NYÍLT FORRÁSKÓDÚ BIZTONSÁGI MENTÉSI ESZKÖZÖK IDŐ- ÉS TÁRHELY-HATÉKONYSÁGÁNAK ÖSSZEHASONLÍTÓ ELEMZÉSE

Kiss Áron[1], Nehéz Károly[2]
[1]PhD hallgató, tanszéki mérnök, [2]egyetemi docens
[1,2]Informatikai Intézet, Gépészmérnöki és Informatikai Kar
[1]kiss.aron@uni-miskolc.hu, [2]karoly.nehez@uni-miskolc.hu

ABSZTRAKT

A digitális korban az információ egyike a legértékesebb vagyontárgyaknak. A rendszeres, megbízható biztonsági másolatok elengedhetetlenek a hardverhibák, emberi tévedések, valamint kibertámadások elleni védelemhez. Habár a deduplikáció és tömörítés előnyei jól ismertek az adattárolás terén, a számos elérhető nyílt forráskódú biztonsági mentési eszköz felveti a kérdést: ezek a hasonló szoftverek mutatnak-e jelentős eltérést futási idő és tárhelyhatékonyság szempontjából? A cikk célja, hogy választ adjon erre a kérdésre, különféle adathalmazokon végzett kísérletek segítségével, négy nyílt forráskódú biztonsági mentési eszköz teljesítményének vizsgálatával, a mért futási idő és tárhelyhatékonyság elemzésével. Kutatásunk segítheti a potenciális felhasználókat tájékozott döntések meghozatalában adatbiztonsági architektúrájukat érintően.

1. BEVEZETÉS

A digitális korban az adat egyike a legértékesebb vagyontárgyaknak, mind a magánszemélyek, mind a szervezetek számára. A technológia fejlődésével a felhalmozott adatvagyon mennyisége és bonyolultsága is folyamatosan nő.

A modern kiberbiztonsági stratégiáknak a proaktív adatvédelmi módszerek mellett napjainkig részét képezi a naprakész, megbízható, és rendszeres biztonsági mentések készítése, hiszen ez hatékony módja többek között a hardverek meghibásodására, az emberi tévedésekre, illetve a célzott kibertámadásokra történő felkészülésnek.

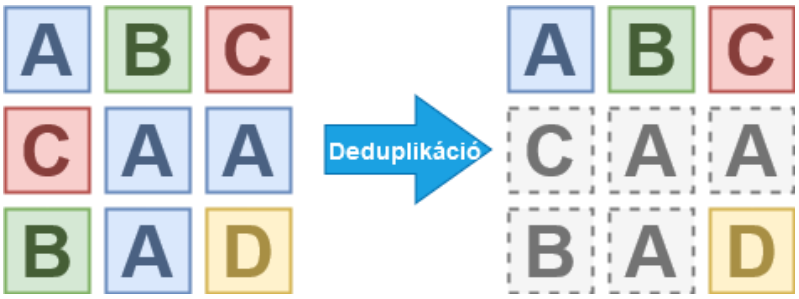
A célzott kibertámadások közül a zsarolóvírusok egyre komolyabb és fokozódóbb veszélyt jelentenek az adatbiztonságra. Ezen kártékony szoftverek titkosítják az érzékeny fájlokat, majd a támadók váltságdíjat követelnek azok visszafejtéséért, sok esetben jelentős anyagi kárt okozva az érintetteknek. [1] magyar kórházakat ért zsarolóvírus támadásokat tanulmányoz, megállapítva, hogy az egyik vizsgált esetben csak a papíralapú nyilvántartás újbóli digitalizációjával voltak helyreállíthatóak az adatok. [2] egy multinacionális kiskereskedelmi vállalat elleni támadást mutat be, mely több országban különböző üzleti szolgáltatások nyújtását napokra ellehetetlenítette. [3] egy olyan hazai IT-disztribútor elleni támadás kezeléséről szolgáltat információkat, mely 3 napig akadályozta az érintett cég üzleti folyamatainak informatikai kiszolgálását.

Ahogy a digitális szolgáltatások fejlődnek, a redundáns vagy elavult adatok tárolása egyre jelentősebb terhet jelent mind tárolási kapacitás, mind költségek szempontjából. A deduplikáció és tömörítés előnyei jól ismertek, a nyílt forráskódú biztonsági mentési eszközök tárháza pedig széles és változatos lehetőségeket kínál. Azonban ez a sokféleség felveti a kérdést: „*Ezek a hasonló célú és architektúrájú eszközök mutatnak-e jelentős eltérést az idő- és tárhelyhatékonyság szempontjá-*

ból?”. Ezen kérdés megválaszolásával célunk, hogy támogassuk a mentési eszközök leendő felhasználóit a tájékozott döntéshozatalban.

2. A BIZTONSÁGI MENTÉS ÁLTALÁNOS LÉPÉSEI

Több modern biztonsági mentési megoldást áttekintve az állapítható meg, hogy a mentési folyamat lépései az egyes eszközök esetében közel azonosak.



1. ábra.
Redundancia csökkentés deduplikációval.

A folyamat első lépése a deduplikáció, mely durva szemcsézettségű tömörítést valósít meg. Lényege, hogy a fájlrendszert kis darabokra, ún. „chunk”-okra bontja, és csak az egyedi chunk-okat tárolja el, az ismétlődő chunk-okra csak referenciát helyez el a mentésben, ezt az 1. ábra szemlélteti.

Fix méretű darabolás (FSC) esetén a chunkok mérete állandó. Ennek implementációja egyszerű, és a folyamat kevésbé számításigényes, azonban érzékeny az ún. „határváltozási problémára”. A 2. ábrán egy olyan fájl látható, ami 1 bájtos darabokra lett bontva. Ha a fájl elejére beillesztünk 1 bit információt, az összes chunk megváltozik, így a lehető legkisebb változtatás hatására a teljes fájl újbóli mentése válik szükségessé.



2. ábra.
Határváltozás fix méretű és tartalom alapú darabolás esetén.

Tartalomtól függő darabolás (Content Dependent Chunking, CDC) során a chunkok mérete két szélsőérték között változhat. A chunkok határait általában Rabin-féle ujjlenyomat számításával határozzák meg, ami számításigényes módszer, azonban a határváltozási problémára kevésbé érzékeny [4, 5]. Napjainkban egyre több optimalizált CDC algoritmus jelenik meg. [6] egy új, a hagyományos módszereknél jelentősen gyorsabb darabolási algoritmust mutat be, Aszimmetrikus Ext-

rémum (AE) néven. [7] egy olyan megközelítést mutat be, amely gyorsabb az AE-nél, miközben megközelíti vagy túlteljesíti a Rabin-alapú CDC-t. [8] egy koszinusz hasonlóságon alapuló fuzzy-interferencia rendszert javasol a hasonló chunkok azonosítására.

Az adathalmaz a deduplikációt követően bit szintű tömörítésen esik át, melyet klasszikus algoritmusokkal (pl. DEFLATE, LZW, ZStandard) végeznek el a deduplikált adathalmazon. Ezzel a chunk-okon belüli redundancia szűrhető ki. A biztonsági mentést gyakran több bérlős környezetben tárolják, mely esetében a szervezet nem rendelkezik megfelelően részletes információval a tárolási infrastruktúráról, nem ismeri például az alkalmazott izolációs megoldásokat és azt sem, hogy fizikailag ki férhet hozzá a tárolt adatokhoz. Emiatt a modern backup szoftverek végpontok közötti titkosítási eljárásokat is alkalmaznak. Az elkészült biztonsági mentés egy repository-ban kerül tárolásra, mely a helyi fájlrendszer mellett számos egyéb helyen tárolható (pl. hálózati meghajtón, FTP vagy SSH szervereken, illetve számos felhőszolgáltató szerverein).

3. VIZSGÁLT ESZKÖZÖK

A vizsgálathoz négy, részben eltérő jellemzőkkel és funkciókkal rendelkező biztonsági mentési eszközt választottunk ki. Ezeket az 1. táblázat mutatja be.

1. táblázat
A vizsgált eszközök fontosabb tulajdonságai ([a] = alapértelmezett).

	borg	**duplicati**	**duplicity**	**restic**
Követők	9 817	9 318	198	21 294
Nyelv	Python, C	C#	Python, C	Go
Platform	Linux Mac	Windows Linux Mac	Linux Mac	Windows Linux Mac
Chunking	CDC[a] FSC	FSC[a]	CDC[a]	CDC[a]
Tömörítés	LZ4[a] ZStandard zlib LZMA	DEFLATE[a] LZMA2	GZip[a]	ZStandard[a]
Titkosítás	AES-256[a]	AES-256[a] GPG	GPG[a]	AES-256[a]
Interfész	CLI	CLI + GUI	CLI	CLI
Licenc	BSD 3-Clause	GNU LGPL	GNU GPL	BSD 2-Clause

A ***borg*** egy UNIX rendszerekre fejlesztett szoftver, mely támogatja a fix és változó méretű darabolást is, illetve 4 különböző tömörítési eljárást biztosít. A ***duplicati*** egy cross-platform szoftver, mely a versenytársaktól eltérően grafikus interfésszel is rendelkezik. A ***duplicity*** egy elsősorban Linux rendszerekre fejlesztett szoftver, mely része egyebek mellett az Ubuntu disztibúciónak is. A mentett adatokat szabványos TAR archívumokban tárolja. A ***restic*** egy Go nyelven fejlesztett, többplatformos szoftver, mely nagy követőtáborral rendelkezik, a CERN felhőalapú kollaborációs platformjában is alkalmazzák, ahol több, mint 37 ezer felhasználó adatainak mentését végzik vele [11].

4. ELVÉGZETT MÉRÉSEK

A kísérletekhez három, különböző mértékben redundáns adathalmazt állítottunk elő, melyek valós felhasználási eseteket szimulálnak. A biztonsági mentést minden adathalmaz-szoftver pár esetében futtattuk.

1. adathalmazként egy *dokumentum archívum* szolgált. Az ilyen archívumokban rendszerint olyan szöveges dokumentumok szerepelnek, melyek részben már tömörítésre kerültek, pl. a DOCX, illetve a PDF formátum LZW, illetve DEFLATE algoritmusokat alkalmaz [9, 10]. Ehhez a Magyar Közlöny és a mellékleteként szolgáló Hivatalos Értesítő 2013 óta megjelent számai kerültek letöltésre.

2. adathalmazunk egy *tiszta rendszerkörnyezet* volt, ehhez a 30 legnépszerűbb Docker-képfájl került letöltésre, melyek különböző szoftverkörnyezetek kezdőállapotát tartalmazzák. Az adathalmazban elsősorban szöveges forráskódok és futtatható bináris állományok szerepelnek, melyek alapértelmezetten nem tömörítettek, ez az 1. adathalmazénál magasabb redundanciát eredményez.

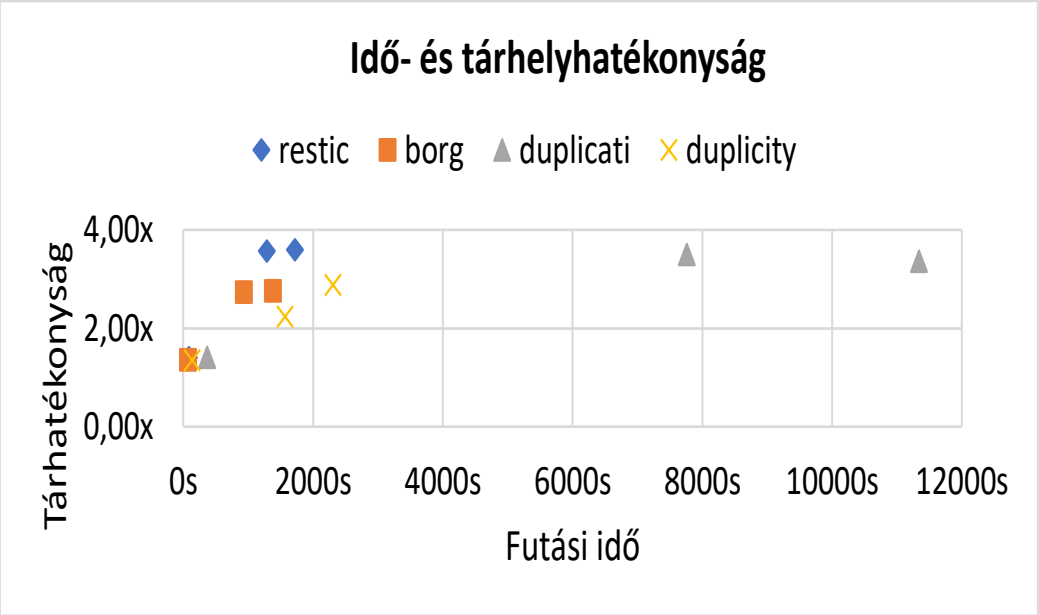
3. adathalmazunk egy *élő rendszerkörnyezet* fájlszintű másolata volt, melyben a rendszer működéséhez szükséges forráskódokon és binárisokon kívül megjelennek többek között felhasználói adatokat tároló adatbázisok, gyorsítótárak és naplófájlok is, a 2. adathalmaznál magasabb redundanciát eredményezve.

$$\textit{Optimalizációs arány} = \textit{Eredeti méret} / \textit{Mentés mérete} \qquad (1)$$

Minden mérést egy virtuális gépen végeztünk el, azonos kezdőállapotból indítva a kísérleteket. A futások során mértük a biztonsági mentés elkészítéséhez szükséges időt, illetve az ún. „optimalizációs arányt”, melyet az (1) egyenlet ír le. Minél magasabb ez az érték, annál hatékonyabb az alkalmazott tömörítési eljárás.

5. EREDMÉNYEK KIÉRTÉKELÉSE

A 3. ábra az idő- és tárhatékonyság közötti kompromisszumokat mutatja be. A vízszintes tengelyen a futási idők, a függőlegesen a tárhelyhatékonyság látható. Jobbról-balra haladva egyre lassabb futási időkkel, míg lentről felfelé haladva egyre jobb tárhelyhatékonysággal találkozhatunk.



3. ábra
Idő- és tárhelyhatékonyság kapcsolata a vizsgált szoftvereknél.

Érdemes megfigyelni, hogy a dokumentum archívum esetében mindegyik eszköz közel azonos eredményt produkált, ez az adathalmaz részleges tömörítettségéből és kiegyensúlyozottságából adódik.

Időhatékonyság tekintetében megfigyelhető, hogy a rendszerkörnyezetek mentése során a *duplicati* 6-8-szor lassabban teljesített a többi vizsgált szoftvernél. Tárhatékonyság tekintetében két csoportra osztható a mezőny. A *restic* és a *duplicati* végezték a leghatékonyabban a tömörítést, míg a *borg* és a *duplicity* kb. 20%-kal gyengébb eredményt értek el.

Látható az is, hogy a többi szoftverhez képest a *duplicati* időben jelentősen gyengébb teljesítményt nyújtott, azonban tárhelyhatékonyságot tekintve a legjobb eszközök között teljesített. A *borg* és a *restic* közel azonos teljesítményt nyújtottak, azonban a *borg* gyengébb tömörítési hatékonyságot volt képes elérni.

6. ÖSSZEFOGLALÁS

Összegzésként elmondható, hogy a vizsgált szoftvereknél eltérést tapasztaltunk időszükséglet és tárhelyhatékonyság szempontjából. Általános esetben a *restic* bizonyult a leghatékonyabb eszköznek, de a mentési eszközről történő döntés során a technológiai szaktudást és a szervezet speciális felhasználási igényeit is figyelembe kell venni. A *duplicati* futási időt tekintve gyengén teljesített ugyan, de tömörítési hatékonyságban és felhasználóbarátságban kiemelkedik a mezőnyből. A *borg* időhatékonyságát tekintve a legjobb eszköznek bizonyult, és a támogatott tömörítési algoritmusok széles köre miatt potenciált mutat a további finomhangolásra. A *duplicity* elfogadható hatékonysággal rendelkezik, de egyik jellemzőjét tekintve sem emelkedett ki a mezőnyből.

A jövőben érdemes lehet megvizsgálni az említett szoftverek olyan további funkcióit is, mint az inkrementális biztonsági mentés, valamint a fájlok visszaállítása. Lehetséges további kutatási terület a biztonsági mentési eszközök optimalizálá-

sa, ahol különböző kombinációkat vizsgálhatunk darabolási algoritmusok, tömörítési módszerek és titkosítási eljárások terén.

Ezen felül érdekes volt azzal szembesülni, hogy ezek az aktív fejlesztés alatt álló eszközök jellemzően nem hasznosítják a tartalomalapú darabolási eljárásokkal kapcsolatos kutatások legújabb eredményeit. Célszerű lenne megvizsgálni ezen CDC algoritmusok korlátait, illetve feltárni az implementációt akadályozó tényezőket.

IRODALOMJEGYZÉK

[1] PALICZ, T., SAS, T., SZABÓ, Z., TÓTH, T., TISÓCZKI, J., BENCSIK, B. and JOÓ, T.: **Magyar kórházakban előfordult zsarolóvírus támadások esetei**. IME - Az egészségügyi vezetők szaklapja, 20(1), 32–38. 2021, https://doi.org/10.53020/ime-2021-105

[2] ABRAMS, L.: **MediaMarkt hit by Hive ransomware, initial $240 million ransom**. 2021, https://www.bleepingcomputer.com/news/security/mediamarkt-hit-by-hive-ransomware-initial-240-million-ransom/, (Hozzáférés: 2023.12.08.)

[3] X. **Amikor tényleg baj van**, 2023, https://itbusiness.hu/technology/security/amikor-tenyleg-baj-van/ (Hozzáférés: 2023.12.08.)

[4] Appaji Nag YASA, G. and NAGESH, P. C.: **Space savings and design considerations in variable length deduplication**. ACM SIGOPS Operating Systems Review, 46(3), 2012, pp. 57-64. https://doi.org/10.1145/2421648.2421657

[5] XIAOWEI, M., KETAI, H., XU, X. and AIJUN, L.: **Research on an Incremental Backup Method Based on CAD Engineering Data File**. IEEE 18th Conference on Industrial Electronics and Applications (ICIEA), 2023, https://doi.org/10.1109/iciea58696.2023.10241897

[6] ZHANG, Y. et al.: **A Fast Asymmetric Extremum Content Defined Chunking Algorithm for Data Deduplication in Backup Storage Systems**. IEEE Transactions on Computers, 2016, https://doi.org/10.1109/tc.2016.2595565

[7] ZHOU, P., WANG, Z., XIA, W. and ZHANG, H.: **UltraCDC: A Fast and Stable Content-Defined Chunking Algorithm for Deduplication-based Backup Storage Systems**. IEEE International Performance, Computing, and Communications Conference, 2022, https://doi.org/10.1109/ipccc55026.2022.9894295

[8] RAJKUMAR, K., DHANAKOTI, V.: **Fuzzy-Dedup: A secure deduplication model using cosine-based Fuzzy interference system in cloud application**. Journal of Intelligent & Fuzzy Systems, 43(3), 2022, pp. 2819-2832. https://doi.org/10.3233/jifs-210511

[9] ISO 32000-2:2020 szabvány (**PDF 2.0**)

[10] ISO 29500:2008-2016 szabvány (**DOCX Transitional (Office Open XML)**)

[11] Valverde CAMESELLE, R., Gonzalez LABRADOR, H.: **Addressing a billion-entries multi-petabyte distributed file system backup problem with cback: From files to objects**. EPJ Web of Conferences, 251, 02071, 2021, https://doi.org/10.1051/epjconf/202125102071