostupný a snadno rozšiřitelný, je možné najít mnoho různých nástrojů a knihoven pro různé využití. [28]

5.1.1 Standardní knihovny

Standardní knihovny jsou knihovny, které jsou již zahrnuté v instalaci jazyka Python.

Knihovna subprocess

Knihovna subprocess v Pythonu poskytuje možnost spouštění externích procesů. Dále umožňuje s nimi manipulovat. Umožňuje spouštět příkazy systému, komunikovat s jejich vstupy a výstupy a získávat návratové kódy a další informace o probíhajících procesech. Tato knihovna je užitečná pro spouštění externích příkazů, spouštění skriptů a práci s výstupy a chybovými zprávami z těchto procesů. V práci je použita pro provádění příkazů při startovacím procesu počítače. [29]

Knihovna os

Knihovna os poskytuje přenosný způsob používání funkcí závislých na operačním systému. Je možné využít nástroje jako je například modul **`os.path`**, který umožňuje manipulaci s cestami. Dále je možné využít modul **`fileinput`** pro čtení souborů v příkazovém řádku. Pro práci se soubory a adresáři na vyšší úrovni funkčnosti je možné využít modul **`shutil`**. V práci je modul použit převážně pro práci se soubory, například vytváření nových adresářů. [30]

Knihovna re

Modul re v Pythonu poskytuje funkce pro práci s regulárními výrazy, což jsou vzory používané pro vyhledávání a manipulaci s textovými řetězci. Pomocí tohoto modulu je možné například vyhledávat vzory v textu, nahrazovat části textu jinými řetězci nebo rozdělovat text na základě určitého vzoru. V modulu můžeme najít operace pro hledání výrazů podobné těm jako jsou v jazyce Perl. V práci je tato knihovna využita pro kontrolu uživatelského vstupu a jako bezpečnostní prvek proti zadávání neoprávněných znaků. [31]

Knihovna base64

Modul base64 poskytuje funkce pro kódování binárních dat na tisknutelné ASCII znaky a dekódování těchto kódování zpět na binární data. Poskytuje funkce pro kódování a dekódování pro kódování specifikovaná v RFC 4648, která definuje algoritmy Base16, Base32 a Base64.[32]

Base64 je způsob reprezentace binárních dat pomocí znaků ASCII, což umožňuje bezpečné přenosy dat přes média, která mohou mít omezené podporované znaky. V práci je použita pro zakódování zašifrovaného uživatelského hesla pro uložení do konfiguračního souboru.[32]

Knihovna datetime

Modul datetime v Pythonu poskytuje třídy a funkce pro práci s časem. Tento modul umožňuje vytvářet, manipulovat a formátovat data a časové údaje. V práci je použit pro vytváření časových známek v historii počítačů, ve kterých USB paměťový disk byl. [33]

5.1.2 Grafická knihovna Tkinter

Balíček Tkinter je standardním rozhraním Pythonu k sadě nástrojů GUI Tcl/Tk. Tk i Tkinter jsou dostupné na většině unixových platforem, včetně macOS, a také na systémech Windows. Tkinter je rozsáhlý aplikační rámec, který přidává značné množství vlastní logiky, aby se programování co nejvíce přiblížilo Pythonu bez rozšiřujících knihoven. Díky tomuto modulu je možné v práci vytvořit grafická okna, například pro aplikaci na úpravu konfigurace. [34]

5.1.3 Knihovna argon2-cffi

Modul argon2-cffi je Pythonová implementace algoritmu Argon2, který je navržen pro bezpečné odvozování klíčů a vytváření otisků hesel. Tato knihovna poskytuje

Pythonové rozhraní pro použití Argon2, které je vhodné pro ukládání a ověřování hesel v aplikacích. V práci je použita pro vytváření otisků MAC adres povolených počítačů. Více bude vysvětleno v kapitole o implementaci. [19]

5.1.4 Knihovna PyCryptodomex

PyCryptodomex je samostatný balíček nízkoúrovňových kryptografických primitiv v jazyce Python. Modul poskytuje funkce pro kryptografické operace jako je šifrování dat a vytváření otisků. Obsahuje implementace různých kryptografických algoritmů, včetně AES (Advanced Encryption Standard), který je použit pro zašifrování uživatelského hesla. Jako klíč je použito UUID počítače. Více je popsáno v kapitole o implentaci. Knihovna PyCryptodomex je užitečná pro zabezpečení dat v aplikacích a pro implementaci různých kryptografických protokolů. [35]

5.2 Nástroje pro vytvoření „živého“ systému

5.2.1 UNetbootin

UNetbootin je nástroj, který umožňuje vytvářet živé operační systémy spustitelné z USB paměťového disku pro distribuce Linuxu bez potřeby vypalování CD obrazů. UNetbootin má funkci stažení z mnoha podporovaných distribucí. Je ale možné použít i vlastní ISO obraz Linuxu. [36]

5.2.2 LiveCD Tools

LiveCD Tools je program pro generování „živých“ obrazů operačních systémů na bázi Fedory včetně odvozených distribucí jako jsou RHEL, CentOS a ostatní. [37]

LiveCD Tools zapíše obraz systému tak, že při spuštění z „živého“ média (v případě této práce USB paměťový disk) by se měl systém jevit co nejvíce jako standardní systém se všemi funkcemi, například čtení a zápis na standardní souborový systém ext4.[37]

Další funkcí programu je, aby „živý“ obraz byl „instalovatelný“, tedy aby uživatel měl možnost nainstalovat z ISO obrazu na USB paměťový disk, aniž by tento proces vyžadoval přístup k síti nebo další média.[37]

Tím, že sada nástrojů je oddělená od konfigurace, je možné použít stejný nástroj pro vytvoření různých „živých“ médií s různými konfiguracemi, například „živé“ médium s uživatelským prostředím GNOME, s prostředím KDE a další. I přes tyto výhody nástroj nedoporučuji, více je pak vysvětleno v kapitole o přípravě systému. [37]

6 Návrh bezpečnostího programu

Tato kapitola seznamuje s programem Ssec a jeho funkcemi. Nejdříve budou popsány jeho cíle a obecný návrh, dále budou popsány konkrétnější úkoly v implementaci.

6.1 Motivace

Cílem je navrhnout a implementovat zabezpečovací program jménem Ssec, který by automaticky dešifroval zašifrovaný diskový oddíl. Jmenuje se Ssec kvůli tomu, že je implementován společně s ostatními programy pro seniory 90+, které usnadňují seniorům naučení se práce s počítačem. „S“ je reprezentuje slovo „senior“ a „sec“ reprezentuje slovo „security“.

Program by rozhodoval na základě daného identifikátoru, v tomto případě MAC adresy, zda má automaticky diskový oddíl dešifrovat. Jako základ bude použita „živá“ verze Ubuntu se separátním diskovým oddílem pro uživatelská data. Dále Ubuntu musí podporovat persistenci dat, aby bylo možné v kořenovém adresáři nechat program Ssec, aniž by se přemazal po restartu.

Celý operační systém se nachází na USB paměťovém disku, který je pro seniory pohodlný na používání. Senioři si mohou na operační systém zvyknout a poté si ho nosit s sebou kamkoliv chtějí a zapojit USB paměťový disk do kteréhokoliv počítače a spustit na něm svůj systém. Nedostatkem tohoto řešení je samozřejmě bezpečnost, například pokud by senior USB paměťový disk někde zapomněl, případný útočník by pak měl k dispozici uživatelská data v otevřeném textu. Proto zabezpečovací program Ssec má za úkol zašifrovat část USB paměťového disku, aby se zamezilo neoprávněnému přístupu k datům při zapomenutí nebo krádeži.

Adresář `/encrypted` bude uchovávat zašifrovaná uživatelská data. Zároveň bude uchovávat změny, není tedy potřeba zařizovat persistenci dat. Adresář se totiž bude nacházet na jiném diskovém oddílu a nedojde tak k navrácení původního stavu formátu ISO. Persistence dat je nutná na oddílu s kořenovým adresářem, vzhledem k tomu, že zde se budou nacházet části programu Ssec, takže nemůžeme dovolit smazání.

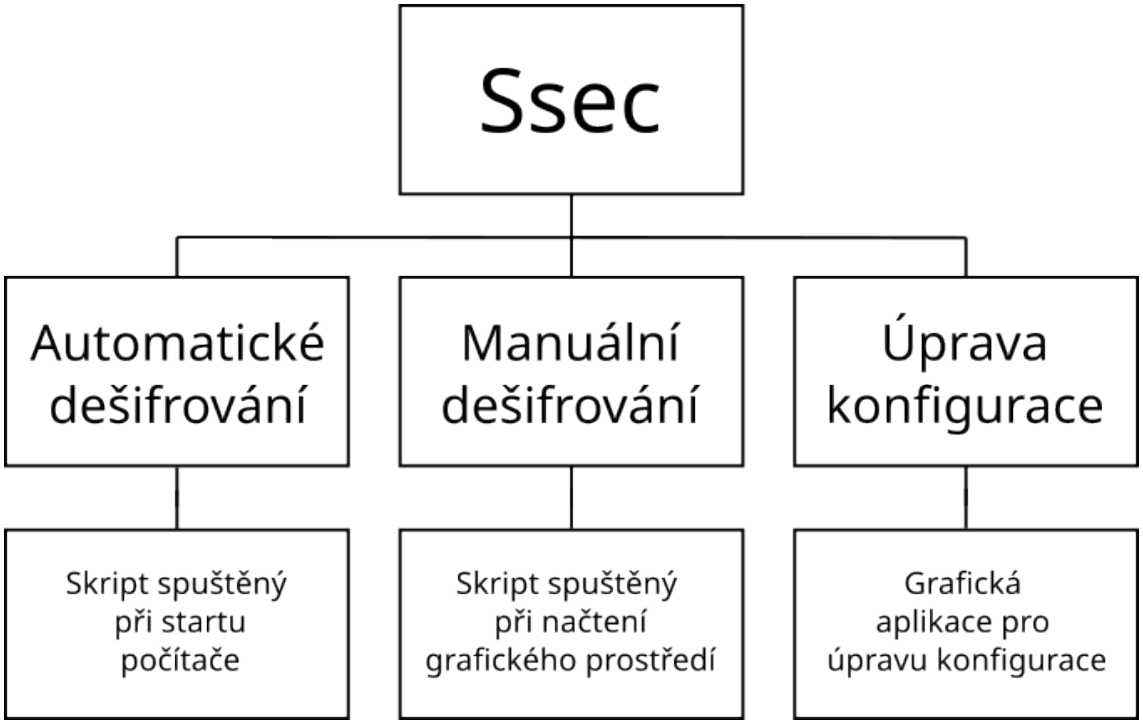
6.2 Rozdělení implementace

Ssec se bude skládat ze tří skriptů, kde každý bude mít jeden úkol. Jako první úkol Ssec by umožňoval automatické dešifrování při startovacím procesu počítače, dále by umožňoval zadání hesla manuálně při načtení uživatelského prostředí a ještě by umožňoval přidávání povolených MAC adres do seznamu v konfiguračním souboru společně s dalšími funkcemi.

Při implementaci bude nutné, aby byl program rozdělen do více souborů, vzhledem k tomu, že každá část se spouští při jiné úrovni startovacího procesu počítače. Automatické dešifrování bude probíhat již při zapínaní počítače, pro větší bezpečnost a pohodlnost užívání.

Okno pro zadání hesla nebo tabulka „automatické dešifrování proběhlo úspěšně“ se bude zobrazovat až po načtení plochy, vzhledem k tomu, že startovací proces počítače není stavěný na uživatelský vstup.

Jako poslední bude program Ssec umožňovat zadání povolených MAC adres do konfiguračního souboru. Aplikace bude mít jednoduché uživatelské rozhraní, ve kterém bude moct asistent seniora zadat povolené MAC adresy.



Obr. 6.1: Návrh funkcí programu Ssec

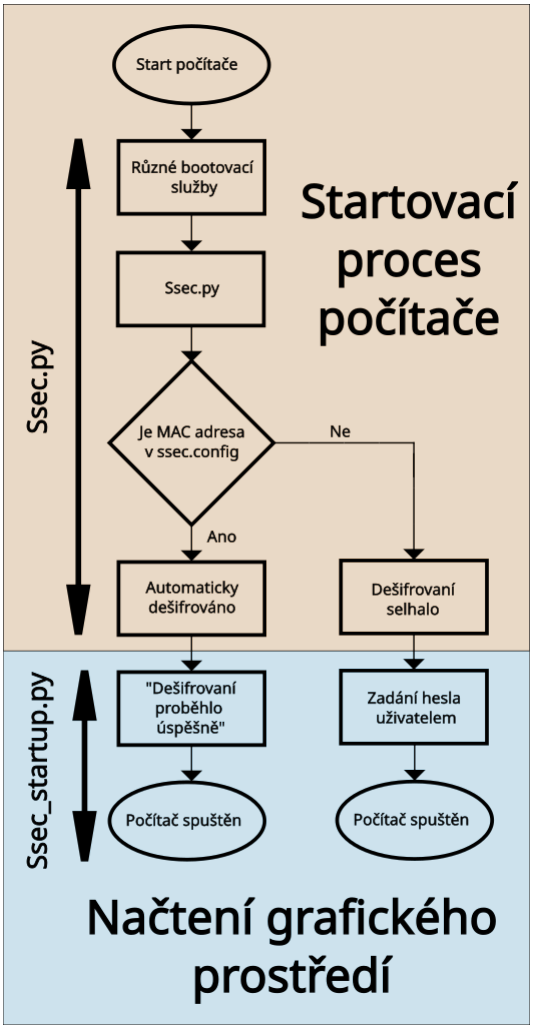
Na obrázku 6.1 můžete vidět shrnutí funkcí programu Ssec. Dále v práci jsou kapitoly rozděleny podle tohoto schématu pro větší přehlednost. Celý program je napsaný v jazyce Python.

7 Implementace

V této kapitole je podrobně popsána implementace programu Ssec. Nejdříve je popsáno spouštění celého operačního systému s vysvětlením role některých skriptů ve startovacím procesu. Navazuje popis přípravy „živého“ operačního systému a jeho vlastností, které je potřeba zavést pro správné fungování programu Ssec. Dále jsou popsány tři úkoly programu Ssec: automatické dešifrování, manuální dešifrování a úprava konfiguračního souboru. Mezi tím je ještě popsán konfigurační soubor. Kapitola je zakončena nastíněním útoků na systém a jak by jim bylo zabráněno.

7.1 Spouštění systému

Na diagramu 7.1 můžete vidět, kdy se části programu Ssec spouštějí a zjednodušeně jakou mají funkci.



Obr. 7.1: Vývojový diagram programu Ssec

Schéma je rozděleno na dvě části: startovací proces a grafické prostředí (plocha). Při startovacím procesu se zapíná zařízení, kde se postupně zapínají služby nutné pro běh operačního systému, mezi nimi se spouští i služba `ssec.py`.

Po zapnutí všech stěžejních služeb a načtení grafického prostředí se spustí program `ssec_startup.py`, který uživateli potvrdí úspěšné dešifrování nebo se ho zeptá na heslo.

7.2 Příprava prostředí

V této části jsou popsány vlastnosti, které jsou potřeba zajistit pro správné fungování programu Ssec.

7.2.1 Požadavky prostředí

Pro správné fungování systému zabezpečení Ssec jsou nutné 3 věci:

- instalace souboru s příponou `.iso` souboru s operačním systémem na konkrétní (první) diskový oddíl bez formátování celého disku,
- persistence dat na diskovém oddílu s operačním systémem,
- diskový oddíl nacházející se na sektorech za diskových oddílem s operačním systémem určený pro uživatelská data.

Podrobný návod pro reprodukci prostředí se nachází v příloze pod názvem `README.md`.

7.2.2 USB paměťový disk

Pro instalaci operačního systému a programu Ssec je potřeba dostatečně velká kapacita USB paměťovém disku. Nejmenší možná velikost je 16GB volného místa.

Pro operační systém je potřeba 8GB místa, dále je možné si určit velikost diskového oddílu s domovským adresářem. Doporučená velikost oddílu s uživatelskými daty je 16GB.

Konkrétně byl použit disk ADATA USB Flash Drive 3.0 (UV150/16GB). Dále pro vyzkoušení některých funkcionalit programu byl použit disk USB Flash Kingston DataTraveler Exodia Onyx 64GB USB 3.2 (DTXON/64GB).

7.2.3 Dělení disku

Pro rozdělení disku na diskové oddíly byl použit nástroj `fdisk`. V nástroji `fdisk` bylo navoleno vytvoření nového diskového oddílu o velikosti 8GB. Dále je potřeba vytvořit diskový oddíl o velikosti 4GB (či více, pokud je dostupný větší USB paměťový disk), který je určený pro uživatelská data.

Na konci disku se musí zanechat místo, které nebude přiřazené žádnému diskovému oddílu. Je to nutné pro správné vytvoření datové persistence instalačním programem Unetbootin, který bude použit v dalším kroku. Dělení disku je nutné provést před instalací operačního systému. Je to nutné pro správné fungování zavaděče.

7.2.4 Instalace operačního systému

Pro instalaci operačního systému byl použit program Unetbootin, který umožňuje rozbalit obraz ISO operačního systému na konkrétní diskový oddíl. Obraz ISO operačního systému Ubuntu se rozbalí na první diskový oddíl a označí se jako „spustitelný“, aby zavaděč poznal, že se jedná o operační systém. Standardně se při instalaci obrazu ISO formátuje celý disk, avšak to by přemazalo připravené diskové oddíly.

7.2.5 Verze s operačním systémem Fedora

Pomocí programu LiveCD tools byla nainstalována „živá“ verze operačního systému. Po instalaci LiveCD stačí v příkazovém řádku následující příkaz:

```
$ livecd-iso-to-disk --nomac --force --noverify
--overlay-size-mb- 4080 --unecrypted-home
Fedora-Workstation-Live-x86\_64-46-1.5.iso /dev/sdb1
```

Bohužel nástroj LiveCD tools je zastaralý a umožnil nainstalovat pouze Fedoru verze 36, která není nejnovější. Novější verze nástroj LiveCD nepodporuje. Stejně tak LiveCD podporuje pouze Legacy GRUB, takže systém není možné spustit na některých novějších počítačích.

Proto také byl pro instalaci operačního systému zvolen program Unetbootin a operačním systémem Ubuntu. Program Unetbootin podporuje nejnovější verzi Ubuntu.

7.2.6 Instalace Ssec

Skripty programu Ssec je nutné uložit do konkrétních adresářů. To udělá instalační skript **`install.sh`**, napsaný v jazyce Bash. Při spuštění skriptu je nutné vybrat správný disk z nabídky, dále totiž s názvem disku pracují některé funkce.

Hlavní skripty v jazyce Python jsou umístěny v adresáři **`/ssec`**. Služba Systemd **`ssec.service`**, která spouští program pro automatické dešifrování při startovacím procesu počítače, se pak nachází mezi ostatními službami Systemd v adresáři **`/etc/system/systemd`**.

Služba **ssec.desktop**, která spustí aplikace pro manuální zadání hesla po načtení grafického prostředí, se nachází v adresáři **/etc/xdg/autostart**.

7.3 Automatické dešifrování

V této části bude popsána hlavní funkce programu Ssec, což je automatické dešifrování na základě identifikátoru počítače, v tomto případě MAC adresy.

Automatické dešifrování probíhá v souboru **ssec.py** a je nejdůležitější pro běh celého programu Ssec kvůli tomu, že na něj další části navazují. V tomto souboru se nachází kód, který který dělá samotné automatické dešifrování pomocí kryptografického modulu jádra. Zašifrovaný diskový oddíl je ve formátu LUKS.

7.3.1 Přečtení MAC adresy

V následující funkci (výpis 7.2) se zjistí MAC adresa zařízení, ve kterém se nachází USB paměťový disk s operačním systémem. Tato MAC adresa se nejdříve zapíše do souboru. Je to uděláno tímto způsobem, protože bylo možné se setkat s neviditelnými znaky, které nebylo možné odstranit.

MAC adresu program načte ze souboru a uloží do proměnné **computer_mac**, kterou i tato funkce vrací. Dále se s MAC adresou pracuje při automatickém dešifrování, kde podmínka porovnává adresu zařízení s adresami nacházejícími se v konfiguračním souboru **ssec.config**.

```
1 def mac_address_function():
2     subprocess.run(f"cat /sys/class/net/*/address | awk 'NR==1
  ↪  {{print; exit}}' > /ssec/ssec_tmp.txt 2>/ssec/ssec_err.log",
  ↪  shell=True)
3     tmp_path = '/ssec/ssec_tmp.txt'
4     with open(tmp_path, 'r') as file:
5         computer_mac = file.readline().strip()
6     subprocess.run(f'echo "yes" | rm /ssec/ssec_tmp.txt
  ↪  2>/ssec/ssec_err.log', shell=True)
7     return computer_mac
```

Výpis 7.2: Funkce pro přečtení MAC adresy počítače

7.3.2 Funkce automatického dešifrování

V první části funkce (výpis 7.3) jsou inicializovány proměnné **mapper_device**, **mount_point** a **encrypted_device**. Proměnná **Mapper_device** v sobě má uloženou cestu k namapovanému virtuálnímu disku, který je potřeba pro správné fungování zašifrovaného diskového oddílu. Proměnná **Mount_point** v sobě má uloženou cestu k adresáři, kde bude možné vidět dešifrovaný diskový oddíl v souborovém systému.

```
encrypted_device = find_encrypted_device()
mapper_device = '/dev/mapper/EncHome'
mount_point = '/home/encrypted'
inside_identification = False
computer_mac = mac_address_function()
index = 0
```

Výpis 7.3: První část funkce automatického dešifrovaní

Proměnná **Encrypted_device** v sobě má uloženou cestu k samotnému zašifrovanému disku. Funkci můžete vidět ve výpisu 7.4. Funkce spustí podproces, který provede příkaz **lsblk**. Tento příkaz vyhledá všechny disky v zapojené v počítači.

```
def find_encrypted_device():
    try:
        output = subprocess.check_output(["lsblk", "-o",
        ↪  "NAME,VENDOR,MODEL", "-d",
        ↪  "-n"]).decode("utf-8").splitlines()
        encrypted_device = ""
        for line in output:
            if "sd" in line:
                flash_name = find_in_config("[Flash name]")
                if flash_name.replace(" ", "").lower() in
                ↪  line.replace(" ", "").lower():
                    parts = line.split()
                    encrypted_device = f"/dev/{parts[0]}2"
        return encrypted_device
    except:
        return "/dev/sdb2"
```

Výpis 7.4: Funkce pro získání cesty k zašifrovanému diskovému oddílu

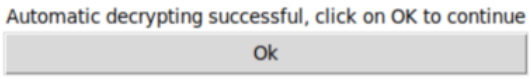
Dále se porovnává jméno USB paměťového disku, které bylo zvoleno při instalaci, s každým záznamem získaným z příkazu pro vypsání disků v počítači (**lsblk**). Funkce pak vrátí cestu k USB paměťovému disku, kde je nainstalovaný operační systém.

Pokud nastane výjimka, funkce vrátí **/dev/sdb2**. Takto je to udělané kvůli tomu, že USB paměťový disk je nejpravděpodobněji druhý disk v počítači, avšak není to podmínka. Pokud program Ssec nevrátí správný diskový oddíl, nedojde k žádné akci, která by ohrozila uživatelská data, program nedělá žádné destruktivní úkony.

Cyklus ve výpisu 7.6 má za úkol zjistit, kde se v konfiguračním souboru **ssec.config** nacházejí MAC adresy. Pod nadpisem **[Identification]** jsou uložené MAC adresy. Jakmile cyklus narazí na tento nadpis, začne ukládat všechny řádky pod touto sekcí do seznamu, kde na každém řádku je jedna povolená MAC adresa. Všechny řádky se porovnají s MAC adresou zařízení (ve kterém se nachází USB paměťový disk) v podmínce která je popsána dále.

Ve výpisu 7.6 můžeme vidět vnitřní část cyklu, ve kterém probíhá automatické dešifrování. Cyklus přečte každou MAC adresu ze souboru **ssec.config** a porovná ji v podmínce s adresou počítače, ve kterém se nachází USB paměťový disk. Pokud se adresy shodují, v podmínce se provede dešifrování diskového oddílu a vytvoří se virtuální disk pro čtení a zápis. Pokud je dešifrování úspěšné, nasadí se virtuální disk, který je připravený k použití.

Pokud se v konfiguračním souboru **.ssec.config** nachází textový řetězec „Decrypted“, při načtení grafického uživatelského prostředí se objeví okno s potvrzovacím tlačítkem a textem „automatické dešifrování proběhlo úspěšně“ (obr. 7.5).



Obr. 7.5: Grafické potvrzovací okno automatického dešifrování

```
for line in file:
    if line.strip() == '[Identification]':
        inside_identification = True
        continue
    if inside_identification:
        if line == '\n':
        break

    if verify_hash(line, computer_mac):
    password = decrypt(get_conf_index(index,
            '[Enc pass]\n')).decode('utf-8')
    subprocess.run(
            ['cryptsetup', 'luksOpen',
            encrypted_device, 'EncHome'],
            input=password.encode(),

            stderr=subprocess.PIPE)
    subprocess.run(
            ['mount', mapper_device, mount_point],
            stderr=subprocess.PIPE)
    pop_in_config('Decrypted\n', '[Ssec decrypt]\n')
    create_log("", index)
    break
```

Obr. 7.6: Nalezení porovnání MAC adres v konfiguračním souboru

7.4 Manuální dešifrování

V této kapitole bude popsáno, jak se program Ssec chová v případě, kdy se USB paměťový disk nachází v počítači, který není mezi povolenými. Bude popsáno okno pro manuální zadání uživatelského hesla pro dešifrování uživatelských dat.

Okno pro manuální zadání hesla se nachází v souboru **ssec_startup.py**, kapitola se zaměří na popsání implementace jeho funkcí.

7.4.1 Grafická okna

Manuální zadání hesla pro dešifrování zašifrované části diskového oddílu probíhá ve skriptu **ssec_startup.py**. Skript ssec_startup.py používá knihovnu jazyku Python Tkinter, aby byl styl grafického uživatelského prostředí jednotný s ostatními programy ForSenior-OS.

V této části kódu (výpis 7.7) probíhá přečtení konfiguračního souboru. V konfiguračním souboru **.ssec.config** se pod sekcí **[Ssec decrypt]** nachází jeden ze 4 textových řetězců - „Encrypted“, „Decrypted“, „Insert“ a „default“.

```
if insert_line == 'Insert':
    frame = tk.Frame(root, bg="white")
    frame.grid(row=0, column=0, padx=10, pady=10, sticky='nsew')
    insert_frame()
elif decrypt_line == 'Decrypted':
    frame = tk.Frame(root, bg="white")
    frame.grid(row=0, column=0, padx=10, pady=10, sticky='nsew')
    decrypted_frame()
else:
    frame = tk.Frame(root, bg="white")
    frame.grid(row=0, column=0, padx=10, pady=10, sticky='nsew')
    encrypted_frame()
```

Výpis 7.7: Podmínka rozhodující o tom, jaké okno má uživateli zobrazit

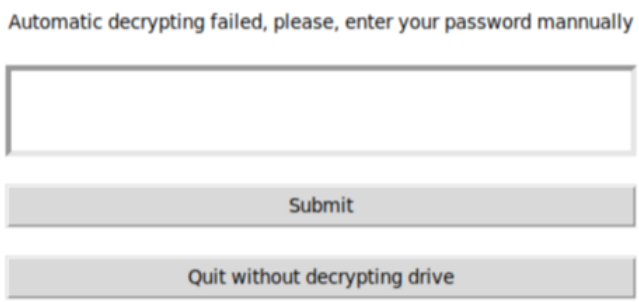
Jak již bylo napsáno v předchozí sekci, pokud se v konfiguračním souboru nachází textový řetězec „Decrypted“, zobrazí se oznamovací okno (obr. 7.5). Po potvrzení tabulky se v konfiguračním souboru textový řetězec „Decrypted“ přepíše na „default“ a poté je uživateli umožněno používat operační systém.

Pokud se v konfiguračním souboru nachází textový řetězec „Encrypted“, zobrazí se uživateli následují okno, které ho vyzývá na zadání hesla (7.8). Po zobrazení

tohoto okna automatické dešifrování selhalo a pro přístup k datům je nutné znát heslo. Data jsou ve chvíli zobrazení okna stále zašifrovaná.

Textový řetězec „Encrypted“ se do konfiguračního souboru zapíše pokud se MAC adresy v konfiguračním souboru neshodují s MAC adresou počítače, ve kterém se nachází USB paměťový disk. Zároveň se textový řetězec zapisuje, pokud jakýmkoliv způsobem selže automatické dešifrování.

Pro případ, že selže celá skript **`ssec.py`**, zůstane v konfiguračním souboru řetězec **`default`**. Okno pro zadání hesla (obr. 7.8) se zobrazí i takovém případě. V podmínce je možné vidět, že zobrazení grafického okna je (výpis 7.7) uděláno jako případ **`else`**, aby při jakémkoliv selhání některé části programu Ssec bylo umožněno uživateli zadat heslo a přistoupit k datům.



Obr. 7.8: Grafické okno pro manuální zadání hesla

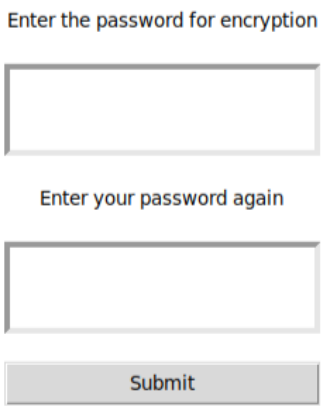
Ve výpisu 7.9 vidíte funkci, která pak předá uživatelské heslo operačnímu systému. Kryptografický nástroj **`dm-crypt`** následně dešifruje diskový oddíl.

Poslední možnost textového řetězce, která se může nacházet v konfiguračním souboru, je „Insert“. Okno (obr. 7.10) se zobrazí po přidání povolené MAC adresy pomocí grafické aplikace pro úpravu konfigurace. Aplikace je popsána v kapitole Úprava konfigurace. Obě okna (obrázky 7.10 a 7.8) se liší kvůli tomu, že heslo zadané v tomto okně se zašifruje pomocí UUID počítače, ve kterém se nachází a uloží se do konfiguračního souboru pro budoucí automatické dešifrování.

Řešení načítáním textových řetězců z konfiguračního souboru (výpis 7.7) může na první pohled vypadat, že není vhodné z pohledu bezpečnosti, kde by případný útočník mohl konfigurační soubor změnit před spuštěním skriptu **`ssec_startup.py`**.

```
def submit(pass_entry):
    input_pass = pass_entry.get()
    if decryption_process(input_pass):
        pop_in_config('default\n', '[Ssec decrypt]\n')
        root.quit()
    else:
        error_window(root, "Wrong password")
        return
```

Výpis 7.9: Funkce submit()



Obr. 7.10: Grafické okno pro zadání hesla na zašifrování

Avšak program Ssec podle konfiguračního souboru pouze určuje, které okno grafického uživatelského rozhraní spustí.

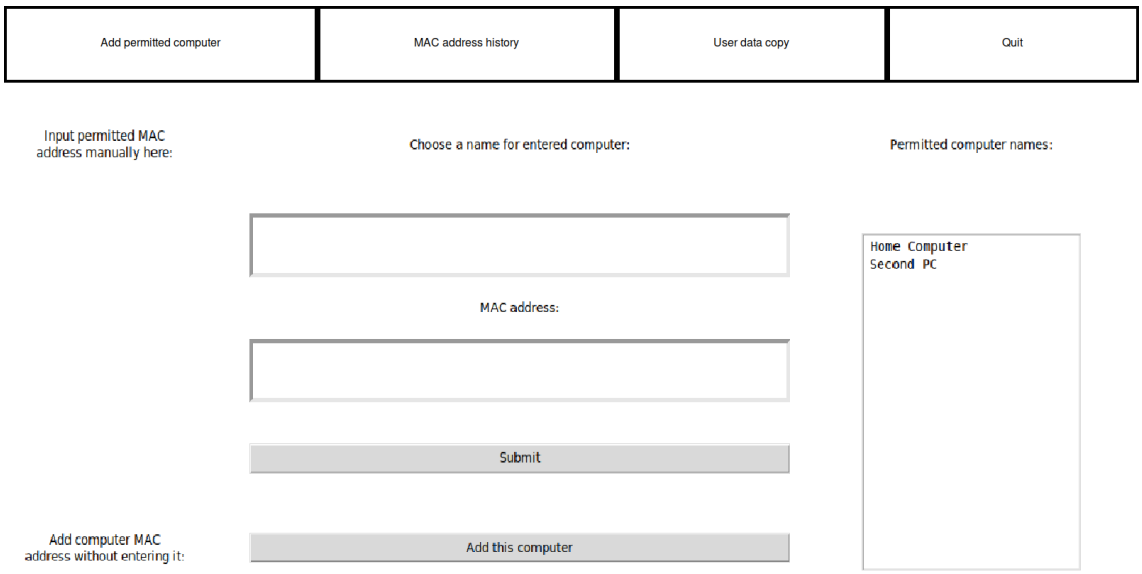
Pokud by tedy útočník do souboru **`ssec_crypt.txt`** napsal řetězec „Decrypted“, pouze by způsobil, že by se mu spustilo okno s potvrzením o úspěšném dešifrování, avšak nedošlo by k samotnému dešifrování, data by byla pořád chráněna kryptografickým formátem LUKS. Útoky na program Ssec budou podrobněji popsány v kapitole Implementace zabezpečení.

7.5 Úprava konfigurace

Zde bude popsán program pro úpravu konfigurace, jeho funkce a implementace. Mezi funkce patří přidání povoleného počítače pro automatické dešifrování zadáním MAC adresy a jeho aliasu, zobrazení historie, ve které se nachází čas a MAC adresa nebo alias počítačů, ve kterých se USB paměťový disk nacházel a možnost zkopírovat uživatelská data ze zašifrovaného oddílu na další disk.

7.5.1 Přidání povoleného počítače

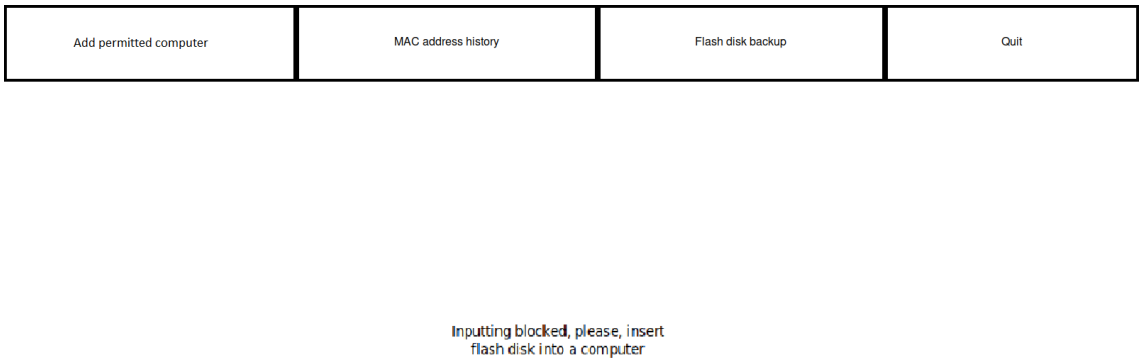
Jedním z cílů této práce je vytvoření grafického programu pro přidávání povolených MAC adres. Bylo tedy vytvořeno bezokrajové okno pomocí grafické knihovny Tkinter. Asistent seniora může jednoduše spustit aplikaci **`ssec_app.py`** a zobrazí se mu následující grafické okno (obr. 7.11). Po zadání adresy a aliasu stačí tlačítkem potvrdit přidání adresy do konfiguračního souboru **`/ssec/ssec.config`**.



Obr. 7.11: Okno pro zadání povolených MAC adres

7.5.2 Vytvoření otisku

Po přidání povolené adresy se přidávání dalších povolených MAC adres zablokuje (obr. 7.12), jelikož program Ssec potřebuje zašifrovat uživatelské heslo pomocí UUID daného počítače, k tomu je však nutné USB paměťový disk do daného počítače vložit a spustit operační systém.



Výpis 7.12: Blokovací okno zobrazené po zadání MAC adresy

Předtím než se MAC adresa zapíše do konfiguračního souboru, vytvoří se její otisk pomocí algoritmu Argon2. Ve výpisu 7.13 můžete vidět použití knihovny argon2-cffi pro vytvoření otisku MAC adresy a následné zapsání do konfiguračního souboru. Funkce pro práci s konfiguračním souborem jsou popsány dále. Z bezpečnostního hlediska je špatné mít zapsané MAC adresy v prostém textu, v kapitole Implementace zabezpečení je vysvětlené proč.

```
1 def hash_mac_func(input_mac):
2     ph = PasswordHasher()
3     mac_hash = ph.hash(input_mac.strip())
4 
5     insert_to_config(mac_hash, '[Identification]\n')
6     pop_in_config('Insert\n', '[Ssec insert]\n')
```

Výpis 7.13: Vytvoření otisku MAC adresy

7.5.3 Šifrování uživatelského hesla

Poté co uživatel zadá novou povolenou MAC adresu a vloží USB paměťový disk do daného počítače, zobrazí se mu okno (obr. 7.10) k zadání hesla pro zašifrování. Heslo

se po zadání zašifruje algoritmem AES-128. Jako klíč se použije UUID systému, ve kterém se USB paměťový disk právě nachází. Funkci, která získá UUID je možno vidět ve výpisu 7.14. Heslo získané z okna pro zadání (obr. 7.10) se zašifruje pomocí funkce ve výpisu 7.15

Heslo se šifruje pomocí UUID, aby nebylo nutné kryptografický klíč ukládat na nešifrované části USB paměťového disku. Kdyby kryptografický klíč byl uložen na USB paměťovém disku, útočník by mohl k němu jednoduše přistoupit a dešifrovat uživatelská data.

```
1 def get_key():
2     subprocess.run(f'cat /sys/class/dmi/id/product_uuid >
      ↪  /ssec/ssec_uuid.txt', shell=True)
3     uuid_path = '/ssec/ssec_uuid.txt'
4     with open(uuid_path, 'r') as file:
5         uuid = file.readline().strip().replace('-', '')
6     subprocess.run(f'echo "yes" | rm /ssec/ssec_uuid.txt',
      ↪  shell=True)
7     return uuid.encode('utf-8')
```

Výpis 7.14: Funkce na vytvoření otisku MAC adresy

```
1 def encrypt(plain_text):
2     iv = 16 * b'\x00'
3     block_size = 16
4     cipher = AES.new(get_key(), AES.MODE_CBC, iv)
5     padded_text = pad(plain_text.encode('utf-8'), block_size)
6     cipher_text = cipher.encrypt(padded_text)
7     return base64.b64encode(cipher_text)
```

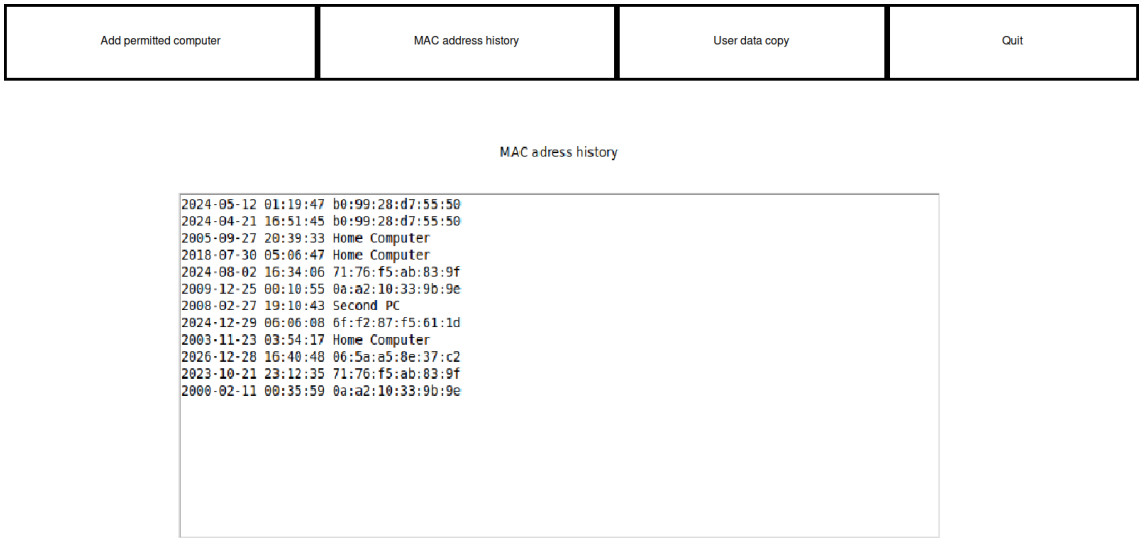
Výpis 7.15: Funkce na zašifrování uživatelského hesla

Aplikace dále umožňuje přidat počítač, na kterém je zrovna operační systém spuštěn, pomocí tlačítka „Add this computer“. Ulehčí to tak práci asistentovi seniora tím, že nemusí MAC adresu zadávat manuálně. MAC adresu program Ssec získá pomocí funkce ve výpisu 7.2.

7.5.4 Zobrazení historie

Zobrazení historie MAC adres je další funkcí (obr. 7.16), kterou program Ssec umožňuje. Aplikace jednoduše načte všechny záznamy z konfiguračního souboru. Do konfiguračního souboru ho zapisuje skript **`ssec.py`** při startu počítače poté, co se vyřeší podmínka rozhodující o automatickém dešifrování.

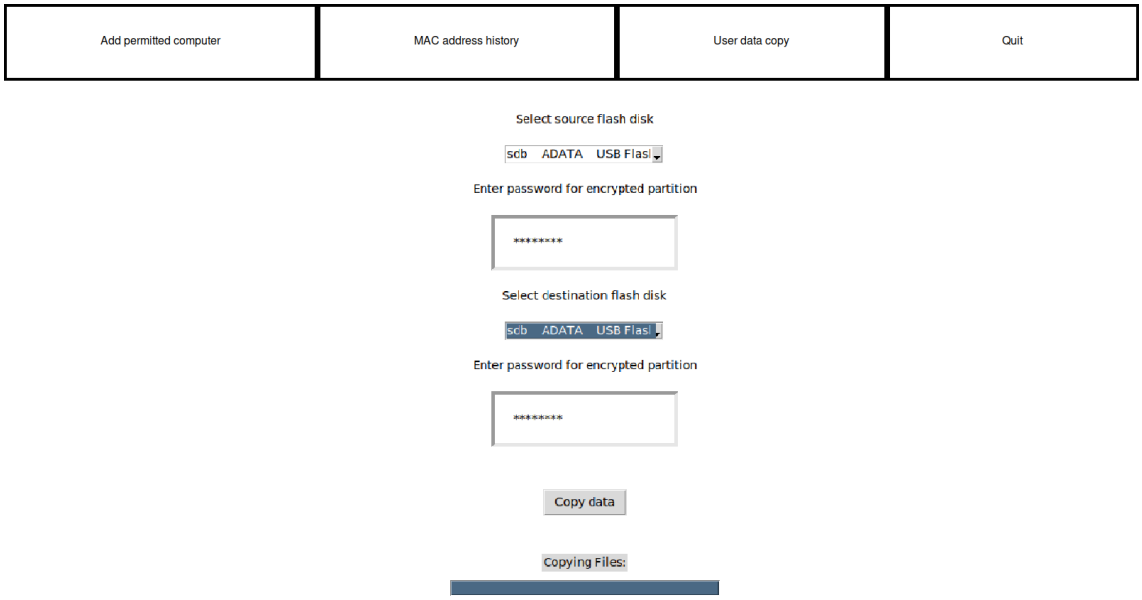
Záznam obsahuje časové razítko a buď MAC adresu daného počítače, ve kterém se nacházel, nebo jeho alias. V případě povolených počítačů se ukládá alias kvůli rizikům vysvětlených v sekci Implementace zabezpečení.



Obr. 7.16: Historie MAC adres

7.5.5 Záloha uživatelských dat

Poslední funkcí (obr. 7.17) aplikace pro úpravu konfigurace je zkopírování uživatelských dat na další disk, na kterém je nainstalovaný ForSenior-OS. V menu stačí vybrat dané USB paměťové disky, ze kterého a na který se mají data zkopírovat. Dále je potřeba zadat uživatelské heslo ke každému disku, jelikož se předpokládá, že oba disky budou mít zašifrovaný druhý diskový oddíl.



Obr. 7.17: Okno pro vytvoření zálohy uživatelských dat

7.6 Konfigurační soubor

V konfiguračním souboru **.ssec.config** se nacházejí všechny údaje nutné pro běh programu. Sekce popisuje strukturu konfiguračního souboru a funkce, které s konfiguračním souborem pracují.

7.6.1 Struktura konfiguračního souboru

Příklad toho, jak může konfigurační soubor vypadat, ukazuje výpis 7.18. Jedná se o unikátní formát vytvořený pro program Ssec. Je rozdělený do několika sekcí, aby funkce interagující s konfiguračním souborem mohly jednoduše vyhledat sekci a přečíst či změnit hodnotu pod ní.

Sekce **[Flash name]** pod sebou má jméno USB paměťového disku s operačním systémem pro seniory. Jméno je uložené kvůli tomu, aby bylo možné dohledat cestu k zašifrovanému diskovému oddílu v souborovém systému. Pomocí jména USB paměťového disku je pak možné dohledat cestu (například **/dev/sb2**).

Do sekce **[Ssec decrypt]** zapisuje skript **ssec.py** při startovacím procesu počítače. Zapíše jednu ze dvou možností textových řetězců, „Encrypted“ nebo „Decrypted“. Skriptu **ssec_startup.py** tak říká, zda bylo automatické dešifrování úspěšné nebo ne a tím pádem jaké grafické okno má uživateli zobrazit.

Stejně jako předchozí sekce, **[Ssec insert]** určuje programu ssec_startup.py jaké okno má zobrazit. Pokud textový řetězec pod polem je **Insert**, služba ssec.py se nespustí a automatické dešifrování neprobíhá a ssec_startup.py při načtení plochy zobrazí okno pro zadání uživatelského hesla, které se poté zašifruje pomocí UUID systému algoritmem AES-128.

Sekce **[Computer names]** v sobě má uložené aliasy k daným povoleným MAC adresám, které jsou ve formě otisku (haše). MAC adresy se ukládají v následující sekci **[Identification]**. Z této sekce se i načítá MAC adresa při pokusu o automatické dešifrovaní.

Sekce **[Enc pass]** v sobě má různá uživatelská hesla zašifrovaná algoritmem AES-128, každé heslo je zašifrované jiným identifikátorem UUID systému. To, jakým identifikátorem je heslo zašifrováno, koresponduje s daným otiskem povolené MAC adresy.

Sekce **[MAC history]** obsahuje záznamy o tom, do jakého počítače byl USB paměťový disk vložen. Každý záznam má časovou známku a MAC adresu nebo alias počítače, ve kterém byl. Aliasy se používají u záznamů s povolenými počítači, aby nedošlo k odhalení některé z povolených MAC adres. Rizika odhalení MAC adresy jsou popsány v sekci Implementace zabezpečení.

```
[Flash name]
ADATA USB Flash Drive

[Ssec decrypt]
default

[Ssec insert]
default

[Computer names]
Home computer

[Identification]
\$argon2i$v=19$m=16,t=2,p=1$YWhvamFob2o$B9t4XPjkhUgPSzNvt4P1FQ

[Enc pass]
YWhvamFob2phaG9qcw==

[MAC history]
2024-04-17 18:21:07 Home computer
2024-04-17 10:21:07 69:13:A9:D2:C4:A6
```

Výpis 7.18: Ukázka konfiguračního souboru

7.6.2 Zápis a čtení konfiguračního souboru

Pro zápis a čtení konfiguračního souboru se používají 3 funkce - 7.19, 7.20 a 7.21. Každá z těchto funkcí má trochu jiný účel. Funkce 7.19 má za úkol najít danou sekci v konfiguračním souboru a přečíst první řádek v sekci. Využívá se například při startu počítače pro rozhodnutí, jaké grafické okno má skript **ssec_startup.py** zobrazit.

Funkce 7.20 má za úkol najít a změnit první hodnotu v dané sekci. Používá se například pro uvedení sekce **[Ssec decrypt]** do původního stavu po potvrzení úspěšného automatického dešifrování.

Funkce 7.21 vkládá daný textový řetězec do dané sekce. Využívá se při zapisování časových známek s MAC adresou do sekce **[MAC history]**.

```
def find_in_config(section):
    inside_decrypt = False
    with open('/ssec/.ssec.config', 'r') as file:
        for line in file:
            if line.strip() == section.strip():
                inside_decrypt = True
                continue
            if inside_decrypt == True:
                return line.strip()
```

Výpis 7.19: Podmínka kontrolující, zda byl disk dešifrován

```
def pop_in_config(insert_text, insert_section):
    with open("/ssec/.ssec.config", "r") as f:
        content = f.readlines()
    insert_index = content.index(insert_section)+1
    content.pop(insert_index)
    content.insert(insert_index, insert_text)
    with open("/ssec/.ssec.config", "w") as f:
        f.writelines(content)
```

Výpis 7.20: Podmínka kontrolující, zda byl disk dešifrován

```
def insert_to_config(insert_text, insert_section):
    with open("/ssec/.ssec.config", "r") as f:
        content = f.readlines()
    insert_index = content.index(insert_section)+1
    content.insert(insert_index, insert_text+'\n')
    if len(content) == 0:
        return
    with open("/ssec/.ssec.config", "w") as f:
        f.writelines(content)
```

Výpis 7.21: Podmínka kontrolující, zda byl disk dešifrován

7.7 Implementace zabezpečení

V této sekci budou popsány možné útoky na program Ssec a ošetření proti těmto útokům. Dále budou popsána některá designová rozhodnutí z hlediska bezpečnosti.

7.7.1 Útok typu command injection

Command Injection je typ útoku, kdy útočník vkládá kód do vstupního pole, který je následně interpretován jako součást příkazu systému. To může vést k provádění neautorizovaných příkazů na cílovém stroji. Útoku se dá zabránit buď filtrováním vstupních dat od nebezpečných znaků a příkazů (sanitizaci) anebo použitím parametrizovaných dotazů pro komunikaci. Všechny znaky v parametrizovaném dotazu se pak počítají jako součást například hesla a nikoliv jako další příkaz.

Kvůli tomu, že program Ssec přímo zadává uživatelský vstup do příkazové řádky, je možné provést útok command injection, kdy útočník zadá do pole znak (například středník), který ukončí předchozí příkaz. Útočník poté může zadat libovolný příkaz. Tím, že program Ssec běží pod uživatelem **root**, tento příkaz by měl neomezená práva. Naivní implementace uživatelského vstupu do příkazového řádku může vypadat například takto (výpis 7.22):

```
subprocess.run(f'echo "{password}" | cryptsetup luksOpen
↪  {encrypted_device} EncHome 2>/ssec/ssec_err.log', shell=True)
```

Výpis 7.22: Zranitelný řádek

Proměnná **password** je získána z uživatelského vstupu, tudíž tento řádek kódu je zranitelný na útok command injection.

Aby kód zranitelný nebyl, byla použita následující funkce (výpis 7.23), která uživatelský vstup ošetří. Znaky, které by útočník zadal, by se počítaly jako součást hesla, nikoliv jako nový příkaz.

```
subprocess.run(
    ['cryptsetup', 'luksOpen',
    encrypted_device, 'EncHome'],
    input=password.encode(),
    stderr=subprocess.PIPE)
```

Výpis 7.23: Příkaz ošetřený proti útoku command injection

7.7.2 Prolomení otisku MAC adresy

Prolomení otisku (haše) by při dostatečně velkém výkonu počítače bylo možné, avšak tím, že je použit moderní algoritmus na vytváření otisků Argon2 (výpis 7.24), předpokládá se tedy vysoká bezpečnost proti prolomení.

```
1 from argon2 import PasswordHasher
2 def hash_mac_func(input_mac):
3         ph = PasswordHasher()
4         mac_hash = ph.hash(input_mac.strip())
```

Výpis 7.24: Funkce pro hashování MAC adresy vložené uživatelem

Znalost povolených MAC adres nemá přímý vliv na bezpečnost programu Ssec. Jak bude popsáno v další sekci, může vést například k lokalizaci v některých případech či k napadení bezdrátových sítí, které senior používá.

7.7.3 Změna MAC adresy počítače na povolenou

Změna MAC adresy (spoofing) je typ útoku, kdy útočník falšuje informace, aby získal neoprávněný přístup k aktivům. Když jde o spoofing MAC adresy, útočník mění MAC adresu svého zařízení tak, aby vypadalo, že patří jinému zařízení a to může vést k tomu, že systém v této práci zpřístupní data neoprávněnému uživateli. Dále může spoofing MAC adresy vést k neoprávněnému přístupu do bezdrátové sítě. Toto může vést k bezpečnostním problémům, jako je například odposlech dat.

Změna MAC adresy počítače na povolenou (spoofing) může představovat velkou hrozbu, nicméně design programu Ssec tomuto typu útoku zabraňuje. Útočník by sice prošel skrz důležitou podmínku určující, zda se má počítač automaticky dešifrovat, ale aby se tak úspěšně stalo, potřebuje k tomu znát uživatelské heslo, které je však zašifrováno kryptografickým algoritmem AES-128 pomocí UUID povoleného počítače, který by útočník stejné neměl. Data by tak zůstala zašifrovaná.

7.7.4 Neoprávněný přístup k programu pro přidávání MAC adres

Celý adresář **`/ssec`** je povolený pouze pro uživatele **`root`**. Avšak program pro přidávání povolených MAC adres se nenachází v zašifrované části disku.

To může vést k dojmu, že útočník by jednoduše přidal MAC adresu některého svého počítače. Avšak pokud by MAC adresu přidal, stále by to nevedlo k automatickému dešifrovaní dat. Po přidání MAC adresy je nutné zadat uživatelské heslo,

aby bylo možné ho zašifrovat algoritmem AES-128 kde jako klíč je použit 128-bitový identifikátor UUID.

7.7.5 Útok hrubou silou na UUID

Jsou dvě verze UUID, se kterými se systém může setkat – verze 4 a verze 1. UUID verze 4 je plně náhodná, až na jedinou číslici. Avšak i přestože se UUID verze 4 nedá označit za plně náhodný kryptografický klíč, náhodnost je dostatečná.

UUID verze 1 náhodné není, jeho největší část je generována pomocí času a náhodné MAC adresy. Kvůli nedostatečné náhodnosti nelze UUID verze 1 označit za bezpečný kryptografický klíč a uživatelská hesla zašifrovaná touto verzí UUID by byla zranitelná proti útoku hrubou silou.

Bohužel není žádný způsob, jak zajistit použití bezpečnější verze 4. Každý počítač má totiž vlastní implementaci toho, jak UUID generuje.

7.7.6 Neoprávněný přístup a změny v konfiguračnímu souboru

Jak již bylo řečeno, celý adresář **`/ssec`** je přístupný pouze pro uživatele **`root`**. To však zabrání přístupu pouze skrz operační systém. Je možné nešifrovanou část USB paměťovém disku otevřít a upravit jiným operačním systémem, než který na na USB paměťovém disku nachází.

To by umožnilo útočníkovi do jisté míry ovlivnit chování programu Ssec. Například by mohl změnit text, který se nachází v sekcích **`[Ssec decrypt]`** a **`[Ssec insert]`**. Pokud by změnil položku „default“ na „decrypted“, může to vést k dojmu, že by automaticky dešifroval zašifrovanou část disku, protože při načtení uživatelského prostředí operačního systému by se zobrazila tabulka s textem „Úspěšně automaticky dešifrováno“.

Avšak se jedná pouze o grafické okno, které slouží k informování uživatele. Automatické dešifrování probíhá při startovacím procesu počítače. Do konfiguračního souboru je možné zasáhnout, ale MAC adresy jsou ve formě otisku a uživatelská hesla jsou zašifrovaná. Obojí je tedy nečitelné. Zároveň případné zásahy je možné udělat, avšak neovlivní to bezpečný provoz programu Ssec.

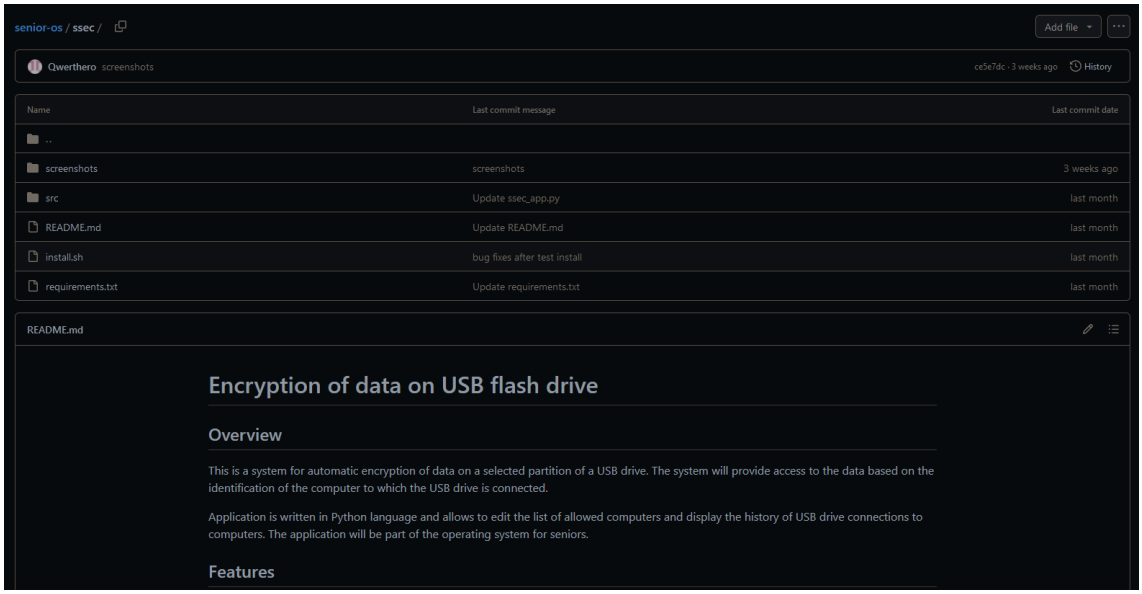
7.8 Struktura Ssec

Zde je podrobně popsán účel každého programu v adresáři Ssec. Každý soubor má jiný účel a je spouštěn při různých úrovních startovacího procesu počítače.

- **`ssec.py`** – skript **`ssec.py`** je hlavní část programu Ssec, zde probíhá zjišťování MAC adres zařízení, do kterých je USB paměťový disk vložen a dále případně dělá automatické dešifrování, pokud se MAC adresa daného zařízení nachází v souboru **`.ssec.config`**
- **`ssec_startup.py`** – **`ssec_startup.py`** je pomocný skript, který se spouští až při načtení grafického uživatelského prostředí operačního systému, buď v podobě potvrzovacího okna úspěšného automatického dešifrování nebo v podobě okna pro zadání hesla pro manuální dešifrování
- **`ssec_app.py`** – aplikace **`ssec_app.py`** umožňuje asistentovi seniora interagovat s programem Ssec, může zde nastavovat povolené MAC adresy, zobrazit historii počítačů nebo zálohovat uživatelská data
- **`ssec_func.py`** – skript **`ssec_func.py`** v sobě má funkce a knihovny, které používají ostatní skripty
- **`.ssec.config`** – v souboru **`.ssec.config`** se nachází povolené MAC adresy a ostatní informace, které program Ssec potřebuje pro svůj běh
- **`ssec.service`** – **`ssec.service`** je jednotkový soubor uložený v adresáři **`/etc/systemd/system/`** a pro program Systemd popisuje, kdy a jak se má daný skript, v tomto případě **`ssec.py`**, spustit při startovacím procesu operačního systému
- **`ssec.desktop`** – **`ssec.desktop`** je jednotkový soubor uložený v adresáři **`/usr/share/autostart`**, soubory v tomto adresáři jsou aplikace, které se spouští při načtení grafického uživatelského prostředí

7.9 Zveřejnění na platformě Github

Všechen kódový výstup této práce byl zveřejněn na platformě Github. Program je součástí ForSenior-OS a je dostupný na tomto odkazu `https://github.com/forsenior/senior-os/tree/main`. Veškerý kód této práce je dostupný v příloze společně s instrukcemi na instalaci a dokumentací v souboru **`README.md`**.



Obr. 7.25: Zveřejnění na platformě Github

Závěr

Cílem práce bylo vytvoření systému pro automatické šifrování dat na zvoleném oddílu USB paměťovém disku a zpřístupnění těchto dat na základě identifikace počítače, do kterého se USB paměťový disk připojí. Dalším cílem bylo vytvořit aplikaci v jazyce Python pro editaci seznamu povolených počítačů. Aplikace je součástí operačního systému pro seniory s názvem ForSenior-OS.

Výsledkem je fungující operační systém se zašifrovanými uživatelskými daty, který se nachází na jiném diskovém oddílu než je operační systém. Součástí je zabezpečovací program Ssec, který umí automaticky dešifrovat uživatelská data, pokud se MAC adresa počítače shoduje s alespoň jednou z MAC adres v konfiguračním souboru. Součástí řešení je grafická aplikace pro přidávání povolených počítačů, která umí i zobrazit historii počítačů, ve kterých se USB paměťový disk nacházel a zálohování uživatelských dat na další disk. Mezi možná vylepšení patří integrace grafického stylu s ostatními programy pro ForSenior-OS.

Byla provedena bezpečnostní analýza programu, díky které bylo možné zjistit případné zranitelnosti programu Ssec. Program Ssec je tak možné označit za bezpečný. Slabým místem však může být šifrování pomocí UUID, které nemusí ve všech implementacích být dostatečně náhodné. Bohužel ale není možné zajistit použití bezpečnější verze, jelikož každý počítač má vlastní implementaci generování UUID.

Zhotovený programový kód bezpečnostního programu je dostupný na platformě GitHub: `https://github.com/forsenior/senior-os/tree/main/ssec`.

Literatura

[1] RED HAT, INC. *What is Linux?*. Online. Red Hat. 2023. Dostupné z: https://www.redhat.com/en/topics/linux/what-is-linux. [cit. 2023-12-10].

[2] STALLMAN, Richard. *Linux and the GNU System*. Online. GNU. 2024. Dostupné z: https://www.gnu.org/gnu/linux-and-gnu.html. [cit. 2024-05-06].

[3] *Odpovědi na časté otázky: Co je to Linux a co GNU/Linux?*. Online. 2023. Dostupné z: http://www.linux.cz/. [cit. 2023-12-10].

[4] *Co je Fedora*. Online. 2023. Dostupné z: https://mojefedora.cz/co-je-fedora/. [cit. 2023-12-11].

[5] CANONICAL LTD. *O ubuntu*. Online. 2024. Dostupné z: https://www.ubuntu.cz/o-ubuntu/. [cit. 2024-05-06].

[6] LAU, Chase. *Creating and using a live installation image*. Online. 2019. Dostupné z: https://docs.fedoraproject.org/en-US/quick-docs/creating-and-using-a-live-installation-image/index.html#Data_persistence. [cit. 2023-12-11].

[7] OPENIZE PTY LTD. *What is an ISO file?*. Online. 2024. Dostupné z: https://docs.fileformat.com/compression/iso/. [cit. 2024-05-06].

[8] PFEIL, Matt. *What is data persistence and why does it matter?*. Online. 2010. Dostupné z: https://www.datastax.com/blog/what-persistence-and-why-does-it-matter. [cit. 2023-12-11].

[9] *What is a MAC adress and how does it work?*. Online. 2023. Dostupné z: https://www.howtogeek.com/764868/what-is-a-mac-address-and-how-does-it-work/. [cit. 2023-12-11].

[10] *What is a MAC Address: How to Find and Identify*. Online. 2023. Dostupné z: https://whatismyipaddress.com/mac-address. [cit. 2023-12-11].

[11] *MDN Web Docs Glossary: Definitions of Web-related terms: UUID*. Online. 2023. Dostupné z: https://developer.mozilla.org/en-US/docs/Glossary/UUID. [cit. 2023-12-11].

[12] *What's a UUID?: What are Universal Unique Identifiers and how do they work?*. Online. 2023 Dostupné z: https://www.uuidtools.com/what-is-uuid. [cit. 2023-12-07].

[13] *UUID Versions Explained: What are the different UUID versions and how to use them?*. Online. 2024 Dostupné z: https://www.uuidtools.com/uuid-versions-explained. [cit. 2024-05-13].

[14] OWASP Foundation *Cryptographic Storage Cheat Sheet*. Online. 2024 Dostupné z: https://cheatsheetseries.owasp.org/cheatsheets/Cryptographic_Storage_Cheat_Sheet.html. [cit. 2024-05-13].

[15] DYER, Bill. *Linux Jargon Buster: What is LUKS Encryption?*. Online. 2023. Dostupné z: https://itsfoss.com/luks/. [cit. 2024-05-06].

[16] SUSE LLC. *MDN Web Docs Glossary: Definitions of Web-related terms: UUID*. Online. 2022. Dostupné z: https://en.opensuse.org/SDB:Encrypted_filesystems?ref=itsfoss.com. [cit. 2024-05-06].

[17] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. *Hash Functions*. Online. 2023. Dostupné z: https://csrc.nist.gov/projects/hash-functions. [cit. 2024-05-07].

[18] NAKOV, Svetlin. *Hash Functions*. Online. 2020. Dostupné z: https://cryptobook.nakov.com/cryptographic-hash-functions. [cit. 2024-05-07].

[19] SCHLAWACK Hynek. *Hash Functions*. Online. 2015. Dostupné z: https://argon2-cffi.readthedocs.io/en/stable/argon2.html. [cit. 2024-05-07].

[20] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. *Argon2: the memory-hard function for password hashing and other applications*. Online. Lucembursko: University of Luxembourg, 2015. Dostupné z: https://www.password-hashing.net/argon2-specs.pdf. [cit. 2024-05-07].

[21] MALVIYA, Nitesh. *Fundamentals of symmetric and asymmetric cryptography*. Online. Infosec Institute, Inc. 2020. Dostupné z: https://www.infosecinstitute.com/resources/cryptography/fundamentals-of-symmetric-cryptography/. [cit. 2024-05-07].

[22] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. *Federal Information Processing Standards Publication 197, Advanced Encryption Standard (AES)*. Online. Vydáno 26. listopadu 2001, upraveno 9. května 2023. Dostupné z: https://nvlpubs.nist.gov/nistpubs/FIPS/NIST.FIPS.197-upd1.pdf. [cit. 2024-05-07].

[23] NATIONAL INSTITUTE OF STANDARDS AND TECHNOLOGY. *Block Cipher Techniques*. Online. Vydáno 2017, upraveno v únoru 2024. Dostupné z: https://csrc.nist.gov/Projects/block-cipher-techniques/BCM. [cit. 2024-05-07].

[24] RED HAT, INC. *A Detailed Look at the Boot Process*. Online. 2023. Dostupné z: https://access.redhat.com/documentation/en-us/red_hat_enterprise_linux/5/html/installation_guide/s1-boot-init-shutdown-process. [cit. 2023-12-11].

[25] BOTH, David. *An introduction to the Linux boot and startup processes*. Online. 2023. Dostupné z: https://opensource.com/article/17/2/linux-boot-and-startup. [cit. 2023-12-11].

[26] RED HAT, INC. *Using systemd unit files to customize and optimize your system*. Online. 2023. Dostupné z: https://access.redhat.com/documentation/enus/red_hat_enterprise_linux/8/html-single/using_systemd_unit_files_to_customize_and_optimize_your_system/index. [cit. 2023-12-11].

[27] SCHRODER, Carla. *Understanding and Using Systemd*. Online. 2014. Dostupné z: https://www.linux.com/training-tutorials/understanding-and-using-systemd/. [cit. 2023-12-11].

[28] PYTHON SOFTWARE FOUNDATION. *The Python Tutorial*. Online. 2023. Dostupné z: https://docs.python.org/3/tutorial/index.html. [cit. 2023-12-11].

[29] PYTHON SOFTWARE FOUNDATION. *subprocess — Subprocess management*. Online. 2023. Dostupné z: https://docs.python.org/3/library/subprocess.html. [cit. 2023-12-11].

[30] PYTHON SOFTWARE FOUNDATION. *os — Miscellaneous operating system interfaces*. Online. 2023. Dostupné z: https://docs.python.org/3/library/os.html. [cit. 2023-12-11].

[31] PYTHON SOFTWARE FOUNDATION. *re — Regular expression operations*. Online. 2024. Dostupné z: https://docs.python.org/3/library/re.html. [cit. 2024-05-07].

[32] PYTHON SOFTWARE FOUNDATION. *base64 — Base16, Base32, Base64, Base85 Data Encodings*. Online. 2024. Dostupné z: https://docs.python.org/3/library/base64.html. [cit. 2024-05-07].

[33] PYTHON SOFTWARE FOUNDATION. *datetime — Basic date and time types*. Online. 2024. Dostupné z: `https://docs.python.org/3/library/datetime.html`. [cit. 2024-05-07].

[34] PYTHON SOFTWARE FOUNDATION. *tkinter: Python interface to Tcl/Tk*. Online. 2023. Dostupné z: `https://docs.python.org/3/library/tkinter.html`. [cit. 2023-12-11].

[35] *PyCryptodome*. Online. 2023. Dostupné z: `https://www.pycryptodome.org/src/introduction`. [cit. 2024-05-07].

[36] KOVACS, Geza. *UNetbootin*. Online. 2023. Dostupné z: `https://unetbootin.github.io/#features`. [cit. 2024-05-07].

[37] ZEUTHEN, David a KATZ, Jeremy. *The Fedora Live CD Tools*. Online. 2023. Dostupné z: `https://github.com/livecd-tools/livecd-tools`. [cit. 2023-12-11].

Seznam symbolů a zkratek

UUID	Universal Unique Identifier
MAC	Media Access Control
GNU	GNU is not Unix
USB	Universal Serial Bus
GUID	Globally Unique Identifier
RFC	Request for Comment
BIOS	Basic Input/Output System
CD-ROM	Compact Disc Read-Only Memory
EFI	Extensible Firmware Interface
MBR	Master Boot Record
GRUB	Grand Unified Bootloader
GUI	Graphical User Interface
RHEL	Red Hat Linux Distribution
RAM	Random Access Memory
LUKS	Linux Unified Key Setup