Bitlocker šifrování disku v Linuxovém prostředí

Bc. Vojtěch Trefný

Diplomová práce
2019

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

*** Nascanované zadání, strana 1 ***

*** Nascanované zadání, strana 2 ***

Prohlašuji, že

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

Prohlašuji,

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

Ve Zlíně .....

podpis autora

ABSTRAKT

Text abstraktu česky

Klíčová slova: Přehled klíčových slov

ABSTRACT

Text of the abstract

Keywords: Some keywords

Zde je místo pro případné poděkování, motto, úryvky knih, básní atp.

OBSAH

ÚVOD

První odstavec pod nadpisem se neodsazuje, ostatní ano (pouze první řádek, odsazení vertikální mezy odstavci je typycké pro anglickou sazbu; czech babel toto respektuje, netřeba do textu přidávat jakékoliv explicitní formátování, viz ukázka sazby tohoto textu s následujícím odstavcem).

Formátování druhého odstavce. Text text text text text text text text text text text text.

I. TEORETICKÁ ČÁST

1 Šifrování disku

2 BitLocker

text

2.1 Použité kryptografické funkce

2.1.1 AES-CBC

2.1.2 Elephant difuzér

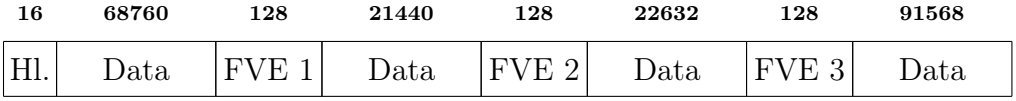
2.1.3 AES-XTS

2.1.4 AES-CCM

2.1.5 Odvození klíče z hesla

2.2 Diskový formát

Pro samotnou práci s BitLocker zařízením v linuxovém prostředí je nejdůležitější formát, tedy způsob, jakým jsou na disku uložena data. Protože je pomocí BitLockeru možné vytvořit šifrovaný flash disk, který lze použít na jiném počítači pouze za znalosti hesla, je zřejmé, že někde na samotném disku jsou uložena všechna potřebná metadata pro jeho „odemčení“[1)] v (alespoň částečně) otevřené podobě[2)].



Obr. 2.1 Zjednodušená struktura BitLocker zařízení

Na obrázku 2.1 je nastíněna zjednodušená struktura uložení dat a metadat na zařízení šifrovaném pomocí BitLockeru. Na začátku zařízení se nachází 8 KiB velká hlavička (podrobněji popsána v části 2.2.1) obsahující základní data pro jeho identifikaci a mezi zašifrovanými daty jsou uloženy tři kopie dalších metadat, každá o velikosti 64 KiB[3)] (podrobněji popsány v části 2.2.2). Čísla nad jednotlivými „částmi“ schématu odpovídají jejich velikosti v sektorech pro testovací zařízení o velikosti 100 MiB, které bylo vytvořeno ve Windows 10.

[1)] U šifrovaných úložných zařízení se běžně používá termín *odemčení* pro jeho „připravení“ pro čtení. Odemčení dává větší smysl, než dešifrování, protože data se dešifrují až při jejich čtení (abychom se vyhnuli relativně pomalému dešifrování dat, která nebudou čtena). Při odemčení se tedy pouze z metadat (nebo z jiného hardwaru, jako například TPM) získá (de)šifrovací klíč a připraví se (virtuální) zařízení, ze kterého lze číst data v otevřené podobě.

[2)] Ač se to může zdát, není to tak zcela samozřejmé. Populární nástroj VeraCrypt na zašifrovaném disku žádná metadata v otevřené podobě nemá.[4]

[3)] Velikosti odpovídají místu, které je pro daná metadata na zařízení vyhrazeno. Ve skutečnosti mohou být metadata mnohem menší.

2.2.1 Hlavička

Stejně jako u většiny diskových formátů, je i u BitLockeru na začátku disku takzvaná hlavička, která obsahuje základní informace o použitém formátu a jeho vlastnostech a také slouží k jeho rychlé identifikaci.

BitLocker hlavička zabírá celkem 512 bajtů a je u ní patrná inspirace u souborového systému NTFS. V tabulce 2.1 jsou zobrazeny jednotlivé (známé[4]) položky hlavičky BitLockeru a pro srovnání také stejné položky v hlavičce souborového systému NTFS.

Struktura NTFS hlavičky je převzata z [6], struktura BitLocker hlavičky je pak částečně převzata z [10], částečně z [8] a částečně výsledkem vlastního zkoumání.

Tab. 2.1 Porování položek hlaviček BitLocker a NTFS

offset	velikost	BitLocker	NTFS
0	3	boot kód	
3	8	OEM název (signatura)	
11	2	počet bajtů na sektor	
13	1	počet sektorů na cluster	
14	2	rezervované sektory	
16	4	nepoužito	
21	1	popisek média	
22	18	nepoužito	
40	8	počet sektorů	
48	8	adresa prvního clusteru MFT	
56	8	kopie adresy prvního clusteru MFT	
64	1	velikost MFT entry	
65	3	nepoužito	
68	1	velikost indexu	
69	3	nepoužito	
72	8	NTFS serial number	
80	4	nepoužito	
84	76	boot kód	
160	16	BitLocker GUID	boot kód
176	8	offset první kopie FVE metadat	
184	8	offset druhé kopie FVE metadat	
192	8	offset třetí kopie FVE metadat	
200	310	boot kód	
510	2	signatura (`0xaa55`)	

Z pohledu identifikace BitLocker zařízení je nejdůležitější částí hlavičky 8 bajtů na offsetu 3, které se u NTFS formátu nazývají *OEM název* a které slouží pro rychlou

[4] Struktura formátu BitLocker není společností Microsoft nikde veřejně zcela kompletně zdokumentována, význam jednotlivých položek tedy nemusí být vždy přesně znám.

```
00000000  eb 58 90 2d 46 56 45 2d  46 53 2d 00 02 08 00 00 |.X.-FVE-FS-.....|
00000010  00 00 00 00 00 f8 00 00  3f 00 ff 00 00 28 03 00 |........?....(..|
00000020  00 00 00 00 e0 1f 00 00  00 00 00 00 00 00 00 00 |................|
00000030  01 00 06 00 00 00 00 00  00 00 00 00 00 00 00 00 |................|
00000040  80 00 29 00 00 00 00 4e  4f 20 4e 41 4d 45 20 20 |..)....NO NAME  |
00000050  20 20 46 41 54 33 32 20  20 20 33 c9 8e d1 bc f4 |  FAT32   3.....|
00000060  7b 8e c1 8e d9 bd 00 7c  a0 fb 7d b4 7d 8b f0 ac |{......|..}.}...|
00000070  98 40 74 0c 48 74 0e b4  0e bb 07 00 cd 10 eb ef |.@t.Ht..........|
00000080  a0 fd 7d eb e6 cd 16 cd  19 00 00 00 00 00 00 00 |..}.............|
00000090  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 |................|
000000a0  3b d6 67 49 29 2e d8 4a  83 99 f6 a3 39 e3 d0 01 |;.gI)..J....9...|
000000b0  00 50 19 02 00 00 00 00  00 d0 c1 02 00 00 00 00 |.P..............|
000000c0  00 a0 73 03 00 00 00 00  00 00 00 00 00 00 00 00 |..s.............|
000000d0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 |................|
*
00000100  0d 0a 52 65 6d 6f 76 65  20 64 69 73 6b 73 20 6f |..Remove disks o|
00000110  72 20 6f 74 68 65 72 20  6d 65 64 69 61 2e ff 0d |r other media...|
00000120  0a 44 69 73 6b 20 65 72  72 6f 72 ff 0d 0a 50 72 |.Disk error...Pr|
00000130  65 73 73 20 61 6e 79 20  6b 65 79 20 74 6f 20 72 |ess any key to r|
00000140  65 73 74 61 72 74 0d 0a  00 00 00 00 00 00 00 00 |estart..........|
00000150  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00 |................|
*
00000190  00 00 00 00 00 00 00 00  78 78 78 78 78 78 78 78 |........xxxxxxxx|
000001a0  78 78 78 78 78 78 78 78  78 78 78 78 78 78 78 78 |xxxxxxxxxxxxxxxx|
*
000001e0  78 78 78 78 78 78 78 78  ff ff ff ff ff ff ff ff |xxxxxxxx........|
000001f0  ff ff ff ff ff ff ff ff  ff ff ff 00 1f 2c 55 aa |.............,U.|
00000200
```

Obr. 2.2 BitLocker hlavička se zvýrazněnou signaturou, GUID a trojicí offsetů FVE metadat

identifikace zařízení. V linuxových systémech se podobné identifikátory obvykle nazývají *signatura*. Pro BitLocker formát je (u všech verzí) signatura v ASCII podobě `-FVE-FS-`.

TODO: To by asi chtělo citaci.

Pro další práci s BitLockerem není většina položek hlavičky zajímavá. Výjimku tvoří GUID identifikátor uložený na offsetu 160 (16 bajtů dlouhý UTF-8 textový řetězec) a trojice 32bitových bezznaménkových celočíselných (`uint32`) hodnot na offsetech 176, 184 a 192, které obsahují umístění (jako relativní offset od začátku zkoumaného zařízení) tří bloků FVE metadat. Všechny tyto čtyři hodnoty jsou v BitLocker hlavičce umístěny na offsetech, které jsou v NTFS součástí *bootcode*.

Umístění všech výše zmíněných „důležitých“ částí BitLocker hlavičky je zobrazeno na obrázku 2.2.

2.2.2 FVE metadata

Samotná výše popsaná hlavička formátu BitLocker neobsahuje o samotném BitLockeru téměř žádné informace. Slouží především pro rychlou identifikaci zařízení jako zařízení šifrovaného pomocí technologie BitLocker. Všechny informace potřebné pro práci s tímto zařízením, tedy především způsob uložení dat, jejich umístění, způsob jakým jsou šifrována a hlavně klíč pro jejich (de)šifrování, jsou uloženy na třech různých místech[5)] definovaných v hlavičce. Jedná se o tři identické kopie[6)] takzvaných *FVE metadat*.

FVE metadata se skládají z celkem tří částí – hlavičky FVE bloku (*FVE metadata block header*), samotné FVE hlavičky (*FVE metadata header*) a různého množství FVE záznamů (*FVE metadata entry*, které obsahují samotné klíče a další důležité informace[10][7)].

Důležité položky v obou hlavičkách, jejich velikosti a offsety (vztažené vůči začátku dané hlavičky) jsou uvedeny v tabulce 2.2. Kompletní struktura obou hlaviček je součástí přílohy .

TODO: odkaz na přílohu

Tab. 2.2 Zjednodušená struktura FVE metadat

Hlavička FVE bloku		
offset	velikost	popis
0	8	signatura (`-FVE-FS-`)
10	2	verze (1 nebo 2)
32	8	offset první kopie FVE metadat
40	8	offset druhé kopie FVE metadat
48	8	offset třetí kopie FVE metadat

FVE hlavička		
0	4	velikost metadat (včetně záznamů)
16	16	GUID
36	4	šifrovací algoritmus
40	8	datum a čas vytvoření

Mezi pro nás zajímavé položky v hlavičce patří její celková velikost (včetně velikosti samotné hlavičky a velikosti za ní následujících záznamů), šifrovací algoritmus použitý pro zašifrování dat uložených na disku (možné algoritmy jsou popsány v části 2.1) a

[5)] Na offsetech přibližně ve 33 %, 44 % a 55 % u testovaných BitLocker zařízení.

[6)] Tři kopie jsou zvoleny pravděpodobně jako záloha pro případ náhodného poškození metadat. Vzhledem k tomu, že bez kompletní nepoškozené kopie těchto metadat není možné data na zařízení dešifrovat, je vícenásobná záloha na místě.

[7)] Toto dělení zavádí Joachim Metz v [10]. Teoreticky by se daly dvě první části metadat spojit, protože na disku se nachází vždy hned za sebou, ale rozdělení dává smysl, protože první část se týká popisu samotných metadat (signatura, verze, umístění všech tří bloků), zatímco druhá část už obsahuje samotná metadata (GUID, čas vytvoření, použitý šifrovací algoritmus).

v některých případech může být užitečný i čas vytvoření, který je uložen ve formátu `FILETIME`[8)].

2.2.3 FVE záznamy

Za výše uvedenou hlavičkou se nachází blíže nespecifikované množství FVE záznamů. Ty slouží v podstatě jako key-value úložiště pro jakékoli další „informace“, které jsou pro práci s BitLockerem potřebné. Tím, že není třeba předem určeno, kolik takových záznamů bude za hlavičkou uloženo, je možné přidávat nové položky při zachování zpětné kompatibility[9)].

Jelikož známe celkovou velikost FVE metadat (je uvedena v hlavičce, viz tabulka 2.2) a celková velikosti hlaviček FVE metadat je pevná (64 a 48 bajtů), pro přečtení všech záznamů stačí číst data ve smyčce, dokud nedojdeme na konec metadat, nebo dokud následující záznam nemá nulovou velikost.

Struktura FVE je relativně jednoduchá a je popsaná v tabulce 2.3. Důležitou součástí je velikost záznamu, protože podle svého typu může mít různou délku.

Tab. 2.3 Struktura FVE záznamu

offset	velikost	popis
0	2	velikost záznamu
2	2	typ záznamu
4	2	typ hodnoty záznamu
6	2	verze (1)
8		data

Typ a hodnota označují, co je v daném záznamu uloženo. Známé typy a hodnoty jsou popsány v tabulce 2.4. U typů se typicky jedná buď o klíč (FVEK, VMK), nebo obecnou *property*, hodnota pak dále specifikuje, jak je daný typ uložen (zašifrovaný klíč, textový řetězec).

Způsob uložení dat záleží na tom, jaká konkrétní data jsou v záznamu uložena. U „jednoduchých“ záznamů, jako je například popisek, je v datech uložen textový řetězec uložený v kódování UTF-16, u „složitějších“ záznamů, jako jsou například klíče, mají data vlastní strukturu včetně dalších záznamů.

[8)] FILETIME je ve skutečnosti struktura sestávající ze dvou 32bit celočíselných hodnot, které dohromady udávají počet 100 nanosekundových intervalů, které k danému datu uplynuly od 1. ledna 1601.[2]

[9)] Celková největší možná velikost FVE metadat je 64 KiB (alespoň tedy tolik je pro FVE metadata vyhrazeno na vytvořených BitLocker zařízeních), teoreticky je tedy možné mít až 64 KiB - 112 B metadat.

[10)] Umístění a velikost NTFS hlavičky otevřeného zařízení. Odpovídá hodnotě 15. Podrobnější informace o umístění NTFS hlavičky na šifrovaném zařízení jsou v části 2.4.1.

Tab. 2.4 Známé typy FVE záznamů

Typy	
typ	popis
0	property
1	VMK
2	FVEK
7	popisek
15	hlavička disku[10)]

Hodnoty	
typ	popis
0	smazáno
1	klíč
2	string
5	AES-CCM šifrovaný klíč
6	TPM klíč
8	VMK
15	offset a velikost

Příklad „jednoduchého“ záznamu je uveden na obrázku 2.3, kde vidíme záznam typu *description* (popisek). Ten v podstatě obsahuje jméno počítače, na kterém bylo dané BitLocker zařízení vytvořeno a také datum vytvoření. Můžeme tedy vidět, že toto konkrétní BitLocker zařízení bylo vytvořeno na počítači `DESKTOP-NPM7RCA` a to 3. února 2019. Tato informace je uloženo jako standardní textový řetězec v kódování UTF-16. Kromě tohoto řetězce jsou pak na obrázku zvýrazněny i další údaje: velikost celého záznamu (64 bajtů), jeho typ (7 — popisek) a hodnota (2 — textový řetězec) a verze (1).

```
02195070  40 00 07 00 02 00 01 00  44 00 45 00 53 00 4b 00 |@.......D.E.S.K.|
02195080  54 00 4f 00 50 00 2d 00  4e 00 50 00 4d 00 37 00 |T.O.P.-.N.P.M.7.|
02195090  52 00 43 00 41 00 20 00  47 00 3a 00 20 00 32 00 |R.C.A. .G.:. .2.|
021950a0  2f 00 33 00 2f 00 32 00  30 00 31 00 39 00 00 00 |/.3./.2.0.1.9...|
```

Obr. 2.3 Příklad FVE záznamu typu „description“ (popisek)

U jednoduchého zařízení — v tomto konkrétním případě USB flash disku — se bude obvykle vyskytovat pouze pět záznamů a to již výše zmíněný popisek, dvojice záznamů typu VMK, jeden záznam typu FVEK (více informací o obou se nachází v části 2.3) a jeden záznam obsahující informace o umístění hlavičky disku (více informací o tomto záznamu se nachází v části 2.4.1).

2.3 Klíče

Pravděpodobně nejdůležitější součástí BitLocker hlavičky jsou šifrovací klíče. Ve FVE metadatech nalezneme celkem dva typy klíčů — Full Volume Encryption Key, neboli FVEK, a Volume Master Key, neboli VMK[11)]. Uloženy jsou v metadatových záznamech odpovídajících typů a to samozřejmě nikoli v otevřené podobě, ale zašifrované.

[11)] Původní varianta BitLockeru má ještě jeden klíč — TWEAK, ten je podrobněji popsán v části 2.5.

2.3.1 Full Volume Encryption Key

Full Volume Encryption Key (dále jen „FVEK“) je nejdůležitějším klíčem pro celý BitLocker. Pomocí tohoto klíče jsou totiž zašifrovaná data uložená na disku. FVEK samotný nejde změnit[12)] a v případě jeho poškození nebo náhodného smazání, není možné uložená data nijak dešifrovat.

FVEK je v metadatech uložen v záznamu typu *FVEK* s hodnotou *AES-CCM šifrovaný klíč* a je, jak hodnota naznačuje, zašifrován pomocí šifry AES-CCM (o této šifře a módu více v části 2.1), kdy je jako klíč použit VMK a jako inicializační vektor 0.

Tab. 2.5 Způsob uložení FVEK v metadatech

offset	velikost	popis
0	8	datum a čas vytvoření (jako FILETIME)
8	4	nonce
12	16	MAC tag
28	44[13)]	šifrovaný klíč

Struktura dat pro FVEK v metadatovém záznamu je popsána v tabulce 2.5. Kromě samotného klíče obsahují datum a čas jeho vytvoření a nonce.

TODO: najít definici a citaci

Samotná zašifrovaná část klíče obsahuje kromě samotného klíče také další data o klíči samotném — velikost, verze a šifrovací metoda použitá pro data zašifrovaná pomocí FVEK. Jejich struktura je popsána v tabulce 2.6.

Tab. 2.6 Obsah FVEK po dešifrování

offset	velikost	popis
0	4	velikost
4	4	verze (1)[14)]
8	4	šifrovací metoda
12	32	klíč

Na obrázku 2.4 je pak vidět příklad dešifrovaného FVEK. Zvýrazněny jsou jeho celková velikost (44 bajtů), verze (1), použitá šifrovací funkce (hexadecimální kód `0x8004` v tomto případě znamená 128bit AES-XTS) a následně samotný 128bit klíč.

[12)] Bez kompletního přešifrování všech dat.

[13)] Velikost šifrovaného klíče záleží na použité šifře — 12 bajtů vždy připadne na informace o klíči a 32 bajtů v tomto případě připadá na samotný klíč, jelikož je použit 128bit AES.

[14)] Některé zdroje [10] uvádějí verzi pouze jako 2 bajtovou a následující 2 bajty jako „neznámé“. Vzhledem k tomu, že v jiných hlavičkách je verze v některých případech 4 bajtová a v některých 2 bajtová a že na testovacích zařízeních byly tyto dva bajty vždy nulové, domnívám se, že je pravděpodobnější, že verze je zde 4 bajtová.

```
00000000  2c 00 00 00 01 00 00 00  04 80 00 00 a4 d0 11 64 |,..............d|
00000010  0c a0 df ec b2 4d a2 39  b1 4e 4a b7 62 56 f2 e3 |.....M.9.NJ.bV..|
00000020  b2 27 54 40 91 21 0e 98  aa 84 5f 52              |.'T@.!...._R|
```

Obr. 2.4 Dešifrovaný FVEK

2.3.2 Volume Master Key

Jak již bylo řečeno výše, FVEK je na disku uložen zašifrován pomocí Volume Master Key (dále jen „VMK“). Ten je uložen také v metadatových záznamech ve FVE metadatech a je také zašifrován. Na rozdíl od FVEK, který je vždy uložen v pouze v jediné kopii, VMK může být ve FVE metadatech uložen vícekrát, pokaždé chráněný jiným způsobem, tedy pokaždé „jinak“ zašifrovaný.

Tento systém umožňuje, aby byl FVEK (jakožto „hlavní“ a nejdůležitější klíč) uložen na disku pouze v jediné kopii, ale zároveň existovala možnost, jak mít pro jedno zařízení více různých hesel (respektive více různých způsobů odemčení daného zařízení). Pro přidání „nového“ hesla tak teoreticky stačí jednoduše znát alespoň jedno již existující, pomocí kterého se VMK dešifruje a následně uloží zašifrovaný pomocí nového hesla. Analogicky tak lze také snadno změnit heslo — jak již bylo zmíněno výše, FVEK nejde změnit bez přešifrování celého zařízení, ale změna hesla díky tomuto systému znamená pouhé uložení nově zašifrovaného VMK.

V odstavci výše je několikrát zmíněno *heslo*, ale VMK může být chráněn více různými způsoby. Dokumentace BitLockeru [3] zmiňuje celkem deset možných typů *protektorů* klíčů (tedy deset způsobů, jak může být daný klíč chráněn, respektive šifrován) v BitLockeru. Tyto možnosti jsou zapsány v tabulce 2.7.

Tab. 2.7 Možnosti ochrany VMK

hodnota	popis
0	neznámý/jiný
1	TPM
2	externí klíč
3	číselné heslo
4	TPM a PIN
5	TPM klíč
6	TPM, PIN a klíč
7	veřejný klíč
8	heslo
9	TPM certifikát
10	CryptoAPI Next Generation (CNG)

Z pohledu této práce je nejobvyklejším protektorem právě heslo, protože použití

BitLocker zařízení v linuxovém prostředí se dá předpokládat primárně u flash disků, u nichž se používá ochrana heslem[15)].

Pro každé vytvořené BitLocker se kromě „primární“ ochrany (v našem případě typicky hesla) vytváří ještě jeden VMK chráněný takzvaným záložním heslem. Způsob ochrany je u něj stejný jako u VMK, který je chráněný heslem, rozdíl je v tom, že heslo zadává uživatel, kdežto záložní heslo je vygenerované a uživateli je při vytváření „předáno“ v podobě souboru, který obsahuje 48 čísel. U něj se předpokládá, že si jej uživatel buď vytiskne nebo bezpečně uloží v elektronické podobě. U strojů přihlášených v síti Active Directory, lze také záložní klíče automaticky zálohovat na doménovém serveru. Pomocí záložního hesla lze pak zařízení odemknout stejně, jako při použití „normálního“ hesla a pomocí nástrojů obsažených v základní instalaci Windows nastavit nové heslo (nebo nastavit nové TPM, či jiný způsob ochrany).[9]

Struktura VMK, naznačená v tabulce 2.8, je ovlivněna tím, že samotný klíč může být chráněn různými způsoby a je pro něj tedy třeba ukládat různá metadata, a i samotný klíč může být potřeba v některých případech ukládat v různých podobách.

První část VMK struktury je v celku běžná — obsahuje identifikátor klíče (GUID), čas vytvoření a typ ochrany. Další metadata jsou pak uložena jako záznamy, stejně jako u samotné FVE hlavičky (podrobněji v části 2.2.3 a tabulce 2.3).

Tab. 2.8 Struktura VMK

offset	velikost	popis
0	16	GUID
16	9	datum a čas vytvoření
24	2	neznámé
26	2	typ ochrany
28		metadatové záznamy

Kompletní VMK klíč chráněný záložním heslem je zobrazen na obrázku 2.5. Zvýrazněno je GUID, typ ochrany (8 — heslo) a dva „připojené“ záznamy, oba typu property, první obsahující sůl potřebnou pro odvození klíče potřebného pro dešifrování VMK ze záložního hesla (funkcionalita odvození klíče z hesla je popsána v části) a druhá obsahující samotný klíč (textový výpis je debugovacím výstupem z nástroje vytvořeného v rámci praktické části).

TODO: odkaz na část popisující KDF

[15)] Ochrana pomocí TPM nedává u přenosných disků smysl, protože TPM čipy jsou nedělitelnou součástí hardwaru a takto chráněný disk by nešlo na jiném počítači dešifrovat.

```
00000000 c1 56 2e 01 d6 4e 27 45  8a bf 7a 9f 29 e0 b5 21  |.V...N'E..z.)..!|
00000010 40 c5 cd 54 a0 bb d4 01  00 00 00 08 ac 00 00 00  |@..T............|
00000020 03 00 01 00 00 10 00 00  46 ee b7 10 0e 43 4d d4  |........F....CM.|
00000030 f1 84 a5 ab eb c6 21 f4  40 00 12 00 05 00 01 00  |....!...@.......|
00000040 40 7d e5 52 a0 bb d4 01  04 00 00 00 72 b0 71 f4  |@}.R........r.q.|
00000050 20 9e c9 8e b7 1b 5e 42  71 b5 bc 21 c6 57 9b 29  | .....^Bq..!.W.)|
00000060 56 2c 92 ad db d7 73 75  a9 78 c2 94 c5 a5 07 d1  |V,....su.x......|
00000070 62 61 0c 56 d8 ca 9d ac  50 00 13 00 05 00 01 00  |ba.V....P.......|
00000080 40 7d e5 52 a0 bb d4 01  05 00 00 00 3e d9 ac 58  |@}.R........>..X|
00000090 e6 86 ba ac 05 48 ea 0b  64 ee 77 7a b4 77 ba cb  |.....H..d.wz.w..|
000000a0 c0 83 83 b0 7b ab 52 c7  0d 9e 8f 62 d7 cb a3 90  |....{.R....b....|
000000b0 cc b8 8e 39 a4 be 8a 0a  5c 16 86 62 c9 64 81 4d  |...9....\..b.d.M|
000000c0 91 9d 27 24 3a 8e a3 7c  50 00 00 00 05 00 01 00  |..'$:..|P.......|
000000d0 40 7d e5 52 a0 bb d4 01  06 00 00 00 97 18 2f d6  |@}.R........../.|
000000e0 83 de e7 63 0a fa 57 48  44 2b 66 90 91 a0 ad e9  |...c..WHD+f.....|
000000f0 0c 08 e8 1e 3d 2f 7d 3b  cc 9f ba e4 ed b5 6b c2  |....=/};......k.|
00000100 e1 a4 53 cf c5 60 2a 92  2d c8 1d 85 10 b7 99 87  |..S..`*.-.......|
00000110 9d 1d 1e 36 46 40 6b e7                          |...6F@k.        |

VMK
        Identifier:     012e56c1-4ed6-4527-8abf-7a9f29e0b521
        Type:           VMK protected with recovery password
        Salt:           46 ee b7 10 0e 43 4d d4 a5 ab eb c6 21 f4 f1 84
        AES-CCM encrypted key
                Nonce data:     2019-02-03 09:10:36.052000
                Nonce counter:  6
                Key:     91 a0 ad e9 0c 08 ... 1d 1e 36 46 40 6b e7
```

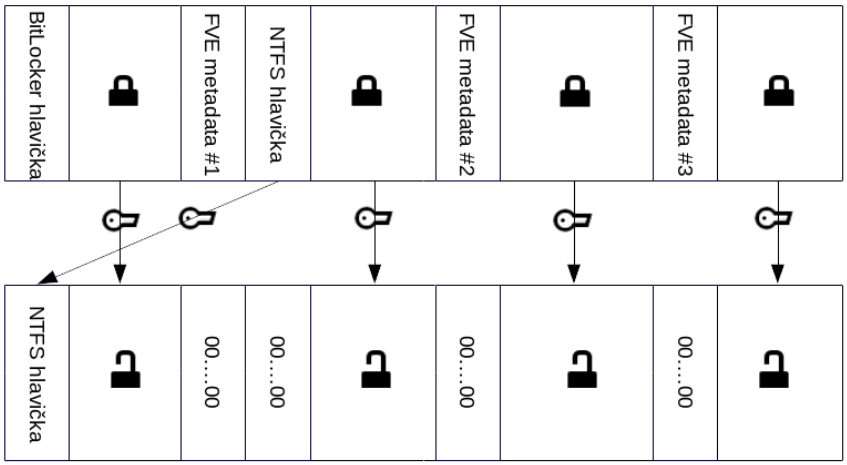
Obr. 2.5 VMK chráněný záložním heslem

2.4 Šifrovaná data

2.4.1 Způsob uložení data

Po hlavičkách a metadatech zbývá popsat jen způsob, jakým jsou na disku uložena samotná šifrovaná data. Protože BitLocker metadata se vyskytují celkem ve třech kopiích na různých místech „uprostřed“ šifrovaného zařízení, jsou uložená data rozdělena celkem na čtyři části. Až na jednu výjimku jsou šifrovaná data uložena na „správných“ místech — tedy na místě, kde mají být uložena i po dešifrování.

Výjimkou je v tomto případě hlavička souborového systému NTFS. Její „nesprávné“ umístění je způsobeno poněkud zvláštním rozhodnutím tvůrců BitLockeru, že otevřené zařízení bude mít stejnou velikost, jako zařízení zašifrované, a to i přesto, že si z jeho celkové velikosti BitLocker metadata uberou přibližně 200 KiB[16)].



Obr. 2.6 Schéma „mapování“ mezi šifrovaným a otevřeným BitLocker zařízením

Speciální zacházení vyžaduje NTFS hlavička proto, že na výsledném otevřeném zařízení musí být na jeho začátku, aby toto zařízení bylo systémem správně rozpoznáno jako NTFS a jako takové připojeno. Proto je třeba NTFS hlavičku přesunout na začátek disku a nahradit jí původní BitLocker hlavičku. Umístění NTFS hlavičky v šifrovaných datech je zapsáno ve FVE metadatech ve speciálním záznamu typu *hlavička disku* (viz tabulka 2.4). V záznamu je uveden offset (relativně k začátku disku), na kterém se zašifrovaná NTFS hlavička nachází a její velikost (u testovaných zařízení 8 KiB, což také odpovídá velikosti vyhrazené pro BitLocker hlavičku).

Vzhledem k tomu, že výsledné otevřené zařízení má mít stejnou velikosti, jako šifrované zařízení, zbývá ještě vyřešit, jak bude v otevřeném zařízení naloženo s metadaty — na jejich místě v otevřeném NTFS musí „něco“ být a zároveň je třeba ochránit je proti náhodnému přepsání nebo smazání. Teoreticky je možné tato metadata prostě dešifrovat stejně jako ostatní šifrovaná data. Výsledkem by pak sice byla nesmyslná data, ale pokud je zařízení již otevřeno, není třeba k metadatům již znovu přistupovat a je tedy jedno, že nejsou „čitelná“. Takováto „nesmyslná“ data by už jen stačilo v rámci NTFS ochránit před přepsáním. Pokud by bylo třeba k metadatům přistupovat i u otevřeného zařízení, bylo by další možností nechat je prostě viditelná tak, jak jsou (a opět je ochránit před přepsání).

Autoři BitLockeru ale nakonec sáhli po třetí možnosti — metadata jsou v otevřeném zařízení nahrazena nulami. Ve výsledném otevřeném NTFS tak jsou metadata viditelná jako speciální systémové soubory uložené ve složce `System Volume Information`. Sou-

[16] U linuxové implementace šifrování disku, technologie LUKS/dm-crypt, byl zvolen jiný přístup — otevřené zařízení je menší a metadata se na něm nijak neřeší — jsou z výsledného zařízení „odstraněna“. První sektor otevřeného zařízení pak obsahuje standardní hlavičku souborového systému bez potřeby dalšího „přesouvání“ jako u BitLockeru.

bory jsou samozřejmě prázdné, respektive plné nul, ale zabraňují přepsání míst, na kterých se skutečná metadata vyskytují. Ve Windows je tato složka ve výchozím nastavení skryta.

2.4.2 Postup při dešifrování

Při znalosti struktury metadat a způsobu uložení šifrovaných dat, je dešifrování již celkem jednoduchou záležitostí — z FVE hlavičky (tabulka 2.2) zjistíme, jaký byl použit šifrovací algoritmus (v nejnovějších verzích BitLockeru to bude AES-XTS), pomocí uživatelem zadaného hesla dešifrujeme VMK s odpovídajícícm typem ochrany (tabulka 2.7) a pomocí něj dešifrujeme FVEK, kterým jsou zašifrována samotná data.

```
00000000  eb 52 90 4e 54 46 53 20  20 20 20 00 02 08 00 00  |.R.NTFS    .....|
00000010  00 00 00 00 00 f8 00 00  3f 00 ff 00 00 28 03 00  |........?....(..|
00000020  00 00 00 00 80 00 00 00  ff 1f 03 00 00 00 00 00  |................|
00000030  55 21 00 00 00 00 00 00  02 00 00 00 00 00 00 00  |U!..............|
00000040  f6 00 00 00 01 00 00 00  52 53 3d 84 7d 3d 84 a4  |........RS=.}=..|
00000050  00 00 00 00 fa 33 c0 8e  d0 bc 00 7c fb 68 c0 07  |.....3.....|.h..|
00000060  1f 1e 68 66 00 cb 88 16  0e 00 66 81 3e 03 00 4e  |..hf......f.>..N|
00000070  54 46 53 75 15 b4 41 bb  aa 55 cd 13 72 0c 81 fb  |TFSu..A..U..r...|
00000080  55 aa 75 06 f7 c1 01 00  75 03 e9 dd 00 1e 83 ec  |U.u.....u.......|
00000090  18 68 1a 00 b4 48 8a 16  0e 00 8b f4 16 1f cd 13  |.h...H..........|
000000a0  9f 83 c4 18 9e 58 1f 72  e1 3b 06 0b 00 75 db a3  |.....X.r.;...u..|
000000b0  0f 00 c1 2e 0f 00 04 1e  5a 33 db b9 00 20 2b c8  |........Z3... +.|
000000c0  66 ff 06 11 00 03 16 0f  00 8e c2 ff 06 16 00 e8  |f...............|
000000d0  4b 00 2b c8 77 ef b8 00  bb cd 1a 66 23 c0 75 2d  |K.+.w......f#.u-|
000000e0  66 81 fb 54 43 50 41 75  24 81 f9 02 01 72 1e 16  |f..TCPAu$....r..|
000000f0  68 07 bb 16 68 52 11 16  68 09 00 66 53 66 53 66  |h...hR..h..fSfSf|
00000100  55 16 16 16 68 b8 01 66  61 0e 07 cd 1a 33 c0 bf  |U...h..fa....3..|
00000110  0a 13 b9 f6 0c fc f3 aa  e9 fe 01 90 90 66 60 1e  |.............f‘.|
...
00000150  0f 82 16 00 66 ff 06 11  00 03 16 0f 00 8e c2 ff  |....f...........|
00000160  0e 16 00 75 bc 07 1f 66  61 c3 a1 f6 01 e8 09 00  |...u...fa.......|
00000170  a1 fa 01 e8 03 00 f4 eb  fd 8b f0 ac 3c 00 74 09  |............<.t.|
00000180  b4 0e bb 07 00 cd 10 eb  f2 c3 0d 0a 41 20 64 69  |............A di|
00000190  73 6b 20 72 65 61 64 20  65 72 72 6f 72 20 6f 63  |sk read error oc|
000001a0  63 75 72 72 65 64 00 0d  0a 42 4f 4f 54 4d 47 52  |curred...BOOTMGR|
000001b0  20 69 73 20 63 6f 6d 70  72 65 73 73 65 64 00 0d  | is compressed..|
000001c0  0a 50 72 65 73 73 20 43  74 72 6c 2b 41 6c 74 2b  |.Press Ctrl+Alt+|
000001d0  44 65 6c 20 74 6f 20 72  65 73 74 61 72 74 0d 0a  |Del to restart..|
000001e0  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00  |................|
000001f0  00 00 00 00 00 00 8a 01  a7 01 bf 01 00 00 55 aa  |..............U.|
00000200
```

Obr. 2.7 První sektor dešifrovaného BitLocker zařízení (NTFS hlavička)

Jedinou neznámou potřebnou pro dešifrování dat tak zůstává inicializační vektor. Ten je naštěstí u nejnovější verze BitLockeru velice jednoduchý a odpovídá offsetu (v sektorech), na kterém jsou daná šifrovaná data uložena na zašifrovaném zařízení (první sektor NTFS hlavičky, který bude v dešifrovaných datech uložen na začátku tak má inicializační vektor daný svou pozicí v šifrovaných datech, nikoli nulový, jak by se mohlo zdát).

Dešifrovaný první sektor je vidět na obrázku 2.7. Jde zde velmi dobře poznat podobnost NTFS hlavičky s BitLocker hlavičkou (obrázek 2.2). Hlavní odlišností je signatura, která je zde jasně viditelná jako `NTFS`, na rozdíl od `-FVE-FS-` u BitLockeru. Chybí také offsety BitLocker metadat, které u BitLocker hlavičky „zabírají“ část boot kódu, který je u NTFS kompletní. Zajímavá je stejná boot signatura (`55 aa`) u obou hlaviček.

U zbývajících dat pak dešifrování probíhá stejně — po 512 B sektorech s inicializačním vektorem nastaveným na číslo sektoru odpovídající jejich umístění na šifrovaném zařízení. Jedinou výjimkou jsou oblasti BitLocker metadata, která jsou nahrazena nulami, jak bylo popsáno v části 2.4.1.

2.5 Odlišnosti ve starších verzích

Výše popsaná struktura diskového formátu BitLocker, způsob uložení klíčů a rozložení dat na šifrovaném a dešifrovaném zařízení, odpovídají aktuální nejnovější verzi BitLockeru dostupné ve Windows 7 a novějších. Původní verze dostupná ve Windows Vista se v některých drobnostech mírně liší. Tato práce se primárně zaobírá nejnovější verzí, protože je v současné době jedinou podporovanou (oficiální podpora Windows Vista byla ukončena 11. dubna 2017[5]). Podpora pro starší verze však může být také v některých případech vyžadovaná, a proto si ve stručnosti představíme nejvýznamnější odlišnosti mezi těmito verzemi.

Nejvýraznější změnou je nejspíše změna algoritmu použitého pro šifrování data z AES-CBC na AES-XTS (rozdíl mezi těmito algoritmy a pravděpodobný důvod pro změnu je popsán v části 2.1), ale menší změny se týkají i samotných metadat a hlavičky.

2.5.1 Hlavička

Samotná BitLocker hlavička se změnila jen minimálně. Zajímavé je, že starší verze obsahuje „odkaz“ pouze na první kopii FVE metadat a to přesto, že i tato verze obsahuje tři kopie. Offsety ostatních kopií je tak třeba vyčíst ze samotných metadat. Díky tomu se všechna metadata specifická pro BitLocker „vešla“ do nevyužitých oblastí v NTFS hlavičce (od které je BitLocker hlavička odvozen, viz 2.2.1) a nezasahují tak do boot kódu. Na druhou stranu v případě poškození první kopie FVE metadat bude složitější najít na disku další dvě „záložní“ kopie.

2.5.2 FVE metadata

FVE metadata se u starší verze liší pouze v drobných detailech. Výhodou FVE metadat je, že jsou verzovaná a lze tak snadno rozpoznat, u jakou variantu BitLockeru se jedná. Starší varianta má verzi 1, novější varianta má verzi 2. Důležité hodnoty (signatura, velikost, umístěné offsetů všech tří kopií FVE metadat) jsou v obou verzích stejné.

2.5.3 Klíče

Struktura klíčů VMK a FVEK se u starší verze BitLockeru nijak neliší. Jediný rozdíl představuje „nový“ klíč TWEAK, který se používá pro šifrování inicializačního vektoru. TWEAK klíč je uložen, podobně jako FVEK, zašifrovaný pomocí VMK ve FVE metadatech jako speciální záznam (viz 2.2.3).

2.5.4 Šifrovaná data

Asi největší odlišnost u starších verzí je ve způsobu uložení šifrovaných dat a to především v umístění NTFS hlavičky otevřeného zařízení. Zatímco u novější verze BitLockeru je tato hlavička zašifrovaná a uložená na speciálním místě zapsaném ve FVE metadatech (způsob umístění šifrované NTFS hlavičky je popsán v části 2.4.1), v původní variantě BitLockeru je NTFS hlavička uložena nezašifrovaná a to přímo na svém „původním“ místě na začátku disku. Vzhledem k malé odlišnosti původní hlavičky BitLockeru a hlavičky NTFS stačí při dešifrování nahradit určité části BitLocker hlavičky a výsledkem je validní NTFS hlavička pro dešifrované zařízení.

Nahradit je třeba signaturu — místo původního `-FVE-FS-` dosadíme `NTFS` (standardní signatura souborového systému NTFS) a offset první kopie FVE metadat — ten nahradí adresa prvního clusteru MFT, která je uložen ve FVE metadatech.

Poslední rozdíl v šifrovaných datech spočívá v inicializačním vektoru použitém pro jejich (de)šifrování. Stejně jako u novější verze BitLockeru se zde použije číslo sektoru, ale nikoli „prosté“, ale zašifrované pomocí TWEAK klíče.

3 Existující řešení pro práci s BitLockerem v Linuxu

Pro Linux již v současné době existují nástroje, které umí s BitLockerem více či méně pracovat. Podle aktivity vývoje a pokrytí funkcionality BitLockeru jsou nejvýznamnější dva projekty — knihovna libbde[12] a nástroj Dislocker[7].

3.1 libbde

Knihovna libbde vytvořená Joachimem Metzem představuje asi nejlepší software pro práci s BitLockerem v linuxových systémech a nejen tam, protože podporuje i systémy Microsoft Windows a MacOS X[11]. Kromě knihovny jsou součástí projektu i nástroje pro koncové uživatele `bdemount` a `bdeinfo`. Ukázka výstupu nástroje `bdeinfo`, který slouží primárně pro analýzu existujících zařízení, je vidět na obrázku 3.1. Užitečná může být také dostupnost rozhraní pro jazyk Python (samotná knihovna je implementována v jazyce C).

```
BitLocker Drive Encryption information:
        Encryption method               : AES-XTS 128-bit
        Volume identifier               : 1f8bf933-8323-4c97-8a89-a67625ac8f40
        Creation time                   : Feb 03, 2019 09:10:22.265405900 UTC
        Description                     : DESKTOP-NPM7RCA G: 2/3/2019
        Number of key protectors        : 2

Key protector 0:
        Identifier                      : f0f61678-fb6f-4ab1-934a-7094f5b68a85
        Type                            : Password

Key protector 1:
        Identifier                      : 012e56c1-4ed6-4527-8abf-7a9f29e0b521
        Type                            : Recovery password
```

Obr. 3.1 Ukázka výstupu nástroje `bdeinfo`

Podpora BitLockeru, kterou libbde poskytuje, je velice rozsáhlá a dokáže pracovat se všemi existujícími formáty a verzemi. Dokumentace pro tuto knihovnu také obsahuje obsáhlý popis BitLockeru, formátu hlaviček a metadat[10], který byl neocenitelný při přípravě této diplomové práce.

Bohužel i přes tyto dobré zprávy má knihovna libbde několik vlastností, které z ní dělají nevhodného kandidáta na nástroj pro každodenní použití. Předně je zde problém s neexistující podporou pro zápis — otevřené zařízení je připojitelné pouze pro čtení a ačkoli je podpora pro zápis plánována již od roku 2014[1], stále není k dispozici. Kvůli tomu se může jednat o nástroj vhodný pro forenzní analýzu nebo záchranu dat, ale

TODO: citaci nebo stačí takhle?

[1] https://github.com/libyal/libbde/issues/1

například pro vytvoření šifrovaného flash disku, který bude sloužit ke sdílení dat mezi Windows a Linuxem, je toto řešení nepoužitelné.

Z hlediska případného dalšího vývoje nebo použití v jiných projektech, je také minimálně diskutabilní použití některých technologií. Knihovna libbde je součástí většího projektu *libyal*[2)], který obsahuje několik desítek různých knihoven. Jednou z nich je i knihovna *libaes*[3)], která poskytuje multiplatformní implementaci AES a kterou libbde používá pro dešifrování dat. Používání „vlastních“ implementací šifrování je obecně nedoporučováno a preferuje se použití standardních knihoven jako například *libopenssl* nebo *libgcrypt*, které mají teoreticky zaručit „správnost“ implementace kryptografických algoritmů. Použitá knihovna libaes například při dešifrování klíčů kvůli špatné implementaci AES-CCM vůbec nekontroluje přiložený MAC tag[4)].

Pro potenciální uživatele může být problematická také implementace v *user space* pomocí FUSE[5)], které sice výrazně zjednodušuje vývoj, ale může mít velmi výrazný vliv na výkon — zpomalení oproti implementaci v kernelu může nastat až o 83 % a zatížení CPU může narůst až o 31 %[14].

Nevýhodou FUSE je také, že vytvořené otevřené „zařízení“ ve skutečnosti není systémový nástroji rozpoznané jako blokové zařízení a to právě proto, že bylo vytvořeno v user space. Takto vytvořené zařízení sice lze připojit pomocí příkazu `mount`, kdy je na něm úspěšně rozpoznám souborový systém NTFS a jako takový je úspěšně připojen, ale protože jej systém nerozpozná jako nově přidané zařízení, není možné jej detekovat a připojit automaticky.

Pro případnou integraci do existujících nástrojů pro práci s úložnými zařízeními, může být teoreticky problém také licence — libbde je dostupná pod licencí GNU LGPL verze 3, která je zpětně nekompatibilní s verzí 2 a použití takové knihovny by (i u nástrojů a knihoven a dostupných pod licencí GNU LGPL verze 2 a novější) automaticky změnilo licenci výsledného programu na GNU LGPL verze pouze 3[1].

Obecně lze říci, že knihovna libbde a s ní dostupné uživatelské nástroje jsou velmi užitečné pro případnou „záchranu“ dat ze zařízení zašifrovaného pomocí BitLockeru, při nemožnosti použít Windows, případně pro různou analýzu dat, ale bohužel nevhodné pro každodenní použití. Volba použitých technologií (FUSE, vlastní implementace kryptografických funkcí) z libbde také dělá nevhodného kandidáta na další rozšíření a případné začlenění do existujících aplikací a nástrojů.

[2)] https://github.com/libyal/libyal/wiki/Overview

[3)] https://github.com/libyal/libcaes/wiki

[4)] https://github.com/libyal/libcaes/issues/2

[5)] FUSE je interface pro práci se souborovými systémy v user space, bez potřeby programovat přímo v kernel space, a je tedy „dostupný“ i pro neprivilegované uživatele[13].

3.2 Dislocker

Druhým projektem, který se zabývá podporou BitLockeru v linuxových systémech, je nástroj Dislocker. Velkou výhodou tohoto nástroje je, že (na rozdíl od knihovny libbde) podporuje i zápis na otevřené BitLocker zařízení. Kromě Linuxu podporuje také MacOS X a BSD systémy. Součástí Dislockeru jsou kromě samotného nástroje `dislocker` i nástroj pro analýzu metadat `dislocker-metadata` (ukázka z jeho výstupu je na obrázku 3.2, nástroj `dislocker-file` sloužící pro dešifrování celého zařízení a uložení dat do souboru a také nástroj `dislocker-find` pro nalezení blokových zařízení s BitLocker formátem.

```
====================[ BitLocker information structure ]====================
  Signature: '-FVE-FS-'
  Total Size: 0x0370 (880) bytes (including signature and data)
  Version: 2
  Current state: ENCRYPTED (4)
  Next state: ENCRYPTED (4)
  Encrypted volume size: 104857600 bytes (0x6400000), ~100 MB
  Size of convertion region: 0 (0)
  Number of boot sectors backuped: 16 sectors (0x10)
  First metadata header offset:  0x2195000
  Second metadata header offset: 0x2c1d000
  Third metadata header offset:  0x373a000
  Boot sectors backup address:   0x21a5000
  ----------------------------{ Dataset header }----------------------------
    Dataset size: 0x00000324 (804) bytes (including data)
    Unknown data: 0x00000001 (always 0x00000001)
    Dataset header size: 0x00000030 (always 0x00000030)
    Dataset copy size: 0x00000324 (804) bytes
    Dataset GUID: '1F8BF933-8323-4C97-8A89-A67625AC8F40'
    Next counter: 10
    Encryption Type: AES-XTS-128 (0x8004)
    Epoch Timestamp: 1549185022 sec, that to say Sun Feb  3 09:10:22 2019
  ------------------------------------------------------------------------
```

Obr. 3.2 Ukázka výstupu nástroje `dislocker-metadata`

Nevýhody Dislockeru jsou pak podobné jako u výše zmíněné knihovny libbde — implementace využívá FUSE a součástí nástroje je také vlastní implementace AES (ale zde, zdá se, bez chyb). Jedná se také pouze o nástroj pro koncové uživatele, nikoli o knihovnu a případná integrace se systémovými nástroji by tak byla složitější. Hlavní nevýhodou je však (pravděpodobně) ukončený vývoj tohoto nástroje — poslední commit v Git repozitáři je z roku 2017.

Obecně je nástroj Dislocker pro koncové uživatele vhodnější, než knihovna libbde a s ní spojené nástroje a to především proto, že umožňuje na BitLocker zařízení také

zapisovat. Sdílí ovšem stejné problémy, které z něj nedělají ideální řešení pro další použití či případné rozšíření nebo začlenění do existujících aplikací a nástrojů.

II. PROJEKTOVÁ ČÁST

4 Nadpis

4.1 Podnadpis

ZÁVĚR

Text závěru

SEZNAM POUŽITÉ LITERATURY

[1] Frequently Asked Questions about the GNU Licenses. Dostupné z: https://www.gnu.org/licenses/gpl-faq.html.en#v2v3Compatibility

[2] Programming reference for Windows API. Dostupné z: https://docs.microsoft.com/en-us/windows/desktop/api/minwinbase/ns-minwinbase-filetime

[3] Security WMI Providers. Dostupné z: https://docs.microsoft.com/en-us/windows/desktop/secprov/bitlocker-drive-encryption-provider

[4] VeraCrypt Volume Format Specification. Dostupné z: https://www.veracrypt.fr/en/VeraCrypt%20Volume%20Format%20Specification.html

[5] Windows Vista support has ended. 2017. Dostupné z: https://support.microsoft.com/en-us/help/22882/windows-vista-end-of-support

[6] Carrier, B.: *File system forensic analysis*. London: Addison-Wesley, první vydání, 2005, ISBN 978-0321268174.

[7] Coltel, R.; Schauer, H.: Dislocker. Dostupné z: https://github.com/Aorimn/dislocker

[8] Ferguson, N.: AES-CBC + Elephant diffuser. 2006.

[9] Hall, J.; Poggemeyer, L.: BitLocker recovery guide. Dostupné z: https://docs.microsoft.com/en-us/windows/security/information-protection/bitlocker/bitlocker-recovery-guide-plan

[10] Metz, J.: BitLocker Drive Encryption (BDE) format specification. Dostupné z: https://github.com/libyal/libbde/blob/master/documentation/BitLocker%20Drive%20Encryption%20(BDE)%20format.asciidoc

[11] Metz, J.: Building. Dostupné z: https://github.com/libyal/libbde/wiki/Building

[12] Metz, J.: Library and tools to access the BitLocker Drive Encryption (BDE) encrypted volumes. 2018. Dostupné z: https://github.com/libyal/libbde

[13] Singh, S.: Develop your own filesystem with FUSE. Dostupné z: https://developer.ibm.com/articles/l-fuse/

[14] Vangoor, B. K. R.; Tarasov, V.: To FUSE or Not to FUSE: Performance of User-Space File Systems. In *FAST'17 Proceedings of the 15th Usenix Conference on File and Storage Technologies*, Santa Clara, CA, USA: USENIX Association Berkeley, 2017, ISBN 978-1-931971-36-2, s. 59–72.

SEZNAM POUŽITÝCH SYMBOLŮ A ZKRATEK

AES	Advanced Encryption Standard
ASCII	American Standard Code for Information Interchange
BSD	Berkeley Software Distribution
CBC	Cipher Block Chaining
CCM	Counter with CBC-MAC
CNG	CryptoAPI Next Generation
CPU	Central Processing Unit
DM	Device Mapper
FS	File System
FVE	Full Volume Encryption
FVEK	Full Volume Encryption Key
FUSE	Filesystem in Userspace
GNU	GNU is Not Unix
GUID	Globally Unique Identifier
LGPL	Lesser General Public License
LUKS	Linux Unified Key Setup
MAC	Message Authentication Code
MFT	Master File Table
NTFS	New Technology File System
OEM	Original Equipment Manufacturer
PIN	Personal Identification Number
TPM	Trusted Platform Module
USB	Universal Serial Bus
UTF	Unicode Transformation Format
VMK	Volume Master Key
XEX	Xor-Encrypt-Xor
XTS	XEX-based Tweaked-codebook with Ciphertext Stealing

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I. NÁZEV PŘÍLOHY

Obsah přílohy