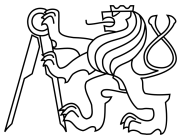
ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE
FAKULTA INFORMAČNÍCH TECHNOLOGIÍ



ZADÁNÍ DIPLOMOVÉ PRÁCE

Název:	Bezpečnostní analýza programu BestCrypt Volume Encryption
Student:	Bc. Juraj Horňák
Vedoucí:	Ing. Josef Kokeš
Studijní program:	Informatika
Studijní obor:	Počítačová bezpečnost
Katedra:	Katedra počítačových systémů
Platnost zadání:	Do konce letního semestru 2016/17

Pokyny pro vypracování

Seznamte se s programem BestCrypt Volume Encryption (BCVE).
Popište BCVE z uživatelského hlediska.
Zdokumentujte kryptologická primitiva používaná programem dle tvrzení výrobce.
Technikou reverzního inženýrství analyzujte bootovací kód BCVE.
Ověřte, že program bezpečně pracuje s heslem.
Pokuste se také ověřit, že BCVE implementujte kryptologická primitiva dle specifikací.

Seznam odborné literatury

Dodá vedoucí práce.

L.S.

prof. Ing. Róbert Lórencz, CSc.
vedoucí katedry

prof. Ing. Pavel Tvrdík, CSc.
děkan

V Praze dne 14. února 2016

České vysoké učení technické v Praze
Fakulta informačních technologií
Katedra počítačových systémů

Diplomová práca

Bezpečnostní analýza programu BestCrypt Volume Encryption

Bc. Juraj Horňák

Vedúci práce: Ing. Josef Kokeš

4. mája 2016

Poďakovanie

Rád by som poďakoval môjmu vedúcemu, Ing. Josefovi Kokešovi, za to, že si na mňa našiel čas a viedol ma pri tvorbe tejto diplomovej práce. Rovnako by som chcel poďakovať mojej rodine a priateľom, ktorí ma podporovali počas celého magisterského štúdia.

Prehlásenie

Prehlasujem, že som predloženú prácu vypracoval(a) samostatne a že som uviedol(uviedla) všetky informačné zdroje v súlade s Metodickým pokynom o etickej príprave vysokoškolských záverečných prác.

Beriem na vedomie, že sa na moju prácu vzťahujú práva a povinnosti vyplývajúce zo zákona č. 121/2000 Sb., autorského zákona, v znení neskorších predpisov. V súlade s ustanovením § 46 odst. 6 tohoto zákona týmto udeľujem bezvýhradné oprávnenie (licenciu) k užívaniu tejto mojej práce, a to vrátane všetkých počítačových programov ktoré sú jej súčasťou alebo prílohou a tiež všetkej ich dokumentácie (ďalej len „Dielo“), a to všetkým osobám, ktoré si prajú Dielo užívať.

Tieto osoby sú oprávnené Dielo používať akýmkoľvek spôsobom, ktorý neznižuje hodnotu Diela, a za akýmkoľvek účelom (vrátane komerčného využitia). Toto oprávnenie je časovo, územne a množstevne neobmedzené. Každá osoba, ktorá využije vyššie uvedenú licenciu, sa však zaväzuje priradiť každému dielu, ktoré vznikne (čo i len čiastočne) na základe Diela, úpravou Diela, spojením Diela s iným dielom, zaradením Diela do diela súborného či zpracovaním Diela (vrátane prekladu), licenciu aspoň vo vyššie uvedenom rozsahu a zároveň sa zaväzuje sprístupniť zdrojový kód takého diela aspoň zrovnateľným spôsobom a v zrovnateľnom rozsahu ako je zprístupnený zdrojový kód Diela.

V Prahe 4. mája 2016 .....

Odkaz na túto prácu

Horňák, Juraj. *Bezpečnostní analýza programu BestCrypt Volume Encryption.* Diplomová práca. Praha: České vysoké učení technické v Praze, Fakulta informačních technologií, 2016.

Abstrakt

Táto diplomová práca sa zaoberá analýzou softvéru na šifrovanie zväzkov pevných diskov alebo výmenných zariadení – BestCrypt Volume Encryption. Analýza sa zameriava na bezpečnostné aspekty bootovacieho kódu aplikácie. Práca popisuje výsledky získané pomocou reverznej analýzy bootovacieho kódu. Obzvlášť sa venuje odvodeniu šifrovacieho kľúča z hesla užívateľa a procesu šifrovania resp. dešifrovania. Ďalej práca overuje korektnosť implementácie kryptologických primitív použitím vytvoreného nástroja na dešifrovanie sektorov.

Kľúčové slová BestCrypt Volume Encryption, symetrická kryptografia, operačný mód XTS, šifrovanie zväzkov

Abstract

This diploma thesis deals with the analysis of software used to encrypt volumes of fixed or removable disks – BestCrypt Volume Encryption. The analysis is focused on the security aspects of application's boot code. The thesis describes results obtained by performing a reverse analysis of the code, especially the derivation of the encryption key from user's password and the process of encryption/decryption. The correctness of application's cryptographic primitives is verified by implementing a program for sector decryption.

Keywords BestCrypt Volume Encryption, Symmetric Cryptography, XTS Operation Mode, Volume Encryption

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam algoritmov

Úvod

V súčasnej dobe ľudia používajú počítače v rôznych formách každý deň takmer na každom kroku. V počítačoch si bežne uchovávajú veľké množstvo užitočných osobných a pracovných informácií, ktorých strata alebo krádež by mohli mať na pôsobenie dotknutých jednotlivcov alebo firiem nepriaznivý vplyv. Jednou z možností, ako znížiť negatívne dopady úniku informácií alebo ako sa vyhnúť možnosti zneužitia cenných počítačových dát neoprávnenými osobami, je ochrana dôležitých informácií prostredníctvom šifrovania. Na trhu existuje viacero softvérových riešení určených na šifrovanie dát uložených na pevných diskoch, napríklad BitLocker [24] alebo VeraCrypt [4]. Ďalším zo zástupcov tohto typu softvéru je program BestCrypt Volume Encryption. Táto práca sa podrobne zaoberá práve týmto programom.

Cieľom tejto diplomovej práce je pomocou techník reverzného inžinierstva vykonať analýzu bootovacej časti vyššie spomenutého programu zameranú na jej bezpečnostné aspekty, t. j. zistiť, ako program na úrovni bootovacieho kódu pracuje s prístupovým heslom užívateľa, pomocou ktorého je možné zašifrovaný zväzok dešifrovať, ako ukladá a chráni šifrovacie kľúče, alebo akým spôsobom používa kryptologické algoritmy. Autor softvéru BestCrypt Volume Encryption udelil súhlas na reverznú analýzu jeho produktu. Práca má zároveň overiť, či sú analyzovaným programom používané kryptologické primitíva implementované presne podľa ich špecifikácií.

Následujúca kapitola sa zaoberá zhrnutím o programe BestCrypt Volume Encryption dostupných informácií. Na začiatku oboznamuje čitateľa so základnými funkcionalitami poskytovanými týmto programom a vymedzením cieľov analýzy. Ďalej sú v nej zdokumentované používané kryptologické algoritmy a spôsoby autentifikácie užívateľa. Ďalšia kapitola popisuje priebeh bootovacieho procesu, uplatnené postupy reverznej analýzy a nástroje pri nej použité. Kapitola 3 diskutuje výsledky získané uskutočnenou reverznou analýzou a posledná kapitola sa zameriava na overovanie korektnosti implementovaných kryptologických primitív prostredníctvom vytvoreného programu slúžiaceho na dešifrovanie sektorov.

Kapitola 1

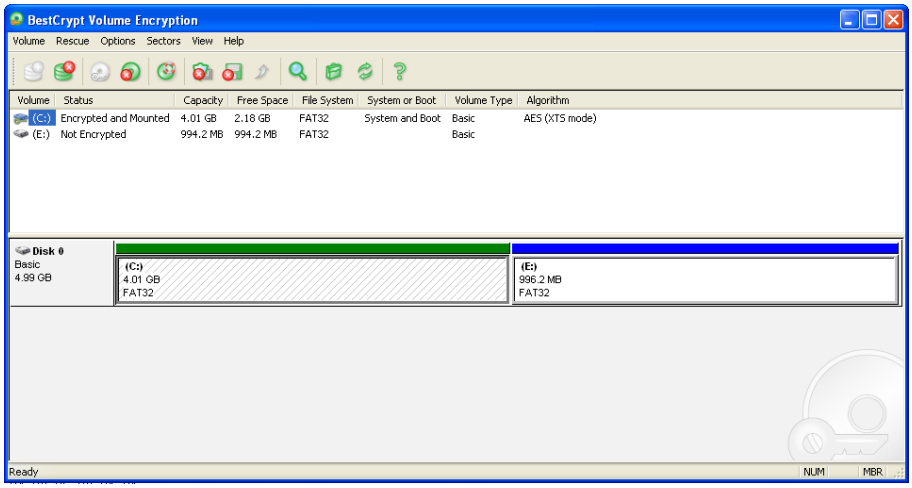
Program BestCrypt Volume Encryption

Program BestCrypt Volume Encryption je produktom fínskej spoločnosti Jetico – spoločnosti, ktorá sa od roku 1995 zaoberá poskytovaním štandardizovaných riešení v oblasti ochrany citlivých počítačových dát (informácie o kreditných kartách, zdravotná dokumentácia a pod.) [16]. Softvér BestCrypt Volume Encryption slúži, ako to už z jeho názvu vyplýva, na ochranu počítačových dát na úrovni zväzkov, napríklad pri strate alebo krádeži počítača resp. dátového úložiska. Jeho primárnym účelom je permanentne zašifrovať zväzky pevného disku (HDD) alebo výmenného média (napríklad USB flash disku) tak, aby neautorizované osoby nemali možnosť prístupu k informáciám na daných zväzkoch.

Aplikácia BestCrypt Volume Encryption je určená pre operačné systémy Microsoft Windows. V čase písania tejto práce bol tento softvér dostupný vo verzii 3.70.22. Táto kapitola sa venuje zmapovaniu základných vlastností práve tejto verzie programu deklarovaných výrobcom v softvérovom manuále [15]. Kapitola čitateľovi prináša stručný užívateľský opis jednotlivých funkcionalít programu BestCrypt Volume Encryption a spôsobu ich použitia. Zároveň popisuje bezpečnostné charakteristiky týkajúce sa šifrovania a overovania užívateľov pri prístupe k chráneným zväzkom. Ďalšie kapitoly tejto diplomovej práce okrem iného zároveň overujú pravdivosť týchto informácií poskytnutých autorom.

1.1 Základné funkcionality

Program BestCrypt Volume Encryption, alebo skrátene BCVE, ponúka užívateľovi viaceré užitočné funkcionality. Ich úplný zoznam spolu s detailnými popismi použitia je spracovaný v softvérovej príručke [15]. Prostredníctvom nastavenia hlavnej programovej časti aplikácie BCVE – súbor `bcfmgr.exe`



Obr. 1.1: Hlavná programová časť BCVE

(obrázok 1.1) – môžu byť užívateľom zvolené tieto podporované funkcionality:

- **Šifrovanie rôznych typov zväzkov**
  BCVE umožňuje šifrovanie rôznych typov zväzkov na pevných diskoch a výmenných médiách. Napríklad šifrovanie jednoduchých zväzkov tvorených jedným diskovým oddielom alebo zväzkov tvorených viacerými diskovými oddielmi (zrkadlené zväzky, zväzky RAID-5, atď.). Táto práca sa zameriava na analýzu bootovacieho kódu pri zašifrovanom zavádzacom/systémovom zväzku tvorenom jedným diskovým oddielom.

- **Autentifikácia pri bootovaní**
  Po zašifrovaní zavádzacieho zväzku (boot volume) alebo systémového zväzku (system volume) bude BCVE vyžadovať pred spustením samotného operačného systému autentifikáciu užívateľa. Operačný systém Windows považuje za zavádzací zväzok ten, ktorý obsahuje systémové súbory (zvyčajne adresár `WINDOWS`), a za systémový zväzok ten, ktorý obsahuje súbory nevyhnutné k spusteniu systému Microsoft Windows (napríklad `NTLDR`, `boot.ini`, `ntdetect.com` v prípade Windows XP) [26]. K spusteniu operačného systému dôjde až po zadaní správneho prístupového hesla (viac o priebehu autentifikácie pri bootovaní v sekcii 1.2.3). Proces autentifikácie pri bootovaní a naň nadväzujúce akcie sú primárnym cieľom analýzy tejto práce.

- **Dvojstupňová autentifikácia**
  BCVE ponúka užívateľovi možnosť používania dvojstupňovej autentifikácie založenej na bežnom výmennom médiu (napríklad USB flash disku) alebo špeciálnych bezpečnostných tokenoch – SafeNet (Aladdin) eToken PRO alebo eToken Java. Aladdin eToken je výmenné zariadenie

podobné USB flash disku, ktoré je navrhnuté na uchovávanie informácií v bezpečnej forme [32]. Pri dvojstupňovej autentifikácii sú tak potrebné na sprístupnenie zašifrovaného zväzku dve veci: príslušný bezpečnostný token resp. výmenné médium s uloženým šifrovacím kľúčom a znalosť prístupového hesla. Spôsob, akým BCVE pri autentifikácii používa prístupové heslo, je popísaný v sekcii 1.2.3. Presné fungovanie dvojstupňovej autentifikácie pri analýze nebolo detailne preskúmané.

- **Bezobslužný reštart**
  BCVE umožňuje reštart počítačového systému so zašifrovaným zavádzacím alebo systémovým zväzkom bez nutnosti manuálneho zadávania hesla pri bootovaní. BCVE zabezpečuje bezobslužný reštart (unattended restart) pomocou špeciálneho hardvéru Trusted Platform Module (TPM) [14] umiestneného na základnej doske počítača. Ak užívateľ povolí bezobslužný reštart, tak BCVE pred reštartom uloží šifrovací kľúč do registra TPM. Pri bootovaní tento kľúč z registra TPM použije a následne ho z registra TPM vymaže. Práca analyzuje túto funkcionalitu len okrajovo.

- **Presun šifrovacieho kľúča na externé zariadenie**
  Za normálnych okolností BCVE ukladá šifrovací kľúč pre určitý zväzok v šifrovanej forme na tom istom zväzku. Na zvýšenie bezpečnosti BCVE umožňuje užívateľovi presúvanie kľúča pre daný zväzok v šifrovanej podobe na externé zariadenie, napríklad na výmenné médium (čo je predpokladom pre dvojstupňovú autentifikáciu) alebo vzdialený server, prostredníctvom ktorého počítač bootuje. V prípade, že je na výmenné médium uložený kľúč pre zavádzací resp. systémový zväzok, tak sa musí nastaviť bootovanie z daného média. Na vzdialený server BCVE umožňuje ukladanie len šifrovacieho kľúča pre zavádzací resp. systémový zväzok, a to takým spôsobom, že sa na vzdialený server umiestni súbor s bootovacím obrazom obsahujúcim šifrovací kľúč a z daného servera musí potom počítač bootovať. Realizácia tejto funkcionality nebola preskúmaná.

1.2 Bezpečnostné charakteristiky

Táto podkapitola sa zaoberá bezpečnostnými charakteristikami BCVE. Konkrétne mapuje použité šifrovacie algoritmy a ich operačný mód. Ďalej sa zaoberá podporovanými spôsobmi autentifikácie užívateľa v prípade zašifrovaného bežného zväzku a priebehom autentifikácie pri bootovaní v prípade zašifrovaného zavádzacieho resp. systémového zväzku.

1.2.1 Šifrovacie algoritmy

BCVE používa na šifrovanie algoritmy symetrickej kryptografie. Konkrétne sú na výber štyri blokové šifry, ktoré sa stali finalistami výberového procesu na normu AES (Advanced Encryption Standard) nahrádzajúcu DES (Data Encryption Standard). Každá z týchto podporovaných blokových šifier umožňuje šifrovať 128-bitové bloky pomocou 128, 192 alebo 256 bitov dlhých šifrovacích kľúčov.

Užívateľ BCVE si na šifrovanie zväzkov môže zvoliť jednu z týchto blokových šifier [15]:

- **AES (Rijndael)**
  Pôvodný názov blokovej šifry AES je Rijndael. Jej autormi sú Joan Daemen a Vincent Rijmen. Rijndael má štruktúru substitučnej permutačnej siete s 10, 12 alebo 14 rundami v závislosti na dĺžke šifrovacieho kľúča. Táto bloková šifra sa stala víťazom výberového procesu na normu AES [27], a preto dnes patrí medzi najčastejšie používané šifrovacie algoritmy. Viac informácií o AES v [6].

- **RC6**
  Blokovú šifru RC6 navrhli Matt Robshaw, Ray Sidney a Yiqun Lisa Yin. Šifra RC6 vychádza z blokovej šifry RC5 a jej základnú štruktúru tvorí Feistelova sieť s 20 rundami. RC6 je rýchla bloková šifra s jednoduchým algoritmom [22], no ako jediná z finalistov v súťaži AES sa nemôže úplne voľne používať [3]. Ide o proprietárny algoritmus spoločnosti RSA Security. Viac o blokovej šifre RC6 v [30].

- **Serpent**
  Autormi blokovej šifry Serpent sú Ross Anderson, Eli Biham a Lars Knudsen. Základom šifry Serpent je substitučná permutačná sieť s 32 rundami. Serpent je považovaný za veľmi bezpečnú šifru [22], no kvôli vysokému počtu rúnd ide o pomalší algoritmus, a preto Serpent skončil na 2. mieste v súťaži AES [35]. Podrobnosti o šifre Serpent v [1].

- **Twofish**
  Blokovú šifru Twofish navrhli Bruce Schneier, John Kelsey, Chris Hall, Niels Ferguson, David Wagner a Doug Whiting. Algoritmus šifry Twofish je inšpirovaný šifrou Blowfish. Je to 16-rundová Feistelova šifra, pre ktorú sú typické S-boxy závislé na kľúči. Detaily k šifre Twofish v [34].

Výrobca v [15] deklaruje, že každý z týchto implementovaných algoritmov na šifrovanie 128 bitov dlhých blokov používa 256-bitový šifrovací kľúč. Zároveň tvrdí, že ako operačný mód bol pre tieto blokové šifry zvolený mód XTS.

1.2.2 Operačný mód XTS

Samotné blokové šifry umožňujú šifrovanie len jedného bloku dát (veľkosť bloku je daná konkrétnou blokovou šifrou). Na šifrovanie dlhších dát sa preto používajú operačné módy. Operačné módy sú pravidlá, ktorých dodržiavanie umožňuje opakovane a bezpečne používať blokové šifry s tým istým šifrovacím kľúčom. Rôzne módy napĺňajú rôzne bezpečnostné požiadavky. Operačný mód XTS bol špeciálne navrhnutý na šifrovanie blokovo orientovaných úložísk dát, kde sa každý blok úložiska spracováva nezávisle na ostatných blokoch. V roku 2007 ho Inštitút pre elektrotechnické a elektronické inžinierstvo (IEEE) prijal za štandard [12].

Vo zvyšku tejto práce sa bude pojem „blok“ používať v dvoch rôznych významoch. Na rozlíšenie týchto dvoch významov bude blok, s ktorým pracuje bloková šifra, označovaný ako blok blokovej šifry, a blok, s ktorým pracuje operačný mód XTS, bude označovaný ako blok úložiska.

XTS spolu s príslušnou blokovou šifrou vytvára tzv. tweakable blokovú šifru. Pre klasické blokové šifry platí, že majú dva vstupy (vstupný blok blokovej šifry $M \in \{0,1\}^n$ a kľúč $K \in \{0,1\}^k$) a produkujú jeden výstup (výstupný blok blokovej šifry $C \in \{0,1\}^n$):

$$E : \{0,1\}^k \times \{0,1\}^n \rightarrow \{0,1\}^n.$$

Tweakable blokové šifry majú na rozdiel od klasických blokových šifier naviac tretí vstup nazvaný tweak ($T \in \{0,1\}^t$) [23]:

$$E' : \{0,1\}^k \times \{0,1\}^n \times \{0,1\}^t \rightarrow \{0,1\}^n.$$

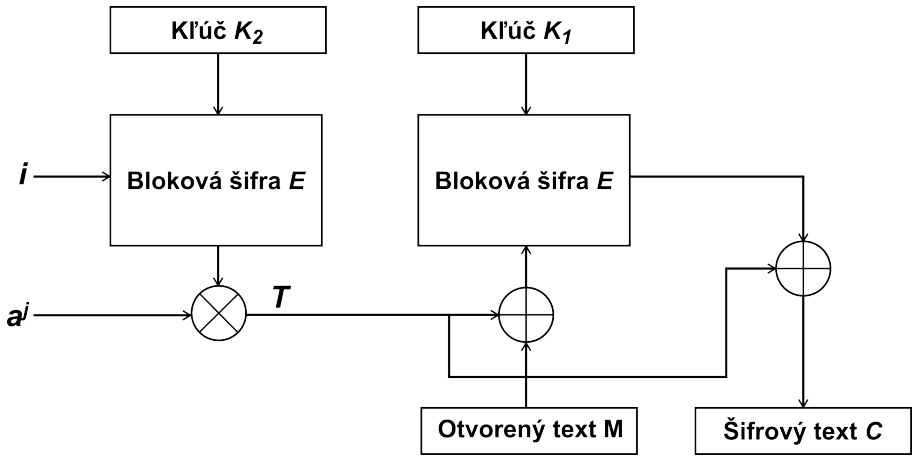
Tweak je číselná hodnota, ktorá zachytáva logickú pozíciu bloku úložiska resp. jeho adresu. XTS ako tweak používa kombináciu pozície aktuálne spracovávaného bloku úložiska v rámci celého úložiska a indexu aktuálne šifrovaného bloku blokovej šifry v rámci daného bloku úložiska.

Operačný mód XTS (XEX encryption mode with tweak and ciphertext stealing) je vlastne mód XEX (XOR Encryption XOR) so skracovaním šifrového textu (ciphertext stealing). Označenie XTS ako módu so skracovaním šifrového textu znamená, že bez nutnosti rozširovania textu umožňuje šifrovanie blokov úložiska, ktorých veľkosť nie je deliteľná veľkosťou bloku príslušnej blokovej šifry. V takom prípade sa posledné dva bloky blokovej šifry (jeden neúplný) v rámci bloku úložiska spracujú inou transformáciou (viac v [12]). V opačnom prípade šifrovanie jedného bloku blokovej šifry pomocou operačného módu XTS prebieha nasledujúcim spôsobom [12]:

$$C = E_{K_1}(M \oplus T) \oplus T,$$

kde funkcia $E_K()$ predstavuje šifrovanie blokovou šifrou so šifrovacím kľúčom K, $\oplus$ označuje operáciu *XOR*, M je otvorený blok blokovej šifry, C je zašifrovaný blok blokovej šifry a T je tweak. Pre tweak T platí:

$$T = E_{K_2}(i) \otimes \alpha^j,$$



Obr. 1.2: Schéma šifrovania pomocou operačného módu XTS

kde i je adresa bloku úložiska, j je index bloku blokovej šifry v rámci bloku úložiska i, α je primitívny prvok konečného telesa $GF(2^{128})$ a operácia $\otimes$ vyjadruje modulárne násobenie dvoch polynómov nad binárnym telesom $GF(2)$ modulo $x^{128} + x^7 + x^2 + x + 1$. Tento postup zároveň schematicky popisuje aj obrázok 1.2. Z daného postupu vyplýva, že sa okrem dátového kľúča K_1 pri šifrovaní pomocou XTS používa aj druhý kľúč K_2, takzvaný tweak kľúč.

Dešifrovanie má podobnú schému ako je na obrázku 1.2. Pri dešifrovaní sa však vymení otvorený text so šifrovým textom a pomocou dátového kľúča K_1 sa bude dešifrovať:

$$M = D_{K_1}(C \oplus T) \oplus T,$$

$$T = E_{K_2}(i) \otimes \alpha^j.$$

V prípade BCVE sa pod pojmom blok vo výraze blokovo orientované úložisko rozumie sektor na disku. XTS sa v BCVE štandardne používa na šifrovanie fyzických sektorov s veľkosťou 512 bajtov a ako tweak slúži kombinácia logickej pozície sektora (výrobca ju bližšie nešpecifikuje) na pevnom disku alebo na výmennom médiu a indexu aktuálne šifrovaného 128 bitov dlhého bloku blokovej šifry v rámci aktuálne spracovávaného sektora. XTS šifrovací kľúč je tvorený dvojicou kľúčov (256-bitový dátový kľúč a tweak kľúč s nešpecifikovanou veľkosťou).

1.2.3 Autentifikácia užívateľa

Užívateľ pred šifrovaním každého zväzku musí nastaviť hlavné prístupové heslo (na zvýšenie bezpečnosti sa môže zároveň použiť jedna z podporovaných foriem dvojstupňovej autentifikácie), ktorým bude daný zväzok chránený. Po nastavení hlavného hesla pre daný zväzok BCVE umožňuje užívateľovi pre ten istý zväzok nastaviť ešte niekoľko ďalších hesiel. Tieto doplnkové heslá sú vhodné

napríklad na poskytnutie dočasných prístupov k zašifrovanému zväzku. Doplnkové heslá je možné kedykoľvek zmazať. Po zašifrovaní zväzku sa užívateľ bez zadania hlavného alebo jedného z doplnkových hesiel k dešifrovaným informáciám už nedostane. Ak zadá správne heslo, tak si BCVE odvodí šifrovacie kľúče a pripojí zašifrovaný zväzok. Zároveň sa zabezpečí, že pri čítaní informácií zo zväzku dôjde k automatickému dešifrovaniu požadovaných dát a pri zápise k šifrovaniu zapisovaných dát. Užívateľ môže toto automatické šifrovanie/dešifrovanie dát zastaviť tým, že šifrovaný zväzok odpojí. Po odpojení zväzku BCVE vymaže šifrovacie kľúče k danému zväzku [15].

Po zašifrovaní zavádzacieho resp. systémového zväzku hlavná programová časť BCVE prepíše hlavný spúšťací záznam (MBR) bootovacím programom, ktorý realizuje autentifikáciu pri bootovaní [15]. Hneď po diagnostickej kontrole hardvéru (POST) sa ako prvý kód spúšťa kód BCVE a na obrazovke sa objaví výzva žiadajúca užívateľa o zadanie prístupového hesla (obrázok 1.3). Proces zavádzania operačného systému Windows tak začne až potom, čo užívateľ zadá správne prístupové heslo. Bez tohto hesla nie je možné dešifrovať sektory, na ktorých sú uložené systémové súbory operačného systému potrebné k bootovaniu, čiže bez hesla nie je možné ani nabootovať samotný operačný systém.

Obr. 1.3: Autentifikácia pri bootovaní

Kapitola 2

Reverzná analýza bootovacieho kódu

Hlavným cieľom tejto práce je pomocou postupov reverzného inžinierstva zanalyzovať bezpečnostné aspekty bootovacieho kódu BCVE. Keďže tento kód beží pred samotným spustením operačného systému, tak je z pohľadu reverznej analýzy dôležité poznať sekvenciu a význam jednotlivých operácií vykonávaných pred zavedením systému Windows. Autor BCVE tvrdí, že pôvodný hlavný spúšťací záznam (master boot record) je nahradený jeho bootovacím kódom. Z týchto dôvodov bude v tejto kapitole popísaný štandardný priebeh bootovacieho procesu a štruktúra spúšťacieho záznamu. Ďalej bude diskutovaný postup odhalenia umiestnenia bootovacieho kódu BCVE a postupy reverznej analýzy operácií pracujúcich s heslom a šifrovacím kľúčom a analýzy procesu šifrovania a dešifrovania dát na zavádzacom/systémovom zväzku. Záver kapitoly sa zameriava na nástroje používané pri reverznej analýze.

2.1 Bootovací proces

Slovo bootovanie označuje zavádzanie operačného systému pri zapnutí alebo reštartovaní počítača. V jednoduchosti ide o skopírovanie jadra operačného systému z pamäťového nosiča do RAM pamäte a jeho následné spustenie. Je to zložitý, často viacstupňový proces, ktorý sa môže na rôznych systémoch navzájom líšiť.

Na počítačoch s procesorom z rodiny x86 (IBM PC kompatibilné) je proces bootovania riešený zvyčajne dvoma spôsobmi, a to buď pomocou klasickej metódy BIOS-MBR alebo novšej metódy UEFI-GPT. Metóda UEFI-GPT je modernejšia, a preto podporuje používanie väčšieho počtu diskových oddielov, pričom jednotlivé oddiely môžu mať zároveň aj väčšiu veľkosť. V porovnaní s BIOS-MBR zvyšuje taktiež rýchlosť a bezpečnosť bootovania [19].

Vzhľadom na výber operačného systému používanom pri analýze (sekcia

2.3.1) sa táto podkapitola detailne zameriava len na metódu BIOS-MBR. Na začiatku je podrobne popísaná postupnosť jednotlivých operácií tohto bootovacieho procesu. Druhá časť sa venuje štruktúre hlavného spúšťacieho záznamu a nakoniec sa podkapitola zaoberá odlišnosťami v bootovacom procese, ktoré so sebou prináša používanie programu BCVE.

2.1.1 Priebeh štandardného bootovacieho procesu

Štandardný BIOS-MBR bootovací proces od zapnutia počítača po spustenie samotného operačného systému na IBM PC kompatibilných počítačoch pozostáva z týchto krokov [21]:

1. **Zapnutie/reštart počítača**
   Po zapnutí resp. reštarte počítača sa procesoru posiela signál RESET, ktorý ukončuje všetky aktivity na zberniciach a nastavuje obsah vybraných registrov na počiatočné hodnoty. Procesor potom začína pracovať v 16-bitovom reálnom móde.

2. **Spustenie kódu BIOS-u**
   Po spustení počítača procesor vždy nastaví obsah inštrukčného ukazovateľa (`ip` – instruction pointer) na výrobcom zvolenú adresu reset vektora (napríklad pre procesor 80836 to je fyzická adresa `0xffffff0` [13]) a začne vykonávať nachádzajúce sa tam inštrukcie. Reset vektor ukazuje do oblasti, ktorá obsahuje kód BIOS-u. BIOS predstavuje firmware počítača a implementuje základný program poskytujúci služby na komunikáciu s hardvérom. Jeho programový kód je uložený v ROM, EEPROM alebo flash pamäti umiestnenej na základnej doske počítača.

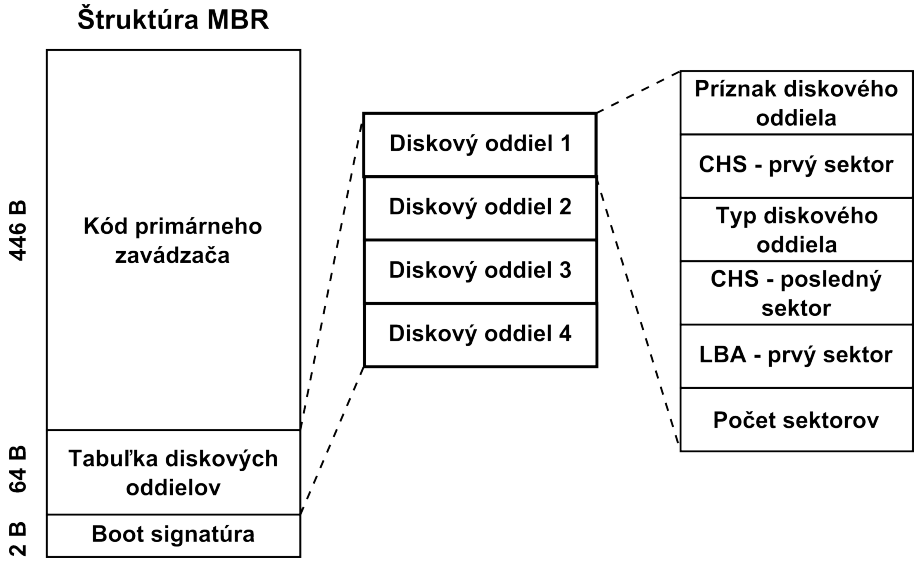
   Medzi základné úlohy BIOS-u pri bootovaní patrí testovanie (POST) a inicializácia hardvéru. Následne BIOS vyvoláva prerušenie `int 19h`. Vyvolaním tohto prerušenia resp. služby sa BIOS na základe svojej konfigurácie pokúša nájsť bootovacie zariadenie (pevný disk, CD, DVD a pod.), ktoré obsahuje bootovací sektor. Bootovací sektor je úplne prvý sektor resp. oblasť 512 bajtov, ktorá sa nachádza na CHS adrese 0/0/1 (cylinder 0, hlava 0, sektor 1) príslušného bootovacieho zariadenia. Aby bol tento sektor platný, musí obsahovať tzv. boot sektor signatúru. To znamená, že bootovací sektor musí byť zakončený dvojicou bajtov `0x55, 0xaa`.

   Na pamäťových médiách, ktoré je možné rozdeliť na viacero diskových oddielov sa prvý sektor nazýva master boot record (MBR). Štruktúra MBR je detailne popísaná v nasledujúcej sekcii. V ostatných prípadoch bootovací sektor nazývame volume boot record (VBR). BIOS však tieto rozdiely nerozlišuje. Snaží sa len nájsť platný bootovací sektor (sektor s boot signatúrou), jeho obsah následne presunúť na preddefinovanú adresu do pamäte RAM a predať mu riadenie.

3. **Spustenie kódu MBR/VBR**
   MBR resp. VBR je presunutý do pamäte RAM na fyzickú adresu `0x7c00`. Segmentový register `cs` a register `ip` sa nastaví tak, aby taktiež ukazovali na túto adresu (zvyčajne `cs = 0x0000` a `ip = 0x7c00`). Program umiestnený na tejto adrese je následne spustený. Vo väčšine prípadov MBR obsahuje krátky program, primárny zavádzač (primary bootloader), ktorého úlohou je nájdenie aktívneho oddielu a následne predanie riadenia do VBR tohto oddielu. VBR väčšinou obsahuje kód sekundárneho zavádzača, ktorý je závislý na operačnom systéme. Zvyčajne jeho hlavnou úlohou je nájsť, nahrať do pamäte RAM a spustiť obsah sektorov, na ktorých je umiestnený súbor skutočného zavádzača operačného systému (napríklad v prípade systému Microsoft Windows XP je to súbor `NTLDR`) [28].

2.1.2 Hlavný spúšťací záznam

Štandardný hlavný spúšťací záznam resp. master boot record je nezávislý na operačnom systéme a pozostáva z dvoch častí – kódu primárneho zavádzača a tabuľky diskových oddielov. Štruktúru MBR názorne zachytáva obrázok 2.1. Prvá časť MBR je tvorená 446-bajtovou oblasťou. V tejto oblasti je zvyčajne umiestnený kód primárneho zavádzača. Jeho hlavným účelom je nájsť aktívny diskový oddiel v tabuľke diskových oddielov, ten skopírovať do pamäte a nakoniec ho spustiť. Tabuľka diskových oddielov obsahuje základné informácie o rozdelení disku na oddiely vo forme záznamov charakterizujúcich vlastnosti

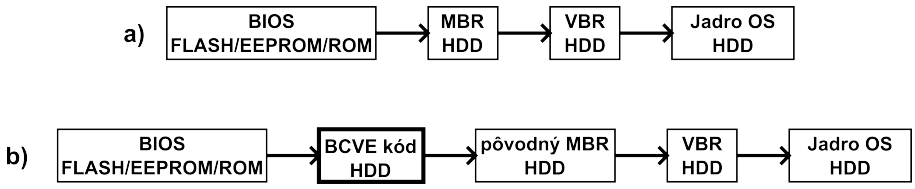


Obr. 2.1: Štruktúra hlavného spúšťacieho záznamu

jednotlivých oddielov. Táto tabuľka môže pozostávať maximálne zo štyroch záznamov. Jednotlivé záznamy udržujú adresy prvého a posledného sektora daného oddiela, veľkosť oddiela, typ systému súborov, podľa ktorého sú organizované dáta na oddiele a informáciu, či je alebo nie je oddiel aktívny, t. j. či je možné z neho bootovať.

2.1.3 Priebeh bootovacieho procesu s použitím BCVE

Priebeh bootovacieho procesu na počítači so zašifrovaným zavádzacím alebo systémovým zväzkom sa od štandardného bootovacieho procesu IBM PC kompatibilných počítačov odlišuje v tom, že sa vkladá medzi jeho druhý a tretí krok (sekcia 2.1.1) určitý medzikrok (rozdiel názorne zachytáva obrázok 2.2). Klasický MBR je nahradený kódom BCVE a ten je spustený hneď po tom, čo sa vykonajú inštrukcie BIOS-u. BCVE si musí obsah originálneho hlavného spúšťacieho záznamu niekam uložiť, pretože po vykonaní všetkých operácií bootovacej časti programu BCVE by sa už malo pokračovať v štandardnom bootovacom procese, počnúc od predania riadenia pôvodnému primárnemu zavádzaču až po zavedenie operačného systému.



Obr. 2.2: Porovnanie bootovacích procesov: a) štandardný BIOS-MBR, b) s použitím BCVE

2.2 Postupy reverznej analýzy

Obsah nasledujúcej podkapitoly sa venuje uplatneným postupom pri reverznej analýze bootovacej časti programu BCVE. Konkrétne sa upriamuje pozornosť na tri najdôležitejšie časti analýzy – umiestnenie bootovacieho kódu BCVE, deriváciu primárneho šifrovacieho kľúča a spôsob používania implementovaných kryptologických algoritmov.

2.2.1 Analýza umiestnenia bootovacieho kódu BCVE

Výrobca BCVE v programovej príručke [15] tvrdí, že v prípade zašifrovaného zavádzacieho resp. systémového zväzku sa modifikuje obsah MBR. Zároveň dodáva, že na uloženie bootovacieho kódu nepoužíva rezervované sektory na pevnom disku. Týmito sektormi výrobca myslí oblasť prvých 62 sektorov na starších resp. prvých 2047 sektorov na novších systémoch umiestnených

za bootovacím sektorom. Spomínaná oblasť sa bežne nepoužíva z dôvodu zarovnania oddielov na diskoch. Prvý oddiel na diskoch s 512-bajtovými fyzickými sektormi začína vždy na LBA adrese 63 a na diskoch s 4096-bajtovými fyzickými sektormi na adrese 2048 [7].

Bootovací kód BCVE sa však vzhľadom na jeho funkciu s istotou nezmestí len do 512 bajtov MBR. Zvyšok tohto kódu musí byť uložený niekde na pevnom disku. Dá sa predpokladať, že primárnym účelom programu v MBR bude presunutie tohto kódu do pamäte RAM a jeho následné spustenie. Môžeme sa preto domnievať, že MBR bude obsahovať nejakú informáciu o tom, kde sa na pevnom disku nachádzajú sektory obsahujúce zvyšnú časť bootovacieho kódu. Analýza programu v MBR tak bude kľúčová vzhľadom na ďalší priebeh celkovej analýzy.

Kód v MBR je možné analyzovať buď dynamicky alebo staticky. V prípade dynamickej analýzy sa využíva možnosť krokovania inštrukcií pomocou ladiaceho nástroja (viac v sekcii 2.3.3), pričom počiatočný bod prerušenia sa nastaví na tú adresu v RAM pamäti, na ktorú je nahraný obsah MBR. Na základe informácií o štandardnom zavádzacom procese vieme, že je to vždy fyzická adresa `0x7c00` (viac v podkapitole 2.1). Pri statickej analýze sa strojový kód z prvého sektora na pevnom disku nespúšťa, len sa pomocou disassemblera (sekcia 2.3.2) prevedie do jazyka symbolických inštrukcií.

Výsledkom reverznej analýzy MBR kódu by malo byť odhalenie umiestnenia sektorov s ďalšími časťami bootovacieho kódu BCVE. Z obsahu týchto spomínaných sektorov sa následne zostaví binárny súbor, ktorý bude možné ďalej efektívne staticky analyzovať.

2.2.2 Analýza odvodenia primárneho šifrovacieho kľúča

Na základe dostupných informácií z [15] vieme, že BCVE ukladá za normálnych okolností šifrovací kľúč potrebný na šifrovanie resp. dešifrovanie sektorov určitého zväzku na tom istom zväzku v šifrovanej podobe. Tento kľúč bude vo zvyšku tejto práce označovaný ako primárny šifrovací kľúč. Primárny kľúč je zašifrovaný pomocou šifrovacieho kľúča odvodeného zo zadaného prístupového hesla. Kľúč odvodený z hesla bude ďalej označovaný ako sekundárny šifrovací kľúč. Informácie o tom, kde je na danom zväzku uložený zašifrovaný primárny kľúč, výrobca BCVE neuvádza. Zároveň nie je známy spôsob, akým sa z prístupového hesla odvodí sekundárny šifrovací kľúč, a ani to, aký algoritmus sa používa na zašifrovanie primárneho kľúča.

Po zapnutí počítača a spustení bootovacieho kódu BCVE sa objaví obrazovka zavádzača BCVE s výzvou o zadanie prístupového hesla (obrázok 1.3). Potvrdením zadaného prístupového hesla začínajú všetky zaujímavé operácie z pohľadu bezpečnostnej analýzy bootovacieho kódu (počnúc od operácie overovania korektnosti prístupového hesla až po operáciu odvodenia primárneho šifrovacieho kľúča). Toto miesto je vhodným vstupným bodom na analýzu ope-

rácií vykonávaných so zadaným prístupovým heslom a ďalších nadväzujúcich akcií.

Analyzovaný bootovací kód beží v režime reálnych adries (real mode). Z toho vyplýva, že na prácu s klávesnicou a displejom počítača sa pravdepodobne budú používať služby BIOS-u. Video služby (napríklad výpis znaku na displej) má na starosti prerušenie `int 10h` a za kontrolu klávesnice je zodpovedné prerušenie `int 16h`. Vyhľadanie týchto prerušení v kóde nám umožní odhaliť miesto s inštrukciami zodpovednými za výpis výzvy na zadanie prístupového hesla a samotné zadávanie hesla.

Ďalší spôsob, akým v reálnom režime môže prebiehať vykresľovanie znakov na displej počítača alebo práca s klávesnicou, spočíva v priamom prístupe k danému hardvérovému zariadeniu. Vykresľovanie znakov je tak možné realizovať prostredníctvom zápisu znaku na príslušné miesto videopamäte namapovanej do RAM pamäte. Odchytávanie stlačených kláves sa dá zas zabezpečiť využitím hardvérového prerušenia `int 9`.

2.2.3 Analýza procesu šifrovania a dešifrovania

Šifrovacie algoritmy používané programom BCVE sú verejne známe (AES, RC6, Serpent, Twofish). Výrobca v [15] taktiež popisuje niektoré parametre týchto blokových šifier a zároveň spomína aplikovaný operačný mód – XTS. Na úspešné otestovanie správnosti implementovaných kryptologických primitív však nie sú dostupné všetky potrebné informácie týkajúce sa používania tohto operačného módu. Autor BCVE vo svojej softvérovej príručke neuvádza veľkosť tweak kľúča a presný spôsob, na základe ktorého sa počíta tweak hodnota.

Dešifrovanie systémových súborov a súborov potrebných na úspešné bootovanie musí prebehnúť pred zavedením operačného systému. Z toho vyplýva, že kód vykonávajúci túto operáciu musí byť súčasťou bootovacieho kódu BCVE. Zároveň sa dá predpokladať, že BCVE po každom zapnutí počítača nedešifruje celý zavádzací/systémový zväzok, ale len tú časť, len tie sektory, ktoré sú aktuálne potrebné. Po zadaní prístupového hesla, jeho úspešnom overení a odvodení primárneho šifrovacieho kľúča by tak mali nasledovať inštrukcie programu, ktorých úlohou je zavedenie nejakého mechanizmu zabezpečujúceho automatické dešifrovanie (šifrovanie) čítaných (zapisovaných) sektorov zavádzacieho/systémového zväzku.

Mechanizmus automatického šifrovania/dešifrovania sektorov je vhodným vstupným bodom analýzy kryptologických primitív. Tento mechanizmus bude najskôr založený na využívaní služieb BIOS-u. Pri čítaní resp. zápise sektorov na disk sa vyvoláva prerušenie `int 13h`. Nahradením štandardnej rutiny obsluhujúcej toto prerušenie inou rutinou v tabuľke prerušení je možné zachytávať každú operáciu čítania/zápisu sektora. V prípade BCVE by to mala byť rutina, ktorá zariadi šifrovanie resp. dešifrovanie sektora príslušnou blokovou

šifrou. Chýbajúce informácie týkajúce sa nastavenia operačného módu XTS tak bude možné získať analýzou tejto časti kódu.

2.3 Nástroje na analýzu

Pri reverznej analýze bootovacej časti aplikácie BCVE bolo použitých viacero užitočných nástrojov. Táto podkapitola sa venuje jednotlivým kľúčovým programov, ktoré boli pri analýze aplikované. Medzi spomínané nástroje možno zaradiť: virtualizačný softvér, ladiaci program (debugger), disassembler a editor binárnych súborov (hex editor).

2.3.1 Virtualizačný program

Pracovať pri reverznej analýze bootovacieho kódu so skutočným pevným diskom na fyzickom počítači by bolo príliš zložité, časovo neefektívne a zároveň by mohla byť ohrozená stabilita operačného systému a konzistencia uložených dát na disku. Z týchto dôvodov je lepšie pristúpiť k inému, výhodnejšiemu riešeniu. Ako najvhodnejšie sa ponúka vykonávanie reverznej analýzy vo virtuálnom prostredí.

Virtualizácia prináša do procesu analýzy mnoho výhod. Namiesto reálneho pevného disku sa bude pracovať s binárnym diskovým obrazom. Obraz disku resp. jeho časť vo forme binárneho súboru je možné následne efektívne spracovávať aj inými nástrojmi, ako sú ladiaci program alebo disassembler. Ďalšou z hlavných výhod používania virtualizačného nástroja je možnosť zachytávať a ukladať si stav virtuálneho stroja v určitom momente (takzvaný snapshot systému). Do takto vytvorených „záložných“ stavov sa je možné neskôr opätovne vrátiť, čo má veľký prínos na priebeh analýzy v prípade výskytu nejakej chyby systému alebo v prípade narušenia konzistencie dát na pevnom disku.

Na trhu je dostupných viacero kvalitných virtualizačných nástrojov. Medzi najlepšie virtualizačné aplikácie určené pre platformu Microsoft Windows podľa [36] patria:

- VirtualBox [29]
- VMware Workstation Pro [38]
- VMware Workstation Player [37]
- Windows Virtual PC [25]

Vzhľadom na množstvo skúseností a možnosť slobodného používania bol z týchto štyroch virtualizačných programov na účely tejto diplomovej práce zvolený nástroj VirtualBox.

Výber operačného systému pre klientský počítač bol podmienený požiadavkami programu BCVE. Ten podporuje výhradne platformu Microsoft Windows. Zostávalo tak uvažovať len o výbere verzie tohto operačného systému.

Pri tomto výbere sa brala do úvahy požiadavka danej verzie na minimálnu potrebnú diskovú kapacitu. Z tohto dôvodu bola zvolená najstaršia verzia systému Windows, ktorá je softvérom BCVE podporovaná – Windows XP. Táto verzia v porovnaní so všetkými nasledujúcimi verziami operačného systému Windows vyžaduje najmenej miesta na pevnom disku, čo zjednodušuje následnú manipuláciu s binárnymi diskovými obrazmi. Dá sa zároveň predpokladať, že rôzne verzie operačného systému Windows nemajú v prípade použitia metódy BIOS-MBR výrazný vplyv na vlastnosti skúmaného bootovacieho kódu.

Pri analýze sa používal klientský počítač s virtuálnym diskom rozdeleným na dva zväzky. Prvý z nich slúžil ako zavádzací/systémový zväzok a bola naň nainštalovaná vyššie spomenutá verzia operačného systému Windows. Druhý predstavoval bežný dátový zväzok. Dáta na obidvoch zväzkoch boli z dôvodu zjednodušenia analýzy organizované pomocou diskového súborového systému FAT32.

2.3.2 Disassembler

Disassembler je počítačový program, ktorý umožňuje prekladať strojový kód procesora do pre človeka zrozumiteľnejšieho jazyka symbolických inštrukcií (assembly language). Táto operácia je inverzná k činnosti vykonávanej prekladačom (assemblerom). Z pohľadu reverznej analýzy ide o najdôležitejší nástroj. Používa sa k statickej analýze (analýze aplikovanej na kód, ktorý sa nespúšťa) preložených programov, u ktorých nie je dostupný ich pôvodný zdrojový kód vo vyššom programovacom jazyku. Pre potreby tejto práce bol zvolený disassembler IDA Pro Free [10].

IDA Pro Free je interaktívny disassembler, ktorý sa môže zdarma používať na nekomerčné účely. Tento program na rozdiel od klasického disassemblera ponúka okrem základnej funkcionality prekladu mnoho ďalších užitočných funkcionalít. Automaticky vykonáva rôzne analýzy kódu, umožňuje používanie krížových referencií alebo vykonávanie rôznych zaujímavých úprav (napríklad premenovanie premenných alebo vkladanie komentárov) vygenerovaného jazyka symbolických inštrukcií.

2.3.3 Ladiaci program

Ladiaci program (debugger), ako to už z jeho názvu vyplýva, je softvérový nástroj, ktorý sa bežne používa na testovanie a ladenie iných programov. Pri reverznej analýze však má ladiaci program iné využitie – je to základný nástroj dynamickej analýzy kódu (analýza aplikovaná na spustený kód), ktorý slúži na pochopenie toho, ako v skutočnosti pracuje neznámy preložený program. Medzi kľúčové funkcionality ladiaceho nástroja na úrovni strojového kódu patrí: prevod strojového kódu do jazyka symbolických inštrukcií (disassem-

bler), krokovanie inštrukcií a možnosť pozastavenia behu programu na zvolenej inštrukcii pomocou bodu prerušenia (breakpoint).

Bootovacia časť programu BCVE beží z dôvodu spätnej kompatibility v režime reálnych adries. V reálnom režime sa vykonáva strojový kód, ktorý je typický pre 16-bitové procesory. V súčasnej dobe sú však už viac rozšírené 32-bitové a 64-bitové procesory. Aj z tohto dôvodu sa už nástrojom na ladenie 16-bitového kódu nevenuje až taká pozornosť, a preto je možné na trhu nájsť len veľmi málo takýchto programov. Väčšinou ani užívateľovi neposkytujú taký komfort, funkcionality a užívateľské rozhranie (zväčša sú realizované len formou textovej konzoly) ako obdobné programy podporujúce inštrukčné sady novších architektúr procesorov. Jedným z mála voľne dostupných ladiacich nástrojov podporujúcich 16-bitovú architektúru procesorov je program Bochs [31].

Nástroj Bochs je v skutočnosti vlastne emulátor IBM PC kompatibilných počítačov. Okrem procesoru podporuje aj emuláciu operačnej pamäte, pevného disku, displeja, BIOS-u a bežných periférií. Vďaka tomu sa tento program okrem emulácie často používa aj na ladenie operačných systémov. Bochs umožňuje prostredníctvom textovej konzoly alebo jednoduchého grafického rozhrania ladenie celého bootovacieho procesu, čo má pre potreby tejto diplomovej práce podstatný význam.

2.3.4 Editor binárnych súborov

Editor binárnych súborov alebo hex editor je počítačový program, ktorý slúži na prehliadanie a manipuláciu súborov na úrovni jednotlivých bajtov. Takýto nástroj poslúži v priebehu reverznej analýzy pri práci s analyzovanými a testovanými binárnymi diskovými obrazmi. Na trhu existuje mnoho voľne dostupných hex editorov. Jedným z nich je aj HxD editor [11]. Tento nástroj prináša okrem klasických funkcionalít bežného editora binárnych súborov aj možnosť prehliadať a meniť dáta na aktuálnom pevnom disku.

Kapitola 3

Výsledky reverznej analýzy

V tejto kapitole budú popísané výsledky reverznej analýzy bootovacej časti programu BCVE získané pomocou postupov a metód popísaných v predchádzajúcej kapitole. Na začiatku sa kapitola zaoberá lokáciou jednotlivých častí bootovacieho kódu programu BCVE na pevnom disku. Potom nasleduje pasáž kapitoly, ktorá sa venuje procesu odvodenia primárneho šifrovacieho kľúča zo zadaného prístupového hesla a nakoniec sú diskutované výsledky analýzy procesu šifrovania a dešifrovanie sektorov pomocou implementovaných kryptologických primitív.

Obsah tejto kapitoly nezachádza do úplných detailov výsledkov analýzy. Čitateľ však môže nájsť ďalšie podrobnosti reverznej analýzy bootovacieho kódu v `idb` súboroch na priloženom CD. Na tomto CD sú umiestnené dva súbory. Prvý zo súborov (`aes.idb`) je detailne okomentovaný a dokumentuje výsledky reverznej analýzy bootovacej časti programu BCVE pri zašifrovanom zavádzacom resp. systémovom zväzku pomocou šifrovacieho algoritmu AES (Rijndael). Tento súbor je zároveň zdrojom všetkých výpisov inštrukcií nájdených v tejto kapitole. Čitateľ tak môže jednotlivé sekvencie inštrukcií jednoducho nájsť na zobrazených adresách v spomínanom `idb` súbore. Keďže analýzou bolo zistené, že po zašifrovaní zväzku sa súčasťou bootovacieho kódu BCVE stane len ten algoritmus blokovej šifry, ktorým bol daný zavádzací/systémový zväzok zašifrovaný, tak bola ešte kvôli overeniu získaných výsledkov vykonaná reverzná analýza bootovacieho kódu v prípade zväzku zašifrovanom pomocou šifry RC6 (súbor `rc6.idb`). Vzhľadom na to, že tieto dva analyzované súbory sa významovo odlišovali len implementovanými šifrovacími algoritmami, už nebola ďalej realizovaná analýza bootovacieho kódu pri zašifrovanom zavádzacom resp. systémovom zväzku blokovými šiframi Serpent a Twofish.

V obidvoch spomínaných prípadoch sa reverzná analýza týka bootovacieho kódu pri zašifrovanom systémovom zväzku, ktorý je zároveň zavádzacím, pričom nebola použitá žiadna z rozširujúcich funkcionalít (bezobslužný reštart, dvojstupňová autentifikácia a pod.). Zašifrovanie bežného dátového zväzku analyzovaného obrazu disku nemalo vplyv na činnosť bootovacieho kódu.

3.1 Výsledky analýzy umiestnenia bootovacieho kódu BCVE

Táto podkapitola prináša výsledky analýzy zameranej na odhalenie umiestnenia bootovacieho kódu BCVE na disku a detailov jeho následného zavedenia do operačnej pamäte. Konkrétne sa zaoberá štruktúrou hlavného spúšťacieho záznamu BCVE a účelom jeho primárneho zavádzača. Ďalej možno v tejto podkapitole nájsť rozloženie bootovacej časti programu v RAM pamäti a v závere sú diskutované bezpečnostné aspekty kódu v MBR.

3.1.1 Štruktúra hlavného spúšťacieho záznamu programu BCVE

Program BIOS-u po zapnutí resp. reštarte počítača so zašifrovaným zavádzacím resp. systémovým zväzkom skopíruje bootovací sektor pevného disku (MBR) na fyzickú adresu `0x7c00` operačnej pamäte. Na túto adresu sa uloží sektor hlavného spúšťacieho záznamu programu BCVE, ktorého štruktúra je zmapovaná v tabuľke 3.1. Táto tabuľka obsahuje len tie položky, ktoré sú dôležité z pohľadu reverznej analýzy umiestnenia bootovacej časti programu BCVE na disku.

Adresa	Počet bajtov	Popis
`0x000`	`0x1a0`	Kód primárneho zavádzača programu BCVE
	⋮	
`0x1a3`	`0x2`	Počiatočný zapisovací offset segmentu 0x7800
`0x1a5`	`0x2`	Počiatočný zapisovací offset segmentu 0x6800
	⋮	
`0x1a8`	`0x4`	Identifikátor disku s BCVE kódom
	⋮	
`0x1ae`	`0x6`	Adresa konfiguračného záznamu BCVE
	⋮	
`0x1fe`	`0x2`	Boot sektor signatúra (`0x55, 0xaa`)

Tabuľka 3.1: Štruktúra hlavného spúšťacieho záznamu BCVE

Interpretácia jednotlivých položiek tabuľky 3.1:

- **Kód primárneho zavádzača programu BCVE**
  Primárnou úlohou kódu BCVE umiestneného v MBR je zavedenie ďalších sektorov bootovacej časti programu BCVE z pevného disku do operačnej pamäte. Podrobne je funkcionalita tohto kódu popísaná v nasledujúcej sekcii.

- **Počiatočný zapisovací offset segmentu `0x7800`**
  Počiatočný zapisovací offset predstavuje počiatočný offset oblasti v rámci segmentu `0x7800` operačnej pamäte v režime reálnych adries, do ktorej je z pevného disku zapísaná ďalšia časť bootovacieho programu. Analyzované hlavné spúšťacie záznamy programu BCVE mali na mieste tejto položky uloženú hodnotu `0x200`.

- **Počiatočný zapisovací offset segmentu `0x6800`**
  Tento offset má rovnaký účel ako offset popísaný v predchádzajúcom bode, len s tým rozdielom, že platí v rámci segmentu `0x6800` operačnej pamäte. V analyzovaných prípadoch to bol nulový offset.

- **Identifikátor disku s BCVE kódom**
  4-bajtová číselná hodnota jednoznačne identifikujúca pevný disk s ďalšou časťou bootovacieho programu BCVE.

- **Adresa konfiguračného záznamu BCVE**
  Položka adresy konfiguračného záznamu BCVE v MBR udáva pozíciu sektora obsahujúceho záznam s konfiguračnými dátami na zavádzacom alebo systémovom zväzku. Konkrétne sa v hlavnom spúšťacom zázname nachádza 6 najmenej významných bajtov z celkových 8 bajtov LBA adresy (2 najvýznamnejšie bajty adresy sú pri čítaní sektorov z disku vždy nastavené na hodnotu 0). Konfiguračný záznam je z pohľadu analýzy umiestnenia bootovacieho kódu BCVE veľmi dôležitý, pretože okrem iného obsahuje informácie o pozíciach ďalších sektorov s kódom a dátami BCVE. Podrobnosti o ňom je možné nájsť v sekcii 3.2.1. V analyzovaných prípadoch položka adresy konfiguračného záznamu BCVE obsahovala LBA adresu 63.

3.1.2 Kód hlavného spúšťacieho záznamu programu BCVE

Ako už bolo vyššie spomenuté, kód uložený v MBR je zodpovedný primárne za zavedenie celého bootovacieho kódu BCVE do pamäte RAM. Tento kód sa začne vykonávať z fyzickej adresy `0x7c00` hneď po tom, čo skončí program BIOS-u. Prvé inštrukcie BCVE MBR programu zachytáva obrázok 3.1. Zobrazená sekvencia inštrukcií kopíruje 512 bajtov z MBR do segmentu `0x7800`, do ktorého vzápätí aj predáva riadenie. V tomto segmente sa bude nakoniec nachádzať väčšina kódu vykonávajúca najdôležitejšie operácie bootovacej časti aplikácie BCVE.

Spustením kódu umiestneného v segmente `0x7800` sa začne realizovať postupnosť inštrukcií, ktorú popisuje algoritmus 3.1. Na začiatku sa na počiatočný zapisovací offset segmentu `0x7800` uloží konfiguračný záznam. Jeho súčasťou je akási „mapa“ zaznamenávajúca pozície sektorov so zvyšným bootovacím kódom a dátami BCVE na pevnom disku. Jednotlivé položky sektorovej

```
7C00h:0000h     mov     ax, 7800h
7C00h:0003h     mov     ss, ax
7C00h:0005h     xor     sp, sp
7C00h:0007h     sti
7C00h:0008h     mov     es, ax
7C00h:000Ah     xor     di, di
7C00h:000Ch     push    0
7C00h:000Fh     pop     ds
7C00h:0010h     mov     si, 7C00h
7C00h:0013h     mov     cx, 100h
7C00h:0016h     cld
7C00h:0017h     rep movsw           ; presun obsahu MBR
7C00h:0019h     push    7800h
7C00h:001Ch     push    20h
7C00h:001Fh     retf                ; predanie riadenia do
                                    ; segmentu 7800h
```

Obr. 3.1: Prvé inštrukcie programu BCVE v MBR

mapy sú zložené z dvoch častí, zo 6 najmenej významných bajtov počiatočnej LBA adresy sektorovej oblasti a 2-bajtovej hodnoty reprezentujúcej počet sektorov danej oblasti. Každá z položiek mapy sektorov je postupne spracovaná a sektory, ktoré sa týkajú danej položky, sú skopírované do RAM pamäte na vopred stanovenú pozíciu. Nakoniec sa po zavedení všetkých týchto sekto-

Algoritmus 3.1 Zavedenie BCVE kódu z pevného disku do pamäte RAM

```
1:  function moveBCVECodeFromDiskToRAM
2:      sectorCounter ← 0
3:      address ← MBR.configurationRecordAddress
4:      configurationRecord ← readSectorFromDisk(address)
5:      writeSectorToRAM(configurationRecord, sectorCounter)
6:      sectorCounter ← sectorCounter + 1
7:      sectorMap ← configurationRecord.sectorMap
8:      for all sectorMapEntry ∈ sectorMap do
9:          numberOfSectors ← sectorMapEntry.numberOfSectors
10:         address ← sectorMapEntry.startAddress
11:         for i ← 1, numberOfSectors do
12:             sector ← readSectorFromDisk(address)
13:             writeSectorToRAM(sector, sectorCounter)
14:             sectorCounter ← sectorCounter + 1
15:             address ← address + 1
16:         end for
17:     end for
18: end function
```

rov do pamäte predáva riadenie do oblasti, ktorá realizuje hlavné funkcionality BCVE (overovanie hesla, odvodzovanie šifrovacieho kľúča, atď.).

Účelom funkcie readSectorFromDisk v znázornenom algoritme je čítanie sektora z disku obsahujúceho BCVE kód. Funkcia writeSectorToRAM prečítaný sektor zapisuje do RAM pamäte na adresu odvodenú na základe počtu dosiaľ zapísaných sektorov.

3.1.3 Umiestnenie bootovacieho kódu BCVE na pevnom disku

Rozloženie bootovacej časti programu BCVE nemá na pevnom disku stálu pozíciu. Fixnú pozíciu má jedine časť BCVE kódu umiestnená v MBR. Pozícia konfiguračného záznamu s informáciami o umiestnení kódu BCVE závisí na hodnote adresy uvedenej v hlavnom spúšťacom zázname a líši sa v závislosti na presnej pozícii zašifrovaného zavádzacieho/systémového zväzku na pevnom disku. Zároveň platí, že položky mapy sektorov v konfiguračnom zázname sa môžu v čase meniť. K týmto zmenám môže dôjsť napríklad po viacnásobnom šifrovaní/dešifrovaní zväzku alebo pri zmene šifrovacieho algoritmu. Ďalej sa dá predpokladať, že nie je fixne daná ani veľkosť bootovacieho kódu BCVE. Tá taktiež závisí na hodnotách sektorovej mapy a je možné, že sa bude meniť pri rôznych verziách a nastaveniach BCVE (pri využívaní rôznych funkcionalít).

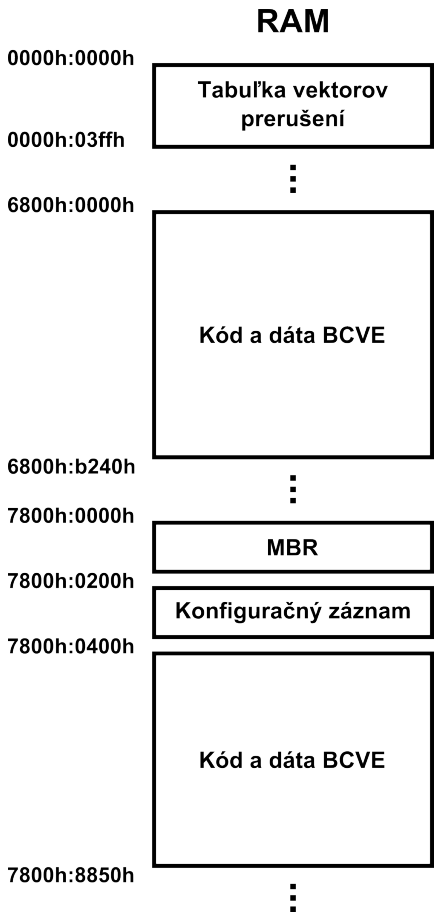
V prípade analyzovanej verzie 3.70.22 programu BCVE a pri danej konfigurácii (šifrovanie zavádzacieho/systémového zväzku bez využitia dvojstupňovej autentifikácie, bezobslúžneho reštartu a presunu šifrovacieho kľúča na externé zariadenie) bootovacia časť celkovo zaberala 80528 bajtov.

3.1.4 Umiestnenie bootovacieho kódu BCVE v RAM pamäti

Umiestnenie bootovacieho kódu analyzovanej verzie programu BCVE pri daných nastaveniach v operačnej pamäti názorne zachytáva obrázok 3.2. Na základe tohto rozloženia kódu boli prostredníctvom pomocného programu (čitateľ môže jeho zdrojový kód nájsť v priečinku `analysis source` na priloženom CD) vytvorené binárne súbory obsahujúce oblasť operačnej pamäte od adresy `6800h:0000h` po adresu `7800h:8850h` (celkovo 100432 bajtov). Tieto súbory s kódom BCVE už boli ďalej analyzované primárne staticky. Práve z nich vznikli `idb` súbory umiestnené na priloženom CD.

3.1.5 Ochrana MBR programu BCVE pred prepísaním

Vzhľadom na to, že kód umiestnený v MBR má ako jediný z bootovacej časti programu BCVE na pevnom disku stále rovnakú pozíciu, tak sa môže stať potenciálnym cieľom pre útočníka. Ten sa môže pokúsiť v snahe získať prístup k zašifrovaným dátam prepísať časť kódu primárneho zavádzača v MBR svojím programom, ktorý mu umožní od oprávneného užívateľa „ukradnúť“



Obr. 3.2: Rozloženie programu BCVE v RAM pamäti

jeho prístupové heslo alebo odvodený primárny šifrovací kľúč (tzv. Evil Maid Attack [33]).

V snahe overiť zabezpečenie MBR pred modifikáciou boli vykonané pokusy o prepísanie kódu primárneho zavádzača alebo inicializovaných položiek MBR (adresa konfiguračného záznamu a pod.) na úrovni operačného systému pomocou diskového editora. Všetky tieto pokusy boli neúspešné. Obsah MBR nebol vôbec zmenený alebo bol vzápätí opravený do jeho pôvodnej podoby. Dá sa predpokladať, že táto ochrana je zabezpečená na úrovni operačného systému pomocou ovládača (filter driver) filtrujúceho požiadavky na zápis dát na pevný disk. Táto skutočnosť nebola bližšie preskúmaná, ale možno skonštatovať, že po predaní riadenia operačnému systému je kód v MBR úspešne ochránený pred nesofistikovanými pokusmi o jeho modifikáciu.

Ďalšia ochrana pred prepísaním MBR je súčasťou bootovacieho kódu, konkrétne sa nachádza v BCVE rutine prerušenia `int 13h` zavádzajúcej po autentifikácii mechanizmus automatického šifrovania a dešifrovania. Táto ochrana

má opodstatnenie napríklad v prípade, ak by útočník umiestnil svoj kód prepisujúci MBR do VBR aktívneho diskového oddiela. Spôsob, akým je táto ochrana v rutine prerušenia realizovaná, môže čitateľ nájsť v sekcii 3.3.3.

Obr. 3.3: Dialógové okno upozorňujúce užívateľa na modifikáciu MBR

Zostávalo tak overiť, či BCVE implementuje na úrovni bootovacieho kódu nejakú ochranu pred prepísaním alebo detekciu modifikácie MBR predtým, ako dôjde k autentifikácii užívateľa. Reverznou analýzou bootovacej časti programu BCVE bolo zistené, že v prípade z pohľadu zavedenia operačného systému nedeštruktívnej modifikácie dát prvej polovice MBR (prvých 256 bajtov) je užívateľ hneď po nabootovaní upozornený prostredníctvom dialógového okna (obrázok 3.3), že je pravdepodobne obeťou nejakého útoku. Avšak pri prepísaní druhej polovice MBR sa spomínané dialógové okno nezobrazí. Príčinou je kód realizujúci detekciu modifikácie znázornený na obrázku 3.4. Implementovaný kód porovnáva obsah prvého sektora na disku s obsahom referenčného hlavného spúšťacieho záznamu BCVE pomocou inštrukcie `rep cmpsw`. Táto inštrukcia používa ako parameter register `cx`, ktorého hodnota udáva počet porovnávaných 2-bajtových slov. Register je pred porovnaním nastavený na hodnotu `0x80`, a preto je porovnávaných len prvých 256 bajtov MBR. Vzhľadom na to, že druhá časť záznamu obsahuje tiež kód a zároveň premenné, ktoré môžu ovplyvniť beh programu, tak spôsob, akým je realizované porovnávanie, vytvára príležitosť na zneužite potenciálnym útočníkom. Ten môže využiť na zavedenie svojho kódu programom BCVE v MBR implementovanú funkciu na načítanie sektorov z disku tak, že prepíše hodnotu adresy konfiguračného záznamu na LBA adresu oblasti s jeho vlastným kódom a následne modifikáciou inštrukcií pôvodného MBR zabezpečí predanie riadenia do danej oblasti. Na umiestnenie svojho kódu môže využiť napríklad bežne nepoužívané a programom BCVE nešifrované sektory za hlavným spúšťacím záznamom.

Vyššie opísaný útok je možné jednoducho realizovať v situácii, keď je porušená nutná podmienka na dosiahnutie informačnej bezpečnosti, a to fyzická bezpečnosť počítača. Fyzický prístup útočníka k zašifrovanému pevnému disku mu umožňuje pripojiť disk k inému zariadeniu a pomocou neho následne prepísať obsah MBR. Avšak ak má útočník fyzický prístup k počítaču, tak je ne-

relevantné, či BCVE nejakým spôsobom chráni dáta uložené na MBR, pretože v takom prípade je útočník schopný prepísať MBR kódom, ktorý dokáže získať heslo oprávneného užívateľa, potom obnoviť pôvodný obsah MBR a následne spustiť originálny kód primárneho zavádzača programu BCVE. No v prípade, ak sa útočníkovi podarí prepísať druhú polovicu MBR bez nutnosti fyzického prístupu, tak program BCVE na to chybne žiadnym spôsobom nezareaguje.

```
7800h:24F6h     mov     si, 0F800h    ; offset s aktuálnym MBR
7800h:24F9h     mov     di, 3766h     ; offset s referenčným MBR
7800h:24FCh     mov     cx, 80h       ; 80h ~ 128
7800h:24FFh     repe cmpsw            ; porovnávanie
7800h:2501h     cmp     cx, 0
7800h:2504h     jz      short loc_7A50B
7800h:2506h     mov     ds:byte_7B0CE, 0AAh
                                      ; nastavenie príznaku
                                      ; prepísania MBR
```

Obr. 3.4: Kód BCVE realizujúci detekciu modifikácie MBR

3.2 Výsledky analýzy odvodenia primárneho šifrovacieho kľúča

Táto podkapitola sa venuje výsledkom analýzy procesu odvodenia primárneho šifrovacieho kľúča z užívateľom zadaného prístupového hesla. Prvá časť sa zaoberá popisom a umiestnením informácií potrebných pri odvodzovaní. Následne je podrobne popísaný proces derivácie primárneho kľúča a nakoniec je z bezpečnostného hľadiska diskutovaná práca s prístupovým heslom a šifrovacími kľúčmi. Keďže sa väčšina tejto podkapitoly zameriava na priebeh derivácie primárneho kľúča pri nastaveniach uvedených v úvode kapitoly 3, tak sú v jej závere ešte diskutované aj spôsoby odvodenia primárneho kľúča pri uplatnení špecifických funkcionalít (dvojstupňová autentifikácia, bezobslužný reštart a pod.).

3.2.1 Konfiguračný záznam

Konfiguračný záznam je umiestnený na začiatku príslušného zašifrovaného zväzku (v jeho 1. sektore). Tento záznam reprezentuje 512-bajtovú oblasť, ktorá obsahuje dôležité informácie o umiestnení bootovacieho kódu a dát programu BCVE na pevnom disku. Ďalej sa v ňom nachádzajú informácie potrebné na odvodenie primárneho kľúča z hlavného prístupového hesla a informácie nevyhnutné na úspešné dešifrovanie sektorov príslušného zväzku. Štruktúru konfiguračného záznamu názorne zachytáva tabuľka 3.2.

<table>
<tr><th></th><th>0x0</th><th>0x1</th><th>0x2</th><th>0x3</th><th>0x4</th><th>0x5</th><th>0x6</th><th>0x7</th><th>0x8</th><th>0x9</th><th>0xa</th><th>0xb</th><th>0xc</th><th>0xd</th><th>0xe</th><th>0xf</th></tr>
<tr><td>0x000</td><td colspan="16">BestCrypt Volume Encryption sektor signatúra</td></tr>
<tr><td>0x010</td><td colspan="16">⋮</td></tr>
<tr><td>0x020</td><td colspan="16" rowspan="2">Zašifrovaný dátový šifrovací kľúč</td></tr>
<tr><td>0x030</td></tr>
<tr><td>0x040</td><td colspan="16" rowspan="2">Zašifrovaný tweak šifrovací kľúč</td></tr>
<tr><td>0x050</td></tr>
<tr><td>0x060</td><td colspan="16" rowspan="2">Zašifrovaný kontrolný haš primárneho šifrovacieho kľúča</td></tr>
<tr><td>0x070</td></tr>
<tr><td>0x080</td><td colspan="16" rowspan="5">Mapa sektorov bootovacieho kódu a dát BCVE</td></tr>
<tr><td>0x090</td></tr>
<tr><td>0x0a0</td></tr>
<tr><td>0x0b0</td></tr>
<tr><td>0x0c0</td></tr>
<tr><td>0x0d0</td><td colspan="16" rowspan="4"></td></tr>
<tr><td>0x0e0</td></tr>
<tr><td>0x0f0</td></tr>
<tr><td>0x100</td><td colspan="16">⋮</td></tr>
<tr><td>⋮</td><td colspan="16">⋮</td></tr>
<tr><td>0x1e0</td><td colspan="8"></td><td>Operačný mód</td><td>Šifra</td><td colspan="6">Kryptografická soľ</td></tr>
<tr><td>0x1f0</td><td colspan="16">⋮</td></tr>
</table>

Tabuľka 3.2: Štruktúra konfiguračného záznamu BCVE

Interpretácia jednotlivých položiek tabuľky popisujúcej konfiguračný záznam:

- **BestCrypt Volume Encryption sektor signatúra**
  Sektory na disku patriace programu BCVE sú odlíšené od ostatných sektorov tým, že na začiatku obsahujú špecifickú sekvenciu 16 bajtov, nazvime ju BCVE sektor signatúra. Táto signatúra je tvorená nasledujúcimi bajtami:

  ```
  0xdd 0xa2 0x6a 0x7e 0x3a 0x59 0xff 0x45

  0x3e 0x35 0x0a 0x44 0xbc 0xb4 0xcd 0xd5
  ```

- **Zašifrovaný dátový šifrovací kľúč**
  Táto oblasť konfiguračného záznamu predstavuje 256-bitový zašifrovaný dátový šifrovací kľúč, ktorý sa používa pri operačnom móde XTS na šifrovanie blokov blokovej šifry.

- **Zašifrovaný tweak šifrovací kľúč**
  Položka tabuľky reprezentuje 256-bitový zašifrovaný tweak šifrovací kľúč používaný pri operačnom móde XTS na odvodenie tweak hodnoty.

- **Zašifrovaný kontrolný haš primárneho šifrovacieho kľúča**
  256-bitový zašifrovaný kontrolný haš vytvorený pomocou hašovacej funkcie SHA-256 je haš dešifrovaného primárneho šifrovacieho kľúča resp. dvoch po sebe zreťazených XTS kľúčov (dátového a tweak kľúča). Používa sa pri overovaní správnosti odvodeného primárneho kľúča.

- **Mapa sektorov bootovacieho kódu a dát BCVE**
  Mapa sektorov zaznamenáva umiestnenie sektorov s bootovacím kódom a dátami programu BCVE na pevnom disku. Štruktúra jednotlivých položiek sektorovej mapy a spôsob ich použitia pri zavádzaní kódu do pamäte je popísaný v sekcii 3.1.2.

- **Operačný mód**
  Na základe štyroch najmenej významných bitov tejto 1-bajtovej položky sa pravdepodobne rozhoduje o tom, aký operačný mód sa bude používať pri šifrovaní/dešifrovaní sektorov. V prípade, že je nastavený (má hodnotu 1) 3. bit sprava, tak sa použije operačný mód XTS. Ak žiaden z týchto bitov nie je nastavený, tak sa uplatní mód CBC. Význam ďalších kombinácii bitov na základe reverznej analýzy nie je možné určiť. Pravdepodobne niektorá z kombinácií niekedy patrila módu LRW, pretože staršie verzie BCVE ho používali ako predchodcu módu XTS [17].

- **Šifra**
  Táto 1-bajtová položka špecifikuje algoritmus použitý na šifrovanie daného zväzku. Podporované hodnoty:

`0x8` – AES
`0x9` – Twofish
`0xa` – Serpent
`0xb` – RC6

- **Kryptografická soľ**
  Kryptografická soľ je tvorená sekvenciou 8 bajtov. Používa sa v rámci procesu odvodzovania sekundárneho kľúča z prihlasovacieho hesla. Viac v sekcii 3.2.2.1.

3.2.2 Odvodenie primárneho šifrovacieho kľúča

Po zobrazení výzvy užívateľovi na zadanie prístupového hesla (obrázok 1.3) a jeho následnom zadaní sa začína proces odvodzovania primárneho šifrovacieho kľúča. Celkový priebeh odvodzovania kľúča z hlavného prístupového hesla popisuje algoritmus 3.2. Výsledok tohto procesu závisí na hodnotách položiek v konfiguračnom zázname a hodnote zadaného hesla. Konkrétne sa pri odvodzovaní používajú tieto položky konfiguračného záznamu: kryptografická soľ, zašifrovaný dátový kľúč, zašifrovaný tweak kľúč, zašifrovaný haš primárneho kľúča (za sebou zreťazená trojica – dátový kľúč, tweak kľúč a haš primárneho kľúča – bude vo zvyšku textu označovaná ako blok primárneho kľúča). V prípade, že užívateľ zadal nesprávne heslo, tak sa celý proces zopakuje. BCVE si zaznamenáva počet týchto neúspešných pokusov a po úspešnom nabootovaní ho oznámi oprávnenému užívateľovi.

Algoritmus 3.2 Odvodenie primárneho šifrovacieho kľúča

```
1:  function derivePrimaryKey(password)
2:      salt ← configurationRecord.salt
3:      dataKey ← configurationRecord.dataKey
4:      tweakKey ← configurationRecord.tweakKey
5:      hash ← configurationRecord.primaryKeyHash
6:      keyBlock ← dataKey|tweakKey|hash            ▷ | – concatenation
7:      secondaryKey ← deriveSecondaryKey(password, salt)
8:      keyBlock ← decryptKeyBlock(secondaryKey, keyBlock)
9:      result ←verifyPrimaryKey(keyBlock)
10:     return result
11: end function
```

Proces derivácie primárneho šifrovacieho kľúča pozostáva z týchto troch hlavných častí:

- Odvodenie sekundárneho šifrovacieho kľúča (DeriveSecondaryKey)
- Dešifrovanie bloku primárneho šifrovacieho kľúča (DecryptKeyBlock)

- Overenie primárneho šifrovacieho kľúča (VerifyPrimaryKey)

3.2.2.1 Odvodenie sekundárneho šifrovacieho kľúča

Sekundárny šifrovací kľúč sa používa na dešifrovanie zašifrovaného bloku primárneho kľúča. Výrobca v programovej príručke BCVE neuvádza veľkosť tohto kľúča ani spôsob jeho odvodenia. Na základe reverznej analýzy bolo zistené, že vstupmi na vygenerovanie sekundárneho šifrovacieho kľúča sú zadané prístupové heslo a kryptografická soľ umiestnená v konfiguračnom zázname. Priebeh procesu derivácie popisuje algoritmus 3.3.

Výsledkom tohto algoritmu je 32 bajtov dlhý haš, ktorý sa neskôr používa zároveň vo význame sekundárneho šifrovacieho kľúča. Tento haš vznikol pomocou hašovacej funkcie SHA-256 tak, že sa na jej vstup dal programom BCVE vytvorený 65536-bajtový blok dát pozostávajúci z viacnásobného zreťazenia 8-bajtovej kryptografickej soli a zadaného prístupového hesla. Detailný postup, akým sa tento blok zostavuje, môže čitateľ nájsť v algoritme 3.3.

Z priebehu derivácie kľúča je zrejmé, že program BCVE pri odvodzovaní šifrovacieho kľúča nepoužíva nejaký zo štandardizovaných postupov (napríklad funkciu PBKDF2 [18]). Namiesto toho implementuje vlastné riešenie, čo môže byť potenciálne nebezpečné. Použitý algoritmus sa napríklad žiadnym spôsobom nesnaží spomaliť prípadný útok hrubou silou, dajme tomu aplikovaním opakovaného hašovania.

Algoritmus 3.3 Odvodenie sekundárneho šifrovacieho kľúča

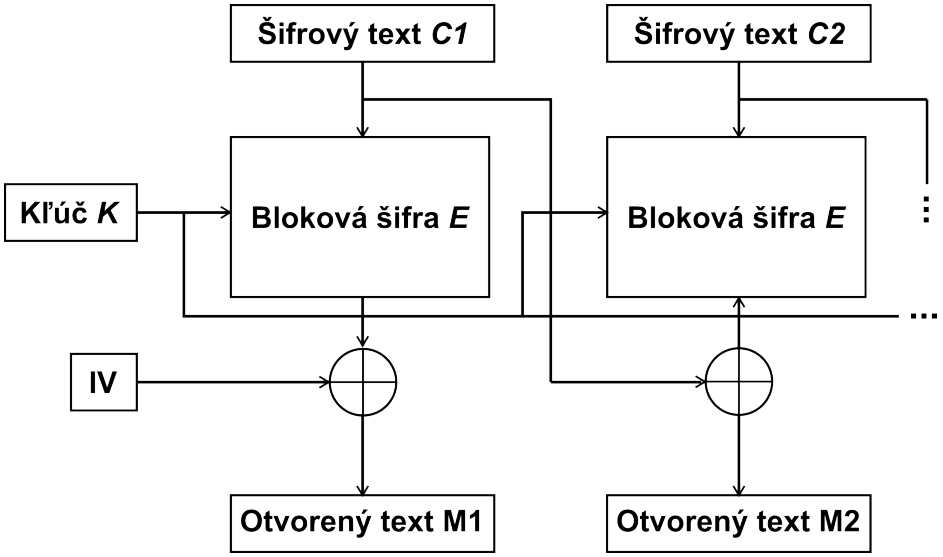
```
1:  function deriveSecondaryKey(password, salt)
2:      maxBufferLen ← 10000_16
3:      bufferLen ← 0
4:      passwordLen ← getLength(password)
5:      saltLen ← getLength(salt)
6:      while maxBufferLen − bufferLen ≥ passwordLen + saltLen do
7:          buffer ← buffer|salt|password                ▷ | – concatenation
8:          bufferLen ← bufferLen + passwordLen + saltLen
9:      end while
10:     if maxBufferLen − bufferLen ≥ saltLen then
11:         buffer ← buffer|salt
12:         bufferLen ← bufferLen + saltLen
13:     else
14:         buffer ← buffer|substr(salt, maxBufferLen − bufferLen)
15:         bufferLen ← maxBufferLen
16:     end if
17:     buffer ← buffer|substr(password, maxBufferLen − bufferLen)
18:     secondaryKey ←sha-256(buffer)
19:     return secondaryKey
20: end function
```

GetLength v zobrazenom algoritme reprezentuje funkciu, ktorej výstupom je dĺžka príslušnej dátovej oblasti. Funkcia substr vracia podreťazec prvého argumentu, ktorý začína od počiatku pôvodného reťazca a jeho celkovú dĺžku udáva druhý argument. Funkcia SHA-256 hašuje vstupný argument hašovacím algoritmom SHA-256.

3.2.2.2 Dešifrovanie bloku primárneho šifrovacieho kľúča

Po odvodení sekundárneho šifrovacieho kľúča sa tento kľúč použije na dešifrovanie bloku primárneho šifrovacieho kľúča. Na jeho dešifrovanie sa využíva tá istá bloková šifra, ktorá bola užívateľom zvolená na zašifrovanie daného zväzku. Rozdiel je však v aplikovanom operačnom móde. V tomto prípade BCVE používa operačný mód CBC.

Pri aplikácii operačného módu CBC sa 96 bajtov zašifrovaného bloku primárneho šifrovacieho kľúča na začiatku rozdelí do šiestich 16-bajtových blokov. Následné dešifrovanie týchto blokov bude prebiehať podľa schémy, ktorú znázorňuje obrázok 3.5. Z tohto obrázka vyplýva, že na začiatku dešifrovania je zároveň potrebný inicializačný vektor (IV). Reverznou analýzou bolo zistené, že sa v tomto prípade ako inicializačný vektor používa vektor 16 nulových bajtov. Avšak z bezpečnostného hľadiska by bolo určite vhodnejšie, ak by tento vektor nemal stálu hodnotu, ale pozostával z premenných bajtov, ktorých hodnoty by si program niekde udržiaval (napríklad v konfiguračnom zázname).



Obr. 3.5: Schéma dešifrovania pomocou operačného módu CBC

3.2.2.3 Overenie primárneho šifrovacieho kľúča

Overovanie získaného primárneho kľúča prebieha podľa algoritmu 3.4. Na začiatku sa pomocou hašovacej funkcie SHA-256 vypočíta haš zo za sebou zreťazených hodnôt dátového a tweak šifrovacieho kľúča z dešifrovaného bloku primárneho kľúča a následne sa porovná s hodnotou kontrolného haša primárneho šifrovacieho kľúča. V prípade, že dôjde k zhode týchto dvoch hodnôt, tak zadané prístupové heslo je správne, daný užívateľ sa úspešne autentifikoval, a tak môže nasledovať zavedenie automatického mechanizmu na dešifrovanie príslušného zväzku. V opačnom prípade sa proces odvodzovania primárneho šifrovacieho kľúča znova opakuje počnúc od zadania prístupového hesla užívateľom.

Algoritmus 3.4 Overenie primárneho šifrovacieho kľúča

```
1: function verifyPrimaryKey(decryptedKeyBlock)
2:     dataKey ← decryptedKeyBlock.dataKey
3:     tweakKey ← decryptedKeyBlock.tweakKey
4:     hash ← decryptedKeyBlock.primaryKeyHash
5:     if hash = SHA-256(dataKey|tweakKey) then      ▷ | – concatenation
6:         result ← true
7:     else
8:         result ← false
9:     end if
10:    return result
11: end function
```

3.2.2.4 Odvodenie primárneho šifrovacieho kľúča z doplnkových prístupových hesiel

V prvej kapitole tejto diplomovej práce bolo spomenuté, že BCVE umožňuje užívateľovi nastaviť okrem hlavného hesla aj niekoľko doplnkových hesiel. Výrobca v softvérovej príručke BCVE neuvádza ich presný počet, avšak reverznou analýzou bootovacieho kódu bolo zistené, že je možné vytvoriť maximálne štyri takéto doplnkové prístupové heslá. Proces odvodzovania primárneho šifrovacieho kľúča z týchto doplnkových hesiel sa zhoduje s procesom týkajúcim sa hlavného hesla, s tým rozdielom, že zašifrované bloky primárneho kľúča súvisiace s jednotlivými doplnkovými heslami neboli uložené v konfiguračnom zázname, ale nachádzali sa v analyzovaných prípadoch v operačnej pamäti na týchto adresách:

7800h:0a20h
7800h:0a84h
7800h:0ae8h
7800h:0b4ch

3.2.3 Bezpečnostné aspekty práce s prístupovým heslom a so šifrovacími kľúčmi

Táto sekcia diplomovej práce diskutuje výsledky reverznej analýzy zameranej na možnosti získania hesla alebo šifrovacieho kľúča z operačnej pamäte neoprávneným užívateľom. V prvej časti sa zaoberá bezpečnostnými aspektami používania prístupového hesla a v druhej časti sa venuje potenciálnym slabinám práce s odvodenými šifrovacími kľúčmi.

3.2.3.1 Ochrana prístupového hesla

Na základe reverznej analýzy bolo zistené, že užívateľ si môže nastaviť maximálne 128 znakov dlhé heslo. Oblasť operačnej pamäte, do ktorej sa ukladajú jednotlivé znaky prístupového hesla, sa v analyzovaných prípadoch nachádzala na adrese `7800h:2ca0h`. Celá táto oblasť (128 bajtov) je pred každým novým zadávaním hesla vynulovaná (obrázok 3.6). Takáto inicializácia pamäte pre prístupové heslo sa zároveň uskutoční aj po zadaní správneho hesla a následnom úspešnom odvodení a overení primárneho šifrovacieho kľúča.

Rovnako pred každým zadávaním hesla a po úspešnej autentifikácii užívateľa dochádza aj k inicializácii vyrovnávacej pamäte klávesnice (sekvenciu inštrukcií realizujúcu túto inicializáciu možno nájsť na obrázku 3.6). Tá začína

```
7800h:200Eh     push    40h
7800h:2011h     pop     es
7800h:2012h     mov     di, 1Eh ; 40h:1Eh (32 bajtov)
                                ; vyrovnávacia pamäť klávesnice
7800h:2015h     mov     cx, 20h ; počet bajtov
7800h:2018h     mov     al, 20h ; znak medzery - ' '
7800h:201Ah     rep stosb
7800h:201Ch     push    cs
7800h:201Dh     pop     es
7800h:201Eh     xor     ax, ax
7800h:2020h     mov     cx, 40h ; počet 2-bajtových slov
7800h:2023h     mov     di, offset byte_7ACA0
                                ; pamäť na prístupové heslo
7800h:2026h     rep stosw
```

Obr. 3.6: Inicializácia vyrovnávacej pamäte klávesnice a pamäte pre prístupové heslo

na adrese `0040h:001eh` RAM pamäte a celkovo ju tvorí 32 bajtov. Vyrovnávacia pamäť klávesnice sa nenuluje, ale zapĺňa znakom medzery (hexadecimálne ide o bajt `0x20`).

Ďalšie miesto bootovacieho kódu BCVE, kde sa pracuje s prístupovým heslom, je pri odvodzovaní sekundárneho šifrovacieho kľúča. Po vykonaní transformácie hesla na kľúč v pamäti nezostávajú žiadne pozostatky po prístupovom hesle. Dá sa tak skonštatovať, že bootovací kód BCVE pracuje s prístupovým heslom bezpečne a po autentifikácii užívateľa sa zadané heslo v pamäti už vyskytovať nebude.

3.2.3.2 Ochrana šifrovacích kľúčov

Oblastiam operačnej pamäte používaných na uchovávanie sekundárneho resp. primárneho šifrovacieho kľúča a z nich expandovaných kľúčov určených na šifrovanie alebo dešifrovanie bootovací kód BCVE takú pozornosť ako v prípade prístupového hesla nedáva. Po presune daného kľúča na iné miesto alebo v prípade, že kľúč z danej oblasti už viac nebude používaný, sa tieto oblasti pred predaním riadenia operačnému systému žiadnym spôsobom nepremažú. Vzniká tak teoretická možnosť na získanie kľúča z RAM pamäte.

Jednou z možností, ako by to mohol útočník dosiahnuť, je takzvaný cold boot útok. Tento útok je založený na tom, že na niektorých systémoch sa pri štarte nemaže celý obsah RAM pamäte. V prípade, že útočník má fyzický prístup k danému počítaču, tak môže za určitých podmienok z operačnej pamäte šifrovacie kľúče získať. Viac o tomto type útoku v [8].

3.2.4 Odvodenie primárneho šifrovacieho kľúča pri použití rozširujúcich funkcionalít

V prípadoch, pri ktorých sa používajú rozširujúce funkcionality programu BCVE, ako sú presun šifrovacieho kľúča na externé zariadenie, dvojstupňová autentifikácia alebo bezobslužný reštart, sa proces odvodzovania primárneho šifrovacieho kľúča môže odlišovať od spôsobu popísaného v predchádzajúcich častiach tejto podkapitoly.

Pri využívaní bezobslužného reštartu nedochádza k procesu odvodzovania primárneho šifrovacieho kľúča. Bootovací kód programu BCVE získava primárny šifrovací kľúč prostredníctvom volaní TCG (Trusted Computing Group) funkcií priamo z registra TPM.

Spôsoby derivácie primárneho kľúča v prípadoch dvojstupňovej autentifikácie pomocou bezpečnostného tokena alebo bežného výmenného média a pri presune šifrovacieho kľúča na externé zariadenie neboli podrobne preskúmané. Dá sa však predpokladať, že v týchto prípadoch bude odvodzovanie kľúča zhodné s tým, ako to bolo popísané v tejto podkapitole, len s tým rozdielom, že dáta potrebné pri derivácii budú uložené na spomínaných zariadeniach.

3.3 Výsledky analýzy procesu šifrovania a dešifrovania

Táto podkapitola sa zaoberá výsledkami reverznej analýzy tých častí bootovacieho kódu BCVE, ktoré sú zodpovedné za šifrovanie resp. dešifrovanie sektorov zavádzacieho/systémového zväzku. Na začiatku je popísaný spôsob, akým si program BCVE udržiava informácie o zašifrovaných oblastiach na pevných diskoch. Potom nasleduje časť kapitoly, ktorá sa venuje mechanizmu umožňujúcemu automatické šifrovanie resp. dešifrovanie. Nakoniec je popísaná rutina programu BCVE nahradzujúca originálnu rutinu služby `int 13h` a proces samotného šifrovania a dešifrovania.

3.3.1 Diskový záznam BCVE

Program BCVE si na disku pre každý zašifrovaný zväzok udržiava informácie o zašifrovaných oblastiach vo forme akýchsi záznamov. Tieto záznamy budú pre potreby tejto práce ďalej označované ako diskové záznamy BCVE.

Diskový záznam BCVE tvorí 64 bajtov. Organizáciu jeho dôležitých položiek popisuje tabuľka 3.3.

Adresa	Počet bajtov	Popis
0x00	0x10	BestCrypt Volume Encryption sektor signatúra
0x10	0x6	Minimálna LBA adresa
0x16	0x6	Maximálna LBA adresa
	⋮	
0x22	0x4	Identifikátor disku
	⋮	
0x35	0x1	Identifikátor diskovej jednotky
	⋮	
0x38	0x6	LBA adresa nahradeného sektora

Tabuľka 3.3: Štruktúra diskového záznamu BCVE

Interpretácia jednotlivých položiek tabuľky 3.3:

- **BestCrypt Volume Encryption sektor signatúra**
  Sekvencia bajtov označujúca oblasť pamäte obsahujúcu dáta programu BCVE. Podrobnosti v sekcii 3.2.1.
- **Minimálna LBA adresa**
  6 najmenej významných bajtov LBA adresy počiatočného sektora zašifrovanej oblasti disku. Táto adresa zdola vymedzuje šifrovanú oblasť disku.

- **Maximálna LBA adresa**
  6 najmenej významných bajtov LBA adresy posledného sektora zašifrovanej oblasti disku. Táto adresa vymedzuje zašifrovanú oblasť disku zhora.

- **Identifikátor disku**
  32-bitová číselná hodnota predstavujúca jedinečný identifikátor pamäťového zariadenia.

- **Identifikátor diskovej jednotky**
  1-bajtová číselná hodnota identifikujúca diskovú jednotku v rámci systému.

- **LBA adresa nahradeného sektora**
  6 najmenej významných bajtov LBA adresy sektora, ktorého obsah bol po zašifrovaní nahradený. Originálny zašifrovaný sektor z danej adresy bol presunutý na programom BCVE preddefinované miesto. V analyzovaných prípadoch na túto adresu program BCVE umiestňoval konfiguračný záznam daného zväzku.

Umiestnenie diskových záznamov BCVE daného zväzku určuje sektorová mapa z príslušného konfiguračného záznamu. Konkrétne 2 záznamy sú umiestnené v 3. sektore sektorovej mapy na offsetoch `0x00` a `0x40` a ďalšie 2 je možné nájsť v 7. sektore na rovnakých miestach. Avšak nie vždy všetky diskové záznamy obsahujú zmysluplné hodnoty. Po zavedení do operačnej pamäte sa v bootovacom kóde diskové záznamy zavádzacieho/systémového zväzku nachádzali na týchto adresách:

```
7800h:0800h
7800h:0840h
7800h:1000h
7800h:1040h
```

3.3.2 Zavedenie automatického šifrovania a dešifrovania

Po úspešnej autentifikácii užívateľa bootovacia časť programu BCVE zavádza mechanizmus automatického šifrovania a dešifrovania. Tento mechanizmus je založený na prepísaní originálnej adresy služby BIOS-u `int 13h` uloženej v tabuľke vektorov prerušení. Táto služba má na starosti diskové operácie vrátane čítania a zápisu jednotlivých sektorov na pevný disk. Nahradením originálnej rutiny program BCVE zabezpečuje, že pri čítaní sektora z disku dôjde k jeho dešifrovaniu a pri zápise sektora na disk k jeho zašifrovaniu. Postup, akým tento automatický proces zavádza kód BCVE, popisuje algoritmus 3.5.

Algoritmus 3.5 Zavedenie automatického šifrovania a dešifrovania

```
1: function setAutomaticEncryptionDecryption()
2:     interruptHandlerMemoryAddress ←allocateHandlerMemory()
3:     memoryCopy(interruptHandlerMemoryAddress, handlerBCVE)
4:     setInterruptVector(interruptHandlerMemoryAddress)
5: end function
```

Úlohou funkcie allocateHandlerMemory je alokácia permanentne voľného miesta (miesto, ktoré nebude neskôr prepísané) v rámci operačnej pamäte určeného pre novú rutinu nahradzujúcu originálnu službu BIOS-u (účelom tejto nahradzujúcej rutiny sa detailne venuje sekcia 3.3.3). Bootovací kód BCVE to realizuje pomocou sekvencie inštrukcií zobrazených na obrázku 3.7. V dátovej oblasti BIOS-u na adrese `0000h:0413h` sa nachádza 2-bajtové číslo určujúce kapacitu konvenčnej pamäte v kilobajtoch. Konvenčná pamäť predstavuje zvyčajne oblasť prvých 640 kilobajtov pamäte RAM (viac o tejto oblasti pamäte v [20]). Znížením kapacity konvenčnej pamäte bootovací kód BCVE rezervuje miesto, na ktoré potom skopíruje svoju rutinu prerušenia (v algoritme 3.5 funkcia memoryCopy), a adresu tohto miesta následne zapíše na príslušné miesto tabuľky vektorov prerušení odpovedajúce službe `int 13h` (funkcia setInterruptVector).

```
7800h:29A9h     push    0
7800h:29ACh     pop     ds
7800h:29ADh     mov     bp, 413h
7800h:29B0h     mov     ax, ds:[bp+0]   ; Kapacita konvečnej pamäte
7800h:29B4h     sub     ax, 26          ; Zmenšenie kapacity pamäte
7800h:29B7h     mov     ds:[bp+0], ax   ; Nastavenie novej kapacity
7800h:29BBh     sub     ax, 9
7800h:29BEh     shl     ax, 6           ; Adresa segmentu pre novú
                                        ; rutinu prerušenia int 13h
```

Obr. 3.7: Alokácia pamäte pre novú rutinu prerušenia int 13h

3.3.3 BCVE rutina prerušenia int 13h

Vzhľadom na to, že rutina programu BCVE zastupuje originálne prerušenie BIOS-u, tak pre bezchybné fungovanie systému musí vykonávať aj všetky jeho pôvodné funkcionality. Prerušenie `int 13h` má na starosti viacero diskových operácií [5]. Výber operácie, ktorá sa má v danej chvíli aplikovať, závisí na hodnote vstupného parametra prerušenia uloženej v registry `ah`. Z pohľadu BCVE rutiny zaujímavé hodnoty sú uvedené v tabuľke 3.4:

Pri vyvolaní prerušenia so vstupným parametrom určujúcim jednu z operácií v tabuľke 3.4 sa v prípade potreby aplikuje šifrovanie resp. dešifrovanie sektorov. Spôsob, akým sa to realizuje, je možné vidieť v zápise algoritmu 3.6

Parameter prerušenia	Disková operácia
`0x2`	Čítanie sektora/sektorov z disku
`0x42`	Rozšírené čítanie sektora/sektorov z disku
`0x3`	Zápis sektora/sektorov na disk
`0x43`	Rozšírený zápis sektora/sektorov na disk

Tabuľka 3.4: Programom BCVE pozmenené diskové operácie

Algoritmus 3.6 BCVE rutina prerušenia int 13h

```
1: function BCVEInterruptHandler13h(parameters)
2:     function ← parameters.function
3:     if function = 2 or function = 42_16 then          ▷ read sector/sectors
4:         if isEncryptedDrive(parameters.driveNumber) then
5:             originalInterruptHandler13h(parameters)
6:             encryptDecryptSectors(parameters, 1)    ▷ 1 – decryption
7:         else
8:             originalInterruptHandler13h(parameters)
9:         end if
10:    else if function = 3 or function = 43_16 then ▷ write sector/sectors
11:        if isEncryptedDrive(parameters.driveNumber) then
12:            encryptDecryptSectors(parameters, 0)    ▷ 0 – encryption
13:            originalInterruptHandler13h(parameters)
14:        else if parameters.driveNumber ≠ BCVEmbrDriveNumber or
                   parameters.addressLBA ≠ 0 then
15:            originalInterruptHandler13h(parameters)
16:        end if
17:    else
18:        originalInterruptHandler13h(parameters)
19:    end if
20: end function
```

popisujúceho riadiaci tok BCVE rutiny. Program BCVE si o zašifrovaných zväzkoch minimálne na úrovni bootovacieho kódu vytvára záznamy obsahujúce základné informácie týkajúce sa šifrovaných oblastí na príslušnom disku – diskové záznamy BCVE (popis organizácie dát na nich obsiahnutých môže čitateľ nájsť v sekcii 3.3.1). To, či je nutné sektory z daného disku šifrovať/dešifrovať, program BCVE určí na základe porovnania 1-bajtovej hodnoty parametra prerušenia uloženej v registry `dl`, hodnoty identifikujúcej diskovú jednotku v rámci systému, s hodnotami z existujúcich diskových záznamov BCVE (v zobrazenom algoritme to realizuje funkcia isEncryptedDrive). Priebehu samotného šifrovania a dešifrovania (funkcia encryptDecryptSectors) sa podrobne venuje nasledujúca sekcia.

Súčasťou BCVE rutiny prerušenia `int 13h` je aj ochrana hlavného spúšťa-

cieho záznamu s BCVE zavádzacím kódom pred modifikáciou. Ak by útočník chcel po zavedení automatického šifrovania/dešifrovania prepísať bootovací sektor, tak to BCVE rutina odchytí a takýto zápis sektora nevykoná (v algoritme riadok 14).

V prípade vyvolania iných diskových operácií alebo v prípade, keď daný disk nie je potrebné šifrovať resp. dešifrovať, rutina programu BCVE zavolá len pôvodnú službu BIOS-u s danými vstupnými argumentami prerušenia. Funkcia originalInterruptHandler13h v algoritme 3.6 reprezentuje originálne prerušenie `int 13h`.

3.3.4 Šifrovanie a dešifrovanie sektorov

Rutina prerušenia BCVE implementuje šifrovanie a dešifrovanie jednotlivých sektorov pevného disku podľa algoritmu 3.7. Vzhľadom na to, že nie všetky sektory na disku sú zašifrované, tak sa na základe LBA adresy daného sektora a informácií z diskových záznamom BCVE najprv rozhoduje o tom, či dôjde k samotnému šifrovaniu resp. dešifrovaniu príslušného sektora (v algoritme funkcia needToEncryptDecryptSector).

K šifrovaniu alebo dešifrovaniu sektora nedochádza v týchto dvoch prípadoch:

- Sektor obsahuje na začiatku BCVE sektor signatúru.
- Sektor sa nachádza mimo rozsahu LBA adries uvedených v diskových záznamoch BCVE ohraničujúcich šifrované oblasti (minimálna a maximálna LBA adresa).

Ďalší hraničný prípad nastáva, ak dochádza k čítaniu sektora z adresy, ktorá je v diskovom zázname BCVE označená ako LBA adresa nahradeného sektora (sekcia 3.3.1). Sektor na danej adrese pevného disku obsahuje dáta programu BCVE, a preto je pôvodný sektor uložený na inom mieste. Pri zavádzaní bootovacieho kódu BCVE do pamäte RAM sa z disku v šifrovanej podobe skopíruje aj originálny sektor prislúchajúci danej LBA adrese. Po úspešnej autentifikácii užívateľa sa tento sektor v operačnej pamäti dešifruje. V prípade, že následne dôjde k požiadavke jeho čítania, tak bootovacia časť programu BCVE vráti namiesto sektora fyzicky uloženého na žiadanej LBA adrese dešifrovaný sektor z operačnej pamäte.

Z výsledkov analýzy odvodzovania primárneho šifrovacieho kľúča vyplýva, že sa pri šifrovaní sektorov používa dvojica 32-bajtových kľúčov. Jedným z nich je tweak kľúč. Ten slúži pri odvodzovaní tweak hodnoty. V softvérovej príručke BCVE nie je uvedený presný spôsob jej derivácie, konkrétne nie je známe, čo presne bootovací kód BCVE používa ako logickú pozíciu sektora (symbol i v schéme šifrovania pomocou XTS – obrázok 1.2). Reverznou analýzou bolo zistené, že táto logická pozícia sektora sa vypočíta tak, že sa od LBA adresy daného sektora odčíta hodnota minimálnej LBA adresy uvedenej v príslušnom

diskovom zázname alebo v prípade nulovej LBA adresy sektora sa použije nulová logická pozícia (riadky 6-10 v algoritme 3.7).

Algoritmus 3.7 Šifrovanie a dešifrovanie sektorov

```
1:  function encryptDecryptSectors(parameters, option)
2:      addressLBA ← parameters.addressLBA
3:      for i ← 1, parameters.numberOfSectors do
4:          if needToEncryptDecryptSector(addressLBA) then
5:              if addressLBA = 0 then
6:                  sectorAddress ← 0
7:              else
8:                  diskRecord ← getDriveRecord(addressLBA)
9:                  sectorAddress ← addressLBA − diskRecord.minimalLBA
10:             end if
11:             if option = 1 then                          ▷ decryption
12:                 decryptXTS(parameters.sectors[i], sectorAddress)
13:             else                                        ▷ encryption
14:                 encryptXTS(parameters.sectors[i], sectorAddress)
15:             end if
16:         end if
17:         addressLBA ← addressLBA + 1
18:     end for
19: end function
```

Funkcia getDriveRecord v zobrazenom algoritme vracia na základe LBA adresy sektora jemu príslušný diskový záznam BCVE. Funkcie encryptXTS a decryptXTS realizujú šifrovanie resp. dešifrovanie sektora príslušnou blokovou šifrou pracujúcou v operačnom móde XTS tak, ako to popisuje podkapitola 1.2.2.

Kapitola 4

Testovanie implementovaných kryptologických primitív

Táto kapitola sa zaoberá testovaním korektnosti implementácie programom BCVE používaných kryptologických algoritmov. V prvých častiach je popísaná implementácia a spôsob použitia pomocného testovacieho programu vytvoreného na základe informácií získaných reverznou analýzou bootovacej časti programu BCVE a v závere kapitoly sú zhodnotené výsledky uskutočnených testov.

4.1 Implementácia testovacieho programu

Otestovanie kryptologických primitív spočíva v porovnaní obsahu sektorov pred zašifrovaním zavádzacieho resp. systémového zväzku programom BCVE s odpovedajúcimi sektormi po zašifrovaní dešifrovanými naprogramovaným testovacím programom. Vytvorený testovací program implementovaný v programovacom jazyku C++ tak predstavuje vlastne program, ktorého primárnou úlohou je dešifrovanie sektorov. Namiesto implementácie jednotlivých kryptologických algoritmov bola v danom programe použitá kryptografická knižnica.

Na začiatku tejto podkapitoly je preto popísaný výber kryptografickej knižnice vhodnej pre potreby tejto práce. Zvyšok podkapitoly sa už venuje vytvorenému programu slúžiacemu na dešifrovanie sektorov zašifrovaných programom BCVE.

4.1.1 Kryptografická knižnica

Na softvérovom trhu možno nájsť viacero voľne dostupných kryptografických knižníc pre jazyk C/C++. Nie každá z nich je však vyhovujúca na otestovanie kryptologických primitív používaných bootovacím kódom programu BCVE. V tomto prípade je potrebná kryptografická knižnica implementujúca nasledujúce algoritmy:

- **AES**: AES-CBC, AES-XTS
- **RC6**: RC6-CBC, RC6-XTS
- **Serpent**: Serpent-CBC, Serpent-XTS
- **Twofish**: Twofish-CBC, Twofish-XTS
- **SHA-256**

Väčšina z kryptografických knižníc bežne umožňuje používanie hašovacej funkcie SHA-256 a vyššie zmienených blokových šifier v operačnom móde CBC. Avšak dostupnosť operačného módu XTS je vzhľadom na jeho malú rozšírenosť problémová. Tento operačný mód nebol vo väčšine prípadov kryptografickými knižnicami vôbec podporovaný alebo bol podporovaný len v spojení s blokovou šifrou AES. Jedna z mála knižníc, ak nie jediná, ktorá v súčasnosti implementuje všetky spomínané kryptologické primitíva, je knižnica Botan. Z tohto dôvodu boli využité pri implementácii programu na dešifrovanie sektorov zašifrovaných pomocou aplikácie BCVE služby tejto kryptografickej knižnice.

4.1.1.1 Kryptografická knižnica Botan

Knižnica Botan je multiplatformová kryptografická knižnica napísaná v programovacom jazyku C++11. Je vydaná pod zjednodušenou BSD licenciou. Botan umožňuje používanie viacerých kryptografických primitív (celkový zoznam môže čitateľ nájsť v [2]) a nástroja príkazového riadka vykonávajúceho rôzne kryptografické operácie.

Knižnica je voľne dostupná na webovej stránke [2]. Na tejto stránke môže čitateľ zároveň nájsť aj detaily týkajúce sa popisu inštalácie knižnice Botan. Inštalácia je riadená pomocou skriptu napísanom v jazyku Python (je potrebný minimálne Python 2.6). Všeobecne na unixových systémoch pozostáva z tejto postupnosti krokov:

```
$ ./configure.py
$ make
$ ./botan-test
$ make install
```

4.1.2 Program na dešifrovanie sektorov

Implementovaný program je založený na postupoch a princípoch nadobudnutých reverznou analýzou bootovacieho kódu BCVE. Tento program umožňuje pri znalosti hlavného prístupového hesla dešifrovanie užívateľom vybraných sektorov zavádzacieho resp. systémového zväzku na zvolenom obraze pevného disku. Program si zo zadaného hesla odvodí primárny šifrovací kľúč, ktorým

dešifruje príslušné sektory. Dešifrovaný obsah nakoniec uloží do výstupného súboru. Avšak v prípade, že zvolený sektor obsahuje signatúru BCVE alebo leží na zavádzacom/systémovom zväzku mimo programom BCVE šifrovanej oblasti, tak k dešifrovaniu nedochádza a program vracia nezmenený obsah daného sektora. Spôsob, akým sa tento program používa, je opísaný v ďalších častiach tejto podkapitoly.

Zdrojový kód programu na dešifrovanie sektorov (`decryptSectors.cpp`) môže čitateľ nájsť na priloženom CD v priečinku `test`.

4.1.2.1 Použitie programu

Skompilovaný program potrebuje od užívateľa k úspešnému spusteniu ešte zadanie piatich vstupných argumentov. Všetky ostatné informácie potrebné k dešifrovaniu sektorov daného zavádzacieho/systémového zväzku program získa samostatne zo zadaného obrazu disku z pozícií odvodených na základe výsledkov vykonanej reverznej analýzy.

Spôsob použitia programu `decryptSectors`:

```
$ ./decryptSectors <disk image> <password> <LBA address>
  <number of sectors> <output file>
```

Význam jednotlivých vstupných parametrov programu na dešifrovanie sektorov:

- **Obraz disku** (`disk image`)
  Argument *obraz disku* predstavuje názov súboru obsahujúceho obraz pevného disku s programom BCVE zašifrovaným zavádzacím resp. systémovým zväzkom.

- **Prístupové heslo** (`password`)
  Tento vstupný parameter reprezentuje hlavné prístupové heslo užívateľa používané na dešifrovanie zavádzacieho/systémového zväzku zašifrovaného pomocou programu BCVE. Zadané heslo môže mať maximálne 128 znakov.

- **LBA adresa** (`LBA address`)
  Tento parameter určuje počiatočnú LBA adresu zašifrovanej sektorovej oblasti. Táto LBA adresa musí byť užívateľom zadaná v hexadecimálnom tvare začínajúc znakmi `0x`. Maximálna povolená hodnota adresy je `0xffffffffffff` (6 bajtov).

- **Počet sektorov** (`number of sectors`)
  Tento argument udáva počet sektorov zašifrovanej oblasti, ktorý má implementovaný program dešifrovať. Minimálny počet je 1 a maximálny 100 000.

- **Výstupný súbor** (`output file`)
  Parameter *výstupný súbor* predstavuje názov súboru, ktorý bude po skončení obsahovať dešifrované sektory.

4.2 Testovanie

Táto podkapitola sa zaoberá samotným procesom testovania kryptologických primitív programu BCVE. Popisuje zvolený postup testovania a potom diskutuje výsledky uskutočnených testov.

4.2.1 Postup testovania

Pri overovaní toho, či program BCVE implementuje v bootovacom kóde používané kryptologické primitíva podľa ich špecifikácií, sa postupovalo tak, že sa najprv na obraze testovaného pevného disku s operačným systémom Microsoft Windows našiel nepoužívaný sektor (sektor obsahujúci samé nulové bajty) patriaci systémovému resp. zavádzaciemu zväzku. Tento sektor sa potom prepísal opakujúcou sa sekvenciou nasledujúcich bajtov:

```
0x00 0x01 0x02 0x03 0x04 0x05 0x06 0x07

0x08 0x09 0x0a 0x0b 0x0c 0x0d 0x0e 0x0f
```

Následne bol zväzok obsahujúci prepísaný sektor zašifrovaný jednou z dostupných blokových šifier. Nakoniec sa pomocou implementovaného programu na dešifrovanie sektorov daný sektor dešifroval a v prípade, že výsledný sektor obsahoval vyššie popísanú opakujúcu sa sekvenciu bajtov, tak test prebehol úspešne.

Testovanie kryptologických primitív prebiehalo šifrovaním zavádzacieho resp. systémového zväzku pri týchto voľbách verzie operačného systému Windows a algoritmu blokovej šifry:

- Windows XP – AES
- Windows XP – RC6
- Windows XP – Serpent
- Windows XP – Twofish
- Windows 7 – AES

4.2.2 Výsledky testovania

V každom testovanom prípade bol výstupom implementovaného programu na dešifrovanie sektorov pôvodný sektor – všetky uskutočnené testy tak môžno hodnotiť ako úspešné (výstupy testovacieho programu môže čitateľ nájsť v priečinku `test results` na priloženom CD). Prvé štyri testy overili správnosť implementácie všetkých programom BCVE používaných kryptologických algoritmov. Keďže reverzná analýza bootovacieho kódu programu BCVE prebiehala len na obraze pevného disku s operačným systémom Windows XP, tak bol uskutočnený ešte jeden test na disku s operačným systémom Windows 7. Úspech tohto testu vypovedá o tom, že verzia operačného systému Windows nemá v prípade použitia metódy BIOS-MBR výrazný vplyv na priebeh skúmanej bootovacej časti programu BCVE.

Záver

Softvér BestCrypt Volume Encryption (BCVE) slúži na šifrovanie dát uložených na pevných diskoch alebo výmenných zariadeniach prostredníctvom šifrovania ich zväzkov vybranými blokovými šiframi v operačnom móde XTS. Po zašifrovaní zavádzacieho resp. systémového zväzku je kód primárneho zavádzača v hlavnom spúšťacom zázname (MBR) nahradený zavádzačom programu BCVE. Jeho primárnou úlohou je zavedenie zvyšného bootovacieho kódu BCVE do operačnej pamäte. Technikami reverzného inžinierstva bol tento bootovací kód zanalyzovaný a bolo zistené, že vykonáva tri základné funkcie, ktorými sú autentifikácia užívateľa pri bootovaní, odvodenie primárneho šifrovacieho kľúča a zavedenie mechanizmu automatického šifrovania a dešifrovania sektorov.

Analyzovaný bootovací kód BCVE zavádza automatické šifrovanie resp. dešifrovanie prostredníctvom nahradenia služby BIOS-u riadiacej diskové operácie svojou vlastnou rutinou odchytávajúcou každé čítanie a zápis sektorov. Táto rutina potrebuje k úspešnému dešifrovaniu sektorov príslušného zašifrovaného zväzku primárny šifrovací kľúč. Ten je súčasťou bloku primárneho kľúča uloženého v šifrovanej podobe v konfiguračnom zázname na danom zväzku alebo externom úložisku. Na jeho dešifrovanie sa používa sekundárny šifrovací kľúč odvodený z prístupového hesla zadaného užívateľom pri autentifikácii hneď po spustení počítača. Bootovacia časť programu BCVE pracuje s týmto prístupovým heslom bezpečne a hneď po odvodení kľúča ho z pamäte vymaže. Avšak, čo sa týka práce so šifrovacími kľúčmi a ochranou BCVE kódu v MBR, boli objavené mierne bezpečnostné nedostatky.

Pre potreby overenia korektnosti programom BCVE v bootovacom kóde implementovaných kryptologických primitív bol zároveň ako súčasť tejto práce na základe výsledkov vykonanej reverznej analýzy vytvorený program umožňujúci dešifrovanie sektorov zavádzacieho resp. systémového zväzku. Prostredníctvom tohto programu boli uskutočnené testy, ktoré v každom testovanom prípade potvrdili, že používané kryptologické algoritmy sú implementované podľa ich špecifikácií.

Možnosti ďalšieho postupu

Na túto diplomovú prácu je možné nadviazať niekoľkými spôsobmi. Prvou možnosťou je reverzná analýza rozširujúcich funkcionalít na úrovni bootovacieho kódu BCVE. Medzi ne patrí napríklad dvojstupňová autentifikácia prostredníctvom bezpečnostného tokenu. Bolo by zaujímavé presne preskúmať, ako sa v jej prípade pracuje so šifrovacím kľúčom.

Ďalšou možnosťou môže byť bezpečnostná analýza hlavnej programovej časti BCVE na úrovni operačného systému, napríklad ovládača, ktorý má na starosti šifrovanie/dešifrovanie.

Literatúra

[1] Anderson, R.; Biham, E.; Knudsen, L.: Serpent: A Proposal for the Advanced Encryption Standard. 1998. Dostupné z: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.35.5585&rep=rep1&type=pdf

[2] Botan: Botan: Webová prezentácia knižnice [online]. [cit.: 2016-04-16]. Dostupné z: http://botan.randombit.net

[3] Citizendium: AES competition [online]. [cit.: 2016-04-06]. Dostupné z: http://en.citizendium.org/wiki/AES_competition

[4] CodePlex: VeraCrypt: Project Description [online]. [cit.: 2016-04-20]. Dostupné z: https://veracrypt.codeplex.com

[5] Computer Tyme: Int 13 [online]. [cit.: 2016-04-02]. Dostupné z: http://www.ctyme.com/intr/int-13.htm

[6] Daemen, J.; Rijmen, V.: *The Design of Rijndael: AES – the Advanced Encryption Standard*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2002, ISBN 3-540-42580-2.

[7] Fischer, W.: Partition Alignment [online]. [cit.: 2016-03-09]. Dostupné z: https://www.thomas-krenn.com/en/wiki/Partition_Alignment

[8] Halderman, J. A.; Schoen, S. D.; Heninger, N.; aj.: Least we remember: Cold boot attacks on encryption keys. In *USENIX Security Symposium*, editace P. C. van Oorschot, USENIX Association, 2008, ISBN 978-1-931971-60-7, s. 45–60.

[9] Hex-Rays: IDA: About [online]. [cit.: 2016-04-27]. Dostupné z: https://www.hex-rays.com/products/ida

[10] Hex-Rays: IDA Support: Freeware Version [online]. [cit.: 2016-03-14]. Dostupné z: https://www.hex-rays.com/products/ida/support/download_freeware.shtml

[11] Hörz, M.: HxD - Freeware Hex Editor and Disk Editor [online]. [cit.: 2016-03-15]. Dostupné z: https://mh-nexus.de/en/hxd

[12] Institute of Electrical and Electronics Engineers (IEEE): Standard for Cryptographic Protection of Data on Block-Oriented Storage Devices. IEEE P1619/D16, 2007. Dostupné z: http://ieeexplore.ieee.org/servlet/opac?punumber=4493431

[13] Intel Corporation: *80386 Programmer's Reference Manual*. Intel Corporation, 1986, ISBN 1-555-12022-9, 512 s.

[14] International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC): Information technology - Trusted Platform Module - Part 1: Overview. ISO/IEC 11889-1:2009, 2009. Dostupné z: http://standards.iso.org/ittf/PubliclyAvailableStandards/c050970_ISO_IEC_11889-1_2009.zip

[15] Jetico Inc.: BestCrypt Volume Encryption Help File. Dostupné z: http://www.jetico.com/web_help/PDF/BCVE.pdf

[16] Jetico Inc.: Mission [online]. [cit.: 2016-02-20]. Dostupné z: http://www.jetico.com/about-jetico/mission-story

[17] Jetico Inc.: New features in version 2 [online]. [cit.: 2016-04-29]. Dostupné z: http://www.jetico.com/web_help/bcve/html/01_introduction/04_new_in_version.htm

[18] Kaliski, B.: PKCS #5: Password-Based Cryptography Specification Version 2.0. RFC 2898, RFC Editor, September 2000. Dostupné z: https://tools.ietf.org/html/rfc2898

[19] Khurshid, U.: [MTE Explains] Differences Between UEFI and BIOS [online]. [cit.: 2016-04-28]. Dostupné z: https://www.maketecheasier.com/differences-between-uefi-and-bios

[20] Kozierok, C. M.: Conventional Memory [online]. [cit.: 2016-03-30]. Dostupné z: http://www.pcguide.com/ref/ram/logicConventional-c.html

[21] Kozierok, C. M.: System Boot Sequence [online]. [cit.: 2016-03-03]. Dostupné z: http://www.pcguide.com/ref/mbsys/bios/bootSequence-c.html

[22] Lee, K.: Advanced Encryption Standard (AES) Selection Process - How Rijndael Won. 2015. Dostupné z: http://www.usna.edu/Users/math/wdj/_files/documents/sm473-capstone/Rinjdael-16WeekFinalDraft-KevinLee.pdf

[23] Liskov, M.; Rivest, R. L.; Wagner, D.: Tweakable Block Ciphers. *Journal of Cryptology*, ročník 24, č. 3, 2010: s. 588–613, ISSN 1432-1378.

[24] Microsoft: BitLocker Drive Encryption Overview [online]. [cit.: 2016-04-20]. Dostupné z: http://windows.microsoft.com/en-us/windows-vista/bitlocker-drive-encryption-overview

[25] Microsoft Corp.: Windows Virtual PC: Webová prezentácia programu [online]. [cit.: 2016-03-13]. Dostupné z: https://www.microsoft.com/en-us/download/details.aspx?id=3702

[26] Microsoft Corporation: Definitions for system volume and boot volume [online]. [cit.: 2016-02-28]. Dostupné z: https://support.microsoft.com/en-us/kb/314470

[27] National Institute of Standards and Technology (NIST): Commerce Department Announces Winner of Global Information Security Competition [online]. [cit.: 2016-04-10]. Dostupné z: http://www.nist.gov/public_affairs/releases/g00-176.cfm

[28] NeoSmart Technologies Inc.: The BIOS/MBR Boot Process [online]. [cit.: 2016-03-08]. Dostupné z: https://neosmart.net/wiki/mbr-boot-process

[29] Oracle Corp.: VirtualBox: Webová prezentácia programu [online]. [cit.: 2016-03-13]. Dostupné z: https://www.virtualbox.org

[30] Rivest, R. L.; Robshaw, M. J. B.; Yin, Y.; aj.: The RC6 Block Cipher. 1998. Dostupné z: http://people.csail.mit.edu/rivest/pubs/RRSY98.pdf

[31] Ruppert, V.: Bochs: The cross platform IA-32 emulator [online]. [cit.: 2016-03-13]. Dostupné z: http://bochs.sourceforge.net

[32] SafeNet Inc.: eToken PRO [online]. [cit.: 2016-02-29]. Dostupné z: http://www.safenet-inc.com/multi-factor-authentication/authenticators/pki-usb-authentication/etoken-pro

[33] Schneier, B.: Evil Maid Attacks on Encrypted Hard Drives [online]. [cit.: 2016-04-28]. Dostupné z: https://www.schneier.com/blog/archives/2009/10/evil_maid_attac.html

[34] Schneier, B.; Kelsey, J.; Hall, C.; aj.: Twofish: A 128-Bit Block Cipher. 1998. Dostupné z: https://www.schneier.com/cryptography/paperfiles/paper-twofish-paper.pdf

[35] Smid, M. E.: AES Issues. Dostupné z: http://csrc.nist.gov/archive/aes/round2/comments/20000523-msmid-2.pdf

[36] TOP Attack reviews and comparison: 2016 Best Virtualization Software Review [online]. [cit.: 2016-03-13]. Dostupné z: http://www.topattack.com/list/best-virtualization-software-for-windows/91

[37] VMware Corp.: VMware Workstation Player: Webová prezentácia programu [online]. [cit.: 2016-03-13]. Dostupné z: http://www.vmware.com/products/player

[38] VMware Corp.: VMware Workstation Pro: Webová prezentácia programu [online]. [cit.: 2016-03-13]. Dostupné z: http://www.vmware.com/products/workstation

Dodatok **A**

Zoznam použitých skratiek

AES Advanced Encryption Standard

BCVE BestCrypt Volume Encryption

BIOS Basic input/output system

CBC Cipher-block chaining

CHS Cylinder-head-sector

DES Data Encryption Standard

EEPROM Electrically erasable programmable read-only memory

HDD Hard disk drive

IDA Interactive disassembler [9]

IEEE Institute of Electrical and Electronics Engineers

LBA Logical block addressing

MBR Master boot record

PBKDF2 Password-Based Key Derivation Function 2

POST Power-on self-test

RAID Redundant array of independent disks

RAM Random-access memory

ROM Read-only memory

TCG Trusted Computing Group

TPM Trusted Platform Module

USB Universal serial bus

VBR Volume boot record

XEX Xor encryption xor

XTS XEX encryption mode with tweak and ciphertext stealing

Dodatok B

Obsah priloženého CD

- `Juraj Hornak - Diplomova Praca (2016)`
  - `readme.txt` — stručný popis obsahu CD
  - `analysis` — reverzná analýza
    - `aes.idb` — súbor s analýzou bootovacieho kódu pri zašifrovanom zavádzacom/systémovom zväzku šifrou AES
    - `rc6.idb` — súbor s analýzou bootovacieho kódu pri zašifrovanom zavádzacom/systémovom zväzku šifrou RC6
  - `src` — zdrojové kódy
    - `impl` — implementácia
      - `analysis source` — zdrojový kód programu zostavujúceho súbor s bootovacím kódom BCVE
      - `test` — zdrojový kód programu použitého na testovanie
    - `thesis` — zdrojová forma práce vo formáte LaTeX
  - `test results` — výsledky testov
  - `text`
    - `DP_Hornak_Juraj_2016.pdf` — text práce vo formáte PDF