|  |  |  |
| --- | --- | --- |
| ***Description: Unigrb*** | UNIVERZITET U NIŠU  ELEKTRONSKI FAKULTET | ***Description: logo_1960_4*** |

**Forenzika *storage medijuma:* Hex editor sa RAW pristupom podacima na disku i sakrivanje particije**

Digitalna Forenzika

|  |  |  |
| --- | --- | --- |
| Student: |  | Mentor: |
| Dušica Milanović, br. ind.2128/25 |  | prof. dr Bratislav Predić |

Niš, januar 2026.

SADRŽAJ

[**UVOD** 3](#_Toc223211607)

[**TVRDI DISK I ORGANIZACIJA PODATAKA** 4](#_Toc223211608)

[Fizička struktura tvrdog diska 4](#_Toc223211609)

[Osnovne karakteristike tvrdog diska 5](#_Toc223211610)

[Organizacija podataka na magnetnim pločama 5](#_Toc223211611)

[Zoned Bit Recording i gustina zapisa 7](#_Toc223211612)

[Struktura sektora 7](#_Toc223211613)

[Logičke celine tvrdog diska 7](#_Toc223211614)

[Klasteri kao osnovne logičke jedinice 8](#_Toc223211615)

[Slack prostor 8](#_Toc223211616)

[Particije i volumeni 9](#_Toc223211617)

[Boot sektor i Master Boot Record-a (MBR) 10](#_Toc223211618)

[Forenzički značaj MBR-a 11](#_Toc223211619)

[Datotečni sistem 13](#_Toc223211620)

[Tipovi datotečnih sistema 13](#_Toc223211621)

[FAT datotečni sitem 14](#_Toc223211622)

[NTFS datotečni sistem 15](#_Toc223211623)

[Poređenje FAT (FAT32) i NTFS datotečnih sistema 16](#_Toc223211624)

[**SAKRIVANJE PARTICIJE TVRDOG DISKA -PRAKTIČNA IMPLEMENTACIJA** 18](#_Toc223211625)

[Radno okruženje i metodologija 18](#_Toc223211626)

[Sakrivanje NTFS particije pomoću HeX editora 18](#_Toc223211627)

[Programska implementacija sakrivanja NTFS particije 22](#_Toc223211628)

[Ograničenja sakrivanja particija i ponašanje operativnog sistema 25](#_Toc223211629)

[Forenzičke implikacije manipulacije MBR-a 27](#_Toc223211630)

[**ZAKLJUČAK** 28](#_Toc223211631)

[**LITERATURA** 29](#_Toc223211632)

# **UVOD**

Razvoj informacionih tehnologija doveo je do rapidnog porasta količine digitalnih podataka koji se svakodnevno skladište na različitim vrstama storage medijuma, kao što su tvrdi diskovi, SSD uređaji i USB memorije. Ovi medijumi predstavljaju osnovu savremenih informacionih sistema, ali istovremeno i čest predmet bezbednosnih incidenata, zloupotreba i digitalnih istraga. U tom kontekstu, digitalna forenzika, a posebno *forenzika storage medijuma*, ima ključnu ulogu u analizi, očuvanju i tumačenju podataka na niskom nivou.

Za razliku od klasičnog rada sa podacima putem operativnog sistema i datotečnog sistema, forenzička analiza često zahteva *RAW pristup podacima*[[1]](#footnote-1), odnosno direktan pristup sektorima diska, nezavisno od njihove logičke organizacije. Ovakav pristup omogućava uvid u skrivene, obrisane ili namerno maskirane podatke koji nisu vidljivi standardnim mehanizmima operativnog sistema.

Poseban značaj u digitalnoj forenzici ima analiza *Master Boot Record-a (MBR)*[[2]](#footnote-2), jer ova struktura sadrži ključne informacije o rasporedu particija i načinu na koji operativni sistem prepoznaje i montira skladišne celine.

Jedan od mehanizama prikrivanja podataka jeste *izmena karakteristika datotečnog sistema*, uključujući i manipulaciju particionim tabelama. Posebno interesantan primer jeste *sakrivanje particije*, koje se može ostvariti promenom određenih bajtova u MBR (Master Boot Record) strukturi diska. Ovakve tehnike mogu imati legitimnu primenu u edukativne i istraživačke svrhe, ali se takođe koriste i u maliciozne svrhe, zbog čega su relevantne za forenzičku analizu.

Cilj ovog rada je da se prikaže teorijska osnova forenzike storage medijuma, sa posebnim fokusom na RAW podatke, strukturu diska, datotečne sisteme i MBR particionu tabelu, kao i da se praktično demonstrira sakrivanje particije na dva načina:

* Korišćenjem heksadecimalnog editora *(HxD)*i
* implementacijom *programskog rešenja sa direktnim pristupom disku.*

Pored same implementacije, u radu će biti analizirana i *ograničenja, rizici i problemi* na koje se nailazi prilikom rada sa RAW podacima, uključujući bezbednosne mehanizme operativnog sistema, rizik od oštećenja podataka i potrebu za pažljivim forenzičkim pristupom.

Rad je strukturisan u dva dela. U prvom delu obrađuje se teorijska osnova, koja obuhvata principe rada storage medijuma, RAW pristup podacima, datotečne sisteme i MBR strukturu. Drugi deo rada posvećen je praktičnoj implementaciji i analizi rezultata, uz osvrt na izazove i ograničenja tokom realizacije.

# **TVRDI DISK I ORGANIZACIJA PODATAKA**

Tvrdi disk (engl. *hard disk, hard disk drive – HDD*) predstavlja osnovni uređaj sekundarne memorije namenjen trajnom skladištenju podataka u računarskim sistemima. Podaci se na tvrdom disku zapisuju magnetnim putem na površini kružnih, rotirajućih ploča, pri čemu se informacije predstavljaju u binarnom obliku. Zapis i čitanje podataka ostvaruju se pomoću magnetnih glava koje se precizno pomeraju iznad površine diska i menjaju magnetni polaritet sitnih oblasti na pločama. Disk je povezan sa računarom pomoću posebnog interfejsa (obično se koriste *IDE*[[3]](#footnote-3) i *SCSI*[[4]](#footnote-4)).

Tvrdi disk ima centralnu ulogu u funkcionisanju računara, jer sadrži operativni sistem, aplikativne programe i korisničke podatke. Zbog svoje uloge u trajnom čuvanju informacija, on predstavlja ključni element u oblasti digitalne forenzike, gde se često zahteva analiza podataka koji nisu dostupni kroz standardne mehanizme operativnog sistema.

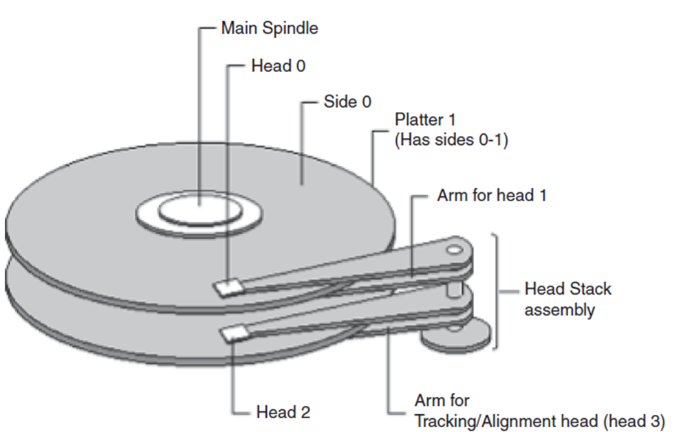
Podaci na tvrdom disku organizovani su kroz jasno definisane fizičke i logičke strukture. Fizička struktura obuhvata mehaničke i elektronske komponente diska i način zapisa podataka na magnetnim pločama, dok logička struktura omogućava apstraktno korišćenje fizičkog prostora kroz sektore, klastere i particije. Između ova dva nivoa nalazi se datotečni sistem, koji definiše pravila organizacije, adresiranja i zaštite podataka, omogućavajući korisniku rad sa datotekama i direktorijumima bez poznavanja njihove fizičke lokacije.

Razumevanje organizacije podataka na tvrdom disku predstavlja osnovu za analizu RAW podataka, kao i za tumačenje struktura poput particionih tabela i boot sektora. Ovaj uvid je neophodan za pouzdanu forenzičku analizu i bezbedan direktan pristup uređajima za skladištenje podataka.

### Fizička struktura tvrdog diska

Tvrdi disk predstavlja uređaj za trajno skladištenje podataka koji funkcioniše na principu magnetnog zapisa. Kada računar koristi određeni program ili podatke, oni se učitavaju sa diska u radnu memoriju, a prilikom izmene sadržaja datoteka, promene se ponovo upisuju na tvrdi disk. Kao medijum za čitanje i upisivanje podataka koriste se kružne ploče premazane magnetnim materijalom koje se rotiraju velikom brzinom.

Tvrdi disk sadrži jednu ili više magnetnih ploča postavljenih na zajedničku osovinu. Iznad i ispod svake ploče nalaze se glave za čitanje i upisivanje podataka, koje su smeštene na pokretnim nosačima. Glave se nalaze na izuzetno malom rastojanju od površine ploče i tokom rada ne dolaze u direktan kontakt sa njom. Kako se ploče rotiraju, glave se pomeraju po poluprečniku diska i vrše čitanje ili upis podataka.



Slika 1 – Unutrašnja struktura tvrdog diska (ploče, glave i osovina)

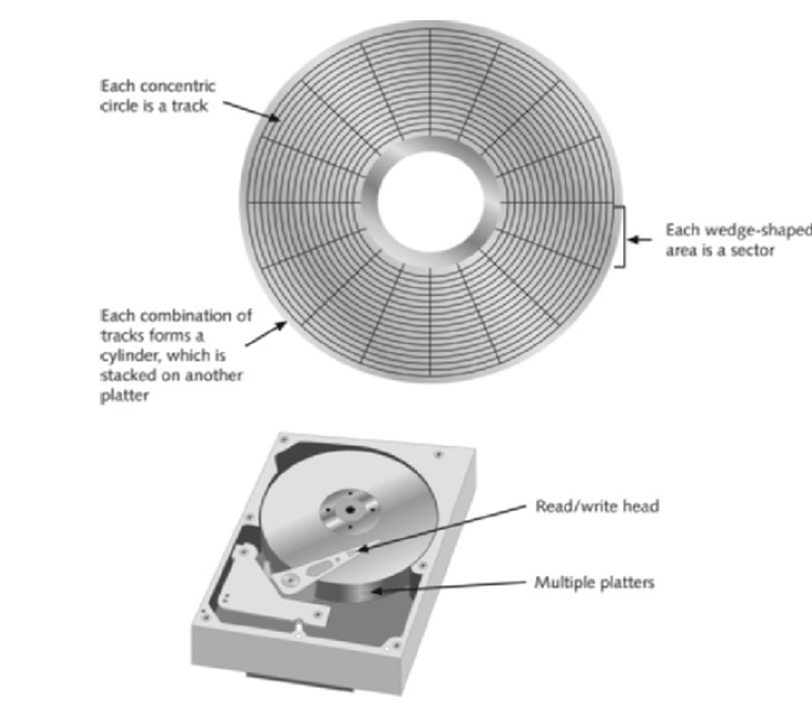
#### Osnovne karakteristike tvrdog diska

Rad tvrdog diska zavisi od više tehničkih karakteristika koje direktno utiču na njegove performanse. Najznačajnije karakteristike su:

* *Kapacitet*– količina podataka koja se može skladištiti na disku, izražena u gigabajtima ili terabajtima, a zavisi od broja ploča i gustine zapisa,
* *Interfejs*– tip veze između tvrdog diska i matične ploče računara (npr. SATA, SAS),
* *Brzina rotacije* – broj obrtaja ploča u minuti (RPM), koji utiče na brzinu pristupa podacima,
* *Vreme pristupa* – vreme potrebno da glava dođe do željene lokacije i započne prenos podataka,
* *Brzina transfera* – brzina prenosa podataka sa i na disk.

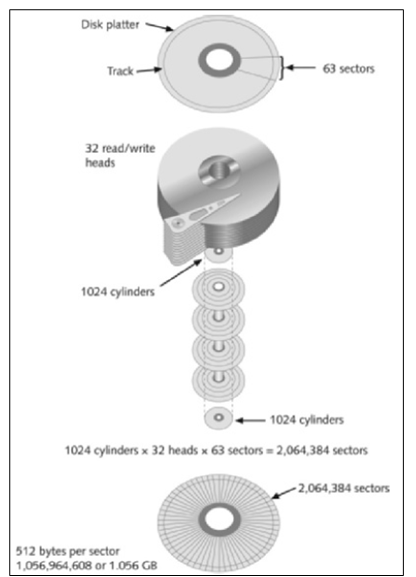
#### Organizacija podataka na magnetnim pločama

Podaci na tvrdom disku organizovani su u *trake (tracks)*, koje predstavljaju koncentrične krugove na površini ploče. Svaka traka podeljena je na *sektore*, koji predstavljaju najmanje fizičke jedinice za skladištenje podataka. Najčešća veličina sektora iznosi 512 bajtova, dok savremeni diskovi koriste i sektore veličine 4096 bajtova.



Slika 2 – Organizacija podataka na ploči tvrdog diska (trake, sektori i cilindri)

Skup traka istog prečnika na svim pločama tvrdog diska naziva se *cilindar*. Da bi se odredio ukupan broj bajtova na disku, potrebno je pomnožiti broj cilindara sa brojem glava i brojem sektora. Ovo računanje se još naziva i *CHS (Cylinder, Head, Sector)* računanje. Koncept cilindra omogućava efikasniji pristup podacima jer se podaci sa više ploča mogu čitati bez promene položaja glava.



Slika 3 – CHS računanje

#### Zoned Bit Recording i gustina zapisa

Za povećanje kapaciteta diska koristi se tehnika *Zoned Bit Recording (ZBR)*, pri kojoj se trake grupišu u zone u zavisnosti od njihove udaljenosti od centra diska. Spoljne trake imaju veći obim i mogu sadržati veći broj sektora u odnosu na unutrašnje trake, čime se povećava ukupna količina skladištenih podataka.

Tvrdi disk karakterišu tri osnovne vrste gustine zapisa:

* *Gustina traka (track density)* – rastojanje između susednih traka na ploči,
* *Površinska gustina (area density)* – broj bitova po kvadratnom inču površine ploče,
* *Bitska gustina (bit density)* – broj bitova po jedinici dužine trake.

Svaka ploča na tvrdom disku ima istu gustinu traka. *Gustina traka (track density)* predstavlja kompaktnost traka na ploči. Proizvođači se trude da povećaju gustinu traka kako bi povećali količinu podataka koja može da se skladišti na disku. Gustina traka određuje količinu podataka koja može da se skladišti na disku. Trake su uglavnom numerisane od 0 do 1023, počevši od spoljne ivice ploče ka centru.

#### Struktura sektora

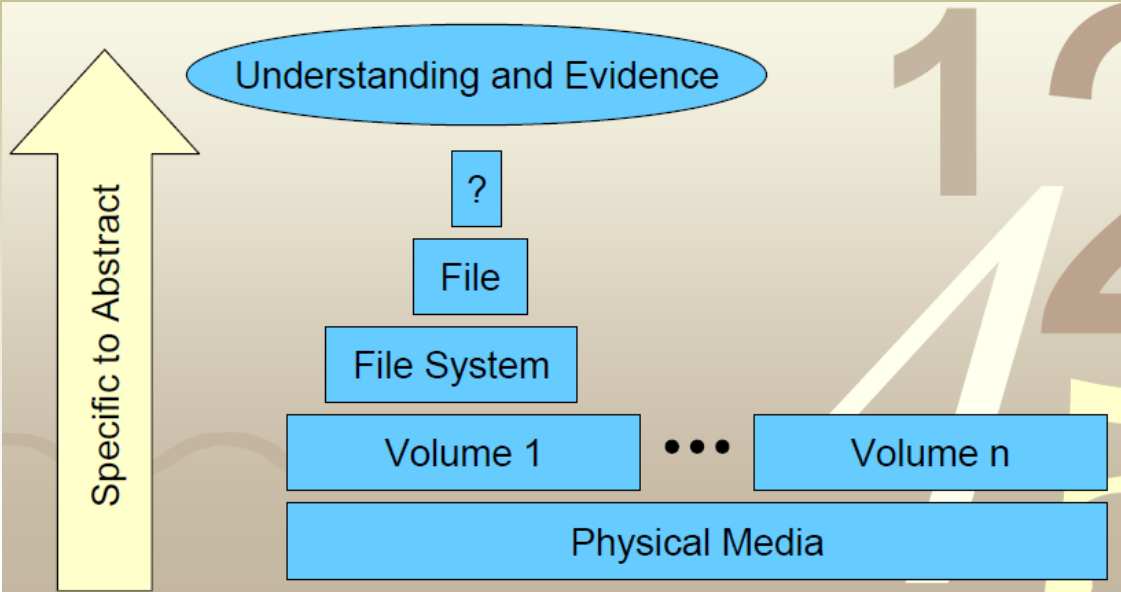
Sektor predstavlja osnovnu fizičku jedinicu za skladištenje podataka na tvrdom disku. Pored korisničkih podataka, sektor sadrži i dodatne informacije neophodne za pouzdano funkcionisanje diska, kao što su identifikacioni podaci i kodovi za detekciju i korekciju grešaka.

Tipičan sektor sadrži sledeće komponente:

* *ID informacije,*
* *sinhronizaciona polja,*
* *polje za podatke,*
* *ECC (Error Correcting Code)* i
* *razmake (gaps) između zapisa.*

### Logičke celine tvrdog diska

Nakon fizičke organizacije tvrdog diska, neophodno je razmotriti njegovu logičku strukturu. Logička struktura diska predstavlja apstraktni nivo organizacije koji omogućava operativnom sistemu da upravlja fizičkim resursima diska na efikasan i korisniku transparentan način. Iako su podaci fizički smešteni u sektore i trake, operativni sistem ih posmatra kroz logičke celine kao što su klasteri, particije i volumeni, čime se pojednostavljuje upravljanje skladištem podataka.



Slika 4 – Hijerarhija fizičke i logičke organizacije podataka na disku

Na slici 4 prikazan je hijerarhijski odnos između fizičkog medijuma za skladištenje podataka i logičkih struktura koje koristi operativni sistem. Fizički medijum može biti podeljen na jedan ili više volumena, nad kojima se formira datotečni sistem. Datoteke predstavljaju najviši nivo apstrakcije sa kojim korisnik neposredno radi, dok su detalji fizičke organizacije podataka sakriveni od korisnika.

#### Klasteri kao osnovne logičke jedinice

Operativni sistem ne upravlja pojedinačnim sektorima, već grupama sektora koje se nazivaju *klasteri (allocation units)*. Klaster predstavlja najmanju logičku jedinicu prostora koju operativni sistem dodeljuje datoteci. Datotečni sistem deli raspoloživi prostor na tvrdom disku na diskretne celine u cilju pojednostavljenja upravljanja prostorom i poboljšanja performansi. Veličina klastera zavisi od kapaciteta diska i tipa datotečnog sistema.

Jedan klaster se sastoji od više fizičkih sektora koji su međusobno susedni i čine neprekinutu celinu na disku. Veličina klastera određuje se prilikom particionisanja i formatiranja diska i zavisi od ukupne veličine particije. Diskovi većeg kapaciteta obično koriste klastere većih dimenzija. U praksi, jedan klaster najčešće sadrži između 4 i 64 sektora, dok se u određenim slučajevima koriste i klasteri koji sadrže 128 sektora.

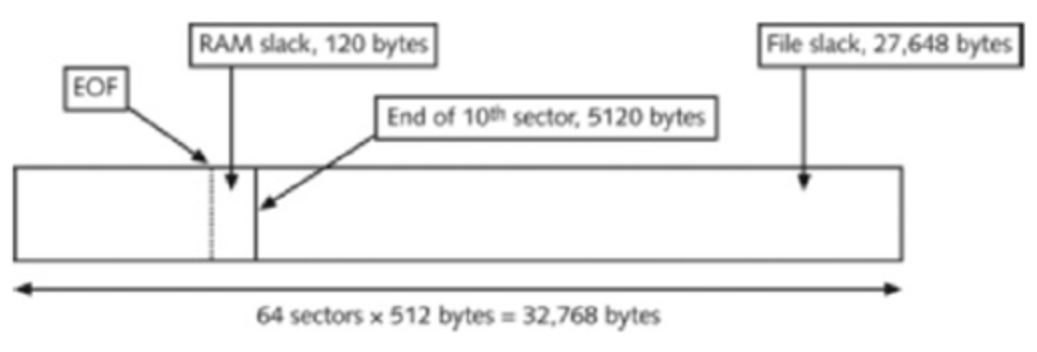
Datoteka na disku uvek zauzima ceo klaster ili celobrojni broj klastera, bez obzira na njenu stvarnu veličinu. Klasteri koji pripadaju istoj datoteci logički su međusobno povezani i ulančani pomoću struktura koje održava datotečni sistem. Ovaj proces ulančavanja klastera nije vidljiv korisniku niti aplikacijama, već se njime upravlja na nivou datotečnog sistema.

#### Slack prostor

Prilikom upisa datoteka na disk često dolazi do pojave neiskorišćenog prostora unutar klastera, poznatog kao *slack space*. Slack prostor predstavlja razliku između stvarne veličine datoteke i ukupnog prostora koji zauzimaju dodeljeni klasteri. Iako deo klastera može ostati neiskorišćen, datoteci se uvek dodeljuje ceo klaster ili celobrojni broj klastera.

Slack prostor se sastoji iz dva dela:

* *RAM slack* – prostor između poslednjeg bajta datoteke i kraja sektora u kojem se datoteka završava,
* *File slack* – prostor između kraja poslednjeg sektora datoteke i kraja klastera.



Slika 5 – Prikaz neiskorišćenog prostora unutar klastera (RAM slack i file slack)

Neiskorišćeni prostor unutar klastera ima poseban značaj u oblasti digitalne forenzike, jer može sadržati ostatke prethodno skladištenih podataka koji nisu prepisani prilikom upisa nove datoteke.

#### Particije i volumeni

Svaki tvrdi disk može biti podeljen na više *particija*, koje predstavljaju logički izdvojene celine za skladištenje podataka. Particionisanje diska omogućava organizaciju prostora, razdvajanje sistemskih i korisničkih podataka, kao i instalaciju više operativnih sistema na istom fizičkom uređaju.

Postoje dve osnovne vrste particija:

* *Primarna particija (Primary Partition)* – particija koja može sadržati operativni sistem i podatke neophodne za proces pokretanja računara,
* *Proširena particija (Extended Partition)* – particija koja služi kao kontejner za jedan ili više logičkih diskova i ne može se direktno koristiti za pokretanje sistema.

Windows operativni sistemi podržavaju najviše četiri primarne particije, ili tri primarne particije i jednu proširenu particiju, unutar koje se može kreirati više logičkih diskova.

Nad jednom ili više particija formira se *volumen*, koji predstavlja logičku celinu dostupnu operativnom sistemu i korisniku. Svaki volumen ima sopstvenu strukturu i može sadržati jedan datotečni sistem.

#### Boot sektor i Master Boot Record-a (MBR)

Prvi sektor na disku ima posebnu ulogu u procesu pokretanja računarskog sistema i naziva se *boot sektor*.

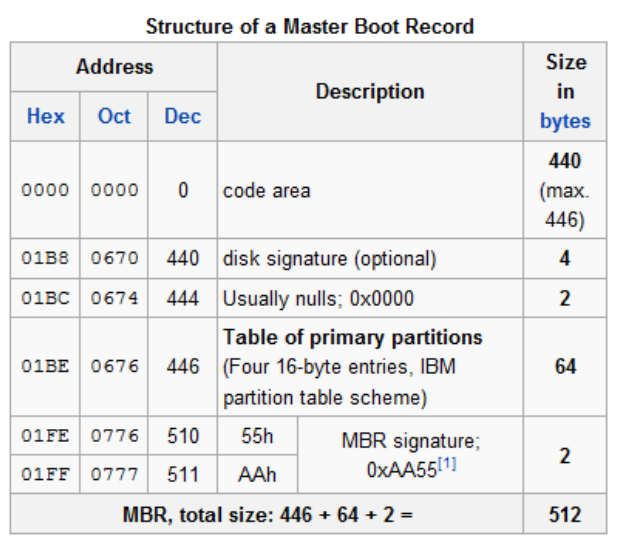
Detaljniji prikaz strukture Master Boot Record-a i njegove analize dat je u cilju potpunijeg razumevanja načina na koji se informacije o particijama fizički skladište na disku i kako se mogu interpretirati u forenzičkom kontekstu.

U zavisnosti od načina organizacije diska, razlikuju se dve osnovne strukture boot sektora:

* *Volume Boot Record (VBR)* predstavlja prvi sektor na uređaju za skladištenje koji još nije particionisan i sadrži osnovni kod za učitavanje operativnog sistema i
* *Master Boot Record (MBR)* predstavlja prvi sektor na particionisanom disku i ima veličinu od 512 bajtova. Struktura MBR-a sastoji se iz tri osnovna dela:
  + *Bootstrap Code Area* – ***bajtovi 0–445 (446 bajtova)*** *-* deo koji sadrži programski kod za pokretanje sistema (446 bajtova),
  + *Master Partition Table (MPT)* – ***bajtovi 446-509*** - tabela particija koja sadrži informacije o rasporedu particija na disku (64 bajta):
* Tačno **4 unosa**, svaki po **16 bajtova** → ukupno 64 bajta,
* Svaki unos opisuje jednu particiju (primarnu, ili proširenu).
  + *Boot potpis* –***bajtovi 510-511*** poslednja dva bajta sektora, čije vrednosti **0x55** i **0xAA** označavaju da je disk boot-abilan.

|  |  |  |
| --- | --- | --- |
| Offset (hex) | Veličina | Opis |
| 0x000 – 0x1BD | 446 B | Boot code |
| 0x1BE – 0x1FD | 64 B | Partition table (4 × 16 B) |
| 0x1FE – 0x1FF | 2 B | MBR signature (55 AA) |

Tabela 1 - Struktura MBR sektora (LBA 0)



Slika 6 - Struktura Master Boot Record-a

#### Forenzički značaj MBR-a

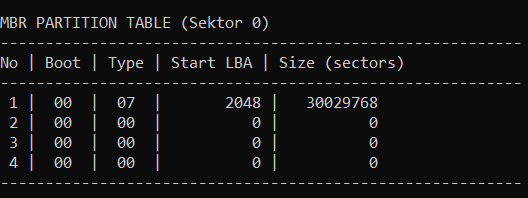
U forenzičkoj analizi, posebna pažnja se posvećuje *Master Partition Table (MPT)*[[5]](#footnote-5) – tabeli particija, jer ona sadrži informacije o početku, veličini i tipu svake particije. Analizom ovih podataka moguće je identifikovati:

* izbrisane ili skrivene particije,
* nelogične granice particija,
* neslaganja između prijavljenog i stvarnog stanja diska.



Slika 7 - Prikaz Master Boot Record-a iz Debug Console

Na slici 7. je prikazan 0 sektor MBR iz Debug Console. *Sektor 0* na disku (LBA 0) je tradicionalno MBR sektor ako je disk particionisan MBR šemom. Belom bojom osenčen je 64-bajtni deo koji predstavlja *Master Partition Table.* U sektoru 0 najbitniji su particiona tabela na offsetu 0x1BE koja sadrži četiri 16-bajtna unosa sa informacijama o particijama, kao i MBR potpis 0x55AA na kraju sektora. Boot kod postoji u prvom delu sektora, ali nije relevantan za analizu particija.



Slika 8 - Prikaz MBR particione tabele dobijen programskom analizom RAW sektora diska

*Partition table*sadrži 4 unosa. Svaki 16B unos opisuje particiju i tipično sadrži:

* *Entry #1 -status/boot flag* - označava da je aktivna particija,
* *Entry #2 - partition type* (npr. NTFS, FAT32, “hidden” varijante itd.),
* *Entry #3 - start LBA* (gde particija počinje) i
* *Entry #4 - size in sectors* (veličina particije izražena u sektorima).

Zapis **07** za tip particije označava da se radi o NTFS particiji, pa prikazani disk sadrži tri NTFS particije. U MBR šemi particionisanja (sektor 0), partition table počinje na offsetu 446 i ima 4 unosa, svaki po 16 bajtova. U svakom 16-bajtnom unosu postoji polje Partition Type (1 bajt) koje OS koristi da odluči kakvu particiju gleda (FAT, NTFS, extended, “hidden” varijante itd.).

Prvi particioni unos (entry #1) sadrži validne metapodatke koji ukazuju na postojanje aktivne particije određenog tipa fajl-sistema, dok zapisi sa vrednostima 0x00 u preostalim particionim unosima (entry #2–#4) ukazuju na to da na disku ne postoje dodatne particije, odnosno da storage uređaj sadrži jednu primarnu particiju.

### Datotečni sistem

Nakon razmatranja fizičke i logičke organizacije tvrdog diska, sledeći nivo apstrakcije u organizaciji podataka predstavlja *datotečni sistem*. Datotečni sistem definiše pravila prema kojima se podaci zapisuju, organizuju, adresiraju i štite unutar particija ili volumena. Njegova osnovna uloga je da omogući operativnom sistemu i korisniku jednostavan i strukturisan rad sa podacima, bez potrebe za poznavanjem njihove stvarne fizičke lokacije na disku.

Datotečni sistem predstavlja vezu između logičkih struktura diska (klastera i volumena) i datoteka sa kojima korisnik neposredno radi. On upravlja raspodelom klastera, održava informacije o strukturi direktorijuma i obezbeđuje mehanizme za očuvanje integriteta podataka.

Osnovni elementi svakog datotečnog sistema su *datoteke* i *direktorijumi*. Datoteka predstavlja logičku celinu podataka koja ima ime, veličinu, vremenske atribute i druge metapodatke. Direktorijumi omogućavaju hijerarhijsku organizaciju datoteka i služe kao način grupisanja povezanih podataka.

Pored samih podataka, datotečni sistem čuva i *metapodatke*, odnosno podatke o podacima. Metapodaci obuhvataju informacije kao što su lokacija datoteke na disku, veličina, prava pristupa, vreme kreiranja, izmene i poslednjeg pristupa. Upravo ovi podaci imaju poseban značaj u digitalnoj forenzici.

#### Tipovi datotečnih sistema

Različiti operativni sistemi koriste različite datotečne sisteme, u zavisnosti od zahteva za performansama, bezbednošću i kompatibilnošću. Windows operativni sistemi najčešće koriste NTFS, uz podršku za starije formate kao što su FAT i FAT32.

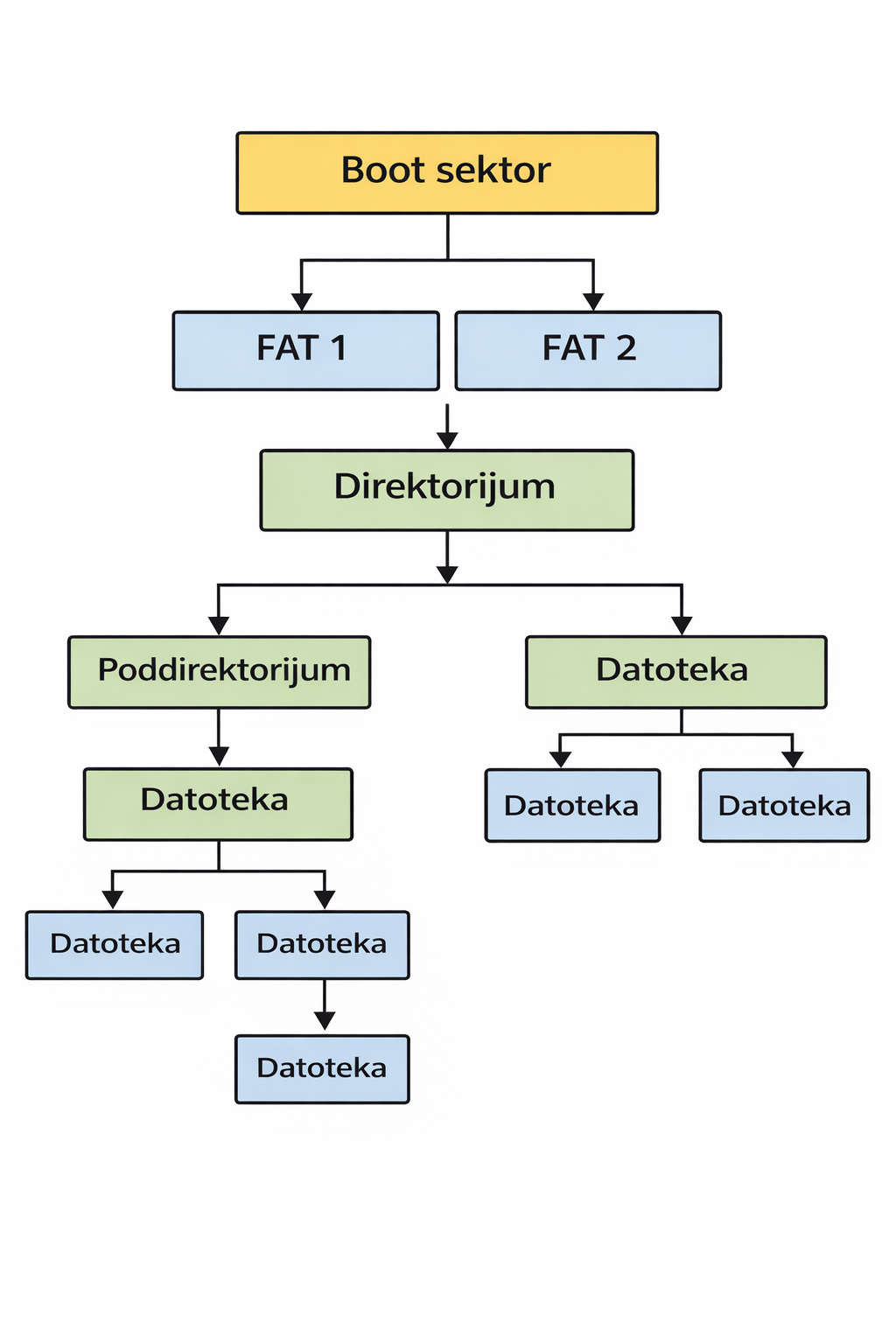
Najčešće korišćeni datotečni sistemi su:

* *FAT (File Allocation Table)* – jednostavan datotečni sistem, koji se koristi u starijim verzijama operativnih sistema i na prenosivim medijumima. Karakteriše ga ograničenje maksimalne veličine datoteke i slabija podrška za bezbednost.
* *FAT32* – unapređena verzija FAT sistema, sa većim kapacitetima, ali i dalje sa ograničenjem maksimalne veličine pojedinačne datoteke.
* *exFAT* – namenjen prenosivim uređajima i većim datotekama, bez ograničenja karakterističnih za FAT32.
* *NTFS (New Technology File System)* – savremeni datotečni sistem koji se koristi u Windows operativnim sistemima. Podržava velike datoteke, napredne bezbednosne mehanizme, journaling i efikasno upravljanje prostorom.
* *ext4* – datotečni sistem koji se koristi u Linux operativnim sistemima, poznat po stabilnosti i dobrim performansama.

#### FAT datotečni sitem

*FAT (File Allocation Table)* predstavlja porodicu datotečnih sistema koja obuhvata varijante FAT12, FAT16 i FAT32, a prvobitno je razvijena za MS-DOS operativne sisteme. Iako je danas u velikoj meri prevaziđen u okviru savremenih računarskih sistema, FAT se i dalje široko koristi na prenosivim medijumima zbog svoje jednostavne strukture i visoke kompatibilnosti. U ovom radu poseban akcenat stavljen je na FAT32, kao najčešće korišćenu varijantu FAT sistema na USB fleš memorijama.

Osnovna karakteristika FAT datotečnog sistema jeste korišćenje tabele alokacije datoteka (FAT), u kojoj se vodi evidencija o tome koji klasteri pripadaju određenoj datoteci i koji klasteri su slobodni. Svaka datoteka se u FAT sistemu predstavlja kao lanac klastera, pri čemu FAT tabela sadrži informacije o redosledu klastera koji čine sadržaj datoteke.



Slika 9 - Hijerarhija FAT datotečnog sistema

FAT datotečni sistem se sastoji od sledećih osnovnih komponenti:

* Boot sektor – sadrži osnovne informacije o datotečnom sistemu i parametre potrebne za njegovo učitavanje;
* FAT tabela – centralna struktura koja povezuje klastere datoteka;
* Root direktorijuma – osnovni direktorijum koji sadrži informacije o datotekama i poddirektorijumima;
* Data region – deo diska u kojem se nalaze stvarni podaci datoteka.

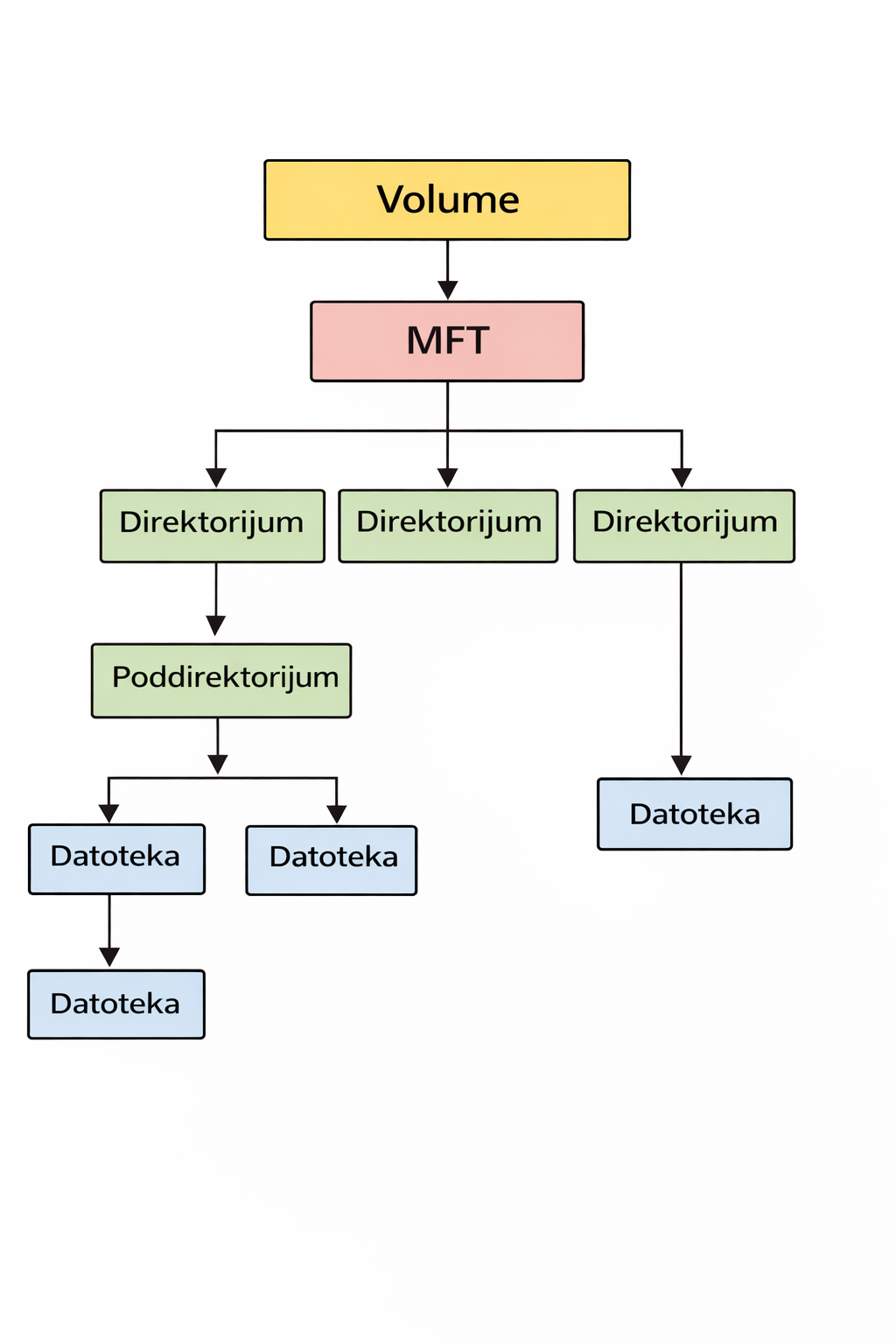
Prilikom upisa datoteke, datotečni sistem pronalazi slobodne klastere i evidentira njihovo ulančavanje u FAT tabeli. Brisanjem datoteke, zapisi u FAT tabeli se uklanjaju, dok stvarni podaci često ostaju na disku sve dok ne budu prepisani.

Prednosti FAT sistema ogledaju se u njegovoj jednostavnosti i širokoj podršci na različitim platformama. Međutim, značajna ograničenja uključuju ograničenje veličine datoteka (posebno kod FAT32), nepostojanje naprednih bezbednosnih mehanizama, veću sklonost fragmentaciji i ograničene mogućnosti oporavka podataka. Zbog toga se FAT danas najčešće koristi na USB fleš memorijama i memorijskim karticama.

#### NTFS datotečni sistem

*NTFS (New Technology File System)* predstavlja savremeni datotečni sistem koji se koristi u Windows operativnim sistemima. Razvijen je sa ciljem da prevaziđe ograničenja FAT sistema i obezbedi veću pouzdanost, bezbednost i efikasnost u radu sa podacima.

Za razliku od FAT sistema, NTFS koristi *Master File Table (MFT)* kao centralnu strukturu za upravljanje datotekama i metapodacima. Svaka datoteka i direktorijum u NTFS sistemu predstavljeni su zapisom u MFT-u.



Slika 10 - Hijerarhija Hijerarhija NTFS datotečnog sistema

*Master File Table* predstavlja srce NTFS datotečnog sistema. MFT sadrži zapise o svim datotekama i direktorijumima, uključujući njihove atribute, lokaciju podataka i bezbednosne informacije.

Svaki MFT zapis ima fiksnu veličinu i može sadržati više atributa, kao što su:

* informacije o imenu datoteke;
* bezbednosni deskriptori;
* vremenski žigovi;
* pokazivači na klastere u kojima se nalaze podaci.

U određenim slučajevima, mali fajlovi mogu biti u potpunosti smešteni unutar MFT zapisa, što dodatno povećava performanse sistema.

NTFS podržava niz naprednih funkcionalnosti koje ga čine pogodnim za savremene operativne sisteme:

* podršku za velike datoteke i volumene;
* kontrolu pristupa i dozvole;
* journaling, koji omogućava oporavak sistema nakon greške;
* enkripciju i kompresiju podataka.

Zahvaljujući ovim karakteristikama, NTFS ima poseban značaj u digitalnoj forenzici, jer omogućava detaljnu analizu metapodataka i aktivnosti nad datotekama.

NTFS predstavlja znatno napredniji datotečni sistem u poređenju sa FAT-om, naročito u pogledu bezbednosti i forenzičke analize. Ipak, FAT i dalje ima primenu u specifičnim slučajevima gde su jednostavnost i kompatibilnost od većeg značaja od naprednih funkcionalnosti.

Datotečni sistem ima ključnu ulogu u digitalnoj forenzici, jer definiše način na koji se podaci i metapodaci skladište i interpretiraju. Upoređivanjem struktura datotečnog sistema sa stvarnim sadržajem diska dobijenim analizom RAW podataka moguće je otkriti neslaganja, pokušaje prikrivanja podataka i manipulacije skladišnim strukturama.

U forenzičkoj analizi često se upoređuje stanje datotečnog sistema sa stvarnim sadržajem diska dobijenim analizom RAW podataka, čime se mogu otkriti neslaganja, pokušaji prikrivanja tragova ili manipulacije podacima.

#### Poređenje FAT (FAT32) i NTFS datotečnih sistema

FAT (FAT32) i NTFS predstavljaju dva fundamentalno različita pristupa organizaciji i upravljanju podacima na storage medijumima. Iako oba sistema imaju istu osnovnu ulogu – skladištenje i pristup podacima – njihove unutrašnje strukture, bezbednosni mehanizmi i način na koji operativni sistem njima upravlja značajno se razlikuju. Ove razlike imaju direktan uticaj na rezultate forenzičke analize i ponašanje operativnog sistema prilikom manipulacije particijama. U Tabeli 2 dat je uporedni prikaz osnovnih karakteristika ova dva sistema, čije je ponašanje dodatno analizirano u praktičnom delu rada.

|  |  |  |
| --- | --- | --- |
| Karakteristika | FAT (FAT32) | NTFS |
| Tip datotečnog sistema | Jednostavan, legacy datotečni sistem | Savremeni datotečni sistem |
| Centralna struktura | FAT tabela (File Allocation Table) | Master File Table (MFT) |
| Organizacija datoteka | Lančano povezani klasteri | Zapisi sa atributima u MFT-u |
| Maksimalna veličina datoteke | Ograničena (do 4 GB) | Praktično neograničena |
| Podrška za velike volumene | Ograničena | Podržana |
| Bezbednosni mehanizmi | Nisu podržani | ACL, dozvole, audit |
| Journaling | Nije podržan | Podržan |
| Oporavak nakon greške | Ograničen | Napredan |
| Fragmentacija | Visoka sklonost | Manja sklonost |
| Metapodaci | Minimalni | Opsežni |
| Mogućnost smeštanja podataka u metapodatke | Nije podržana | Podržana (resident files) |
| Tipična primena | USB fleš memorije, memorijske kartice | Sistemski i eksterni hard diskovi |
| Forenzički značaj | Ograničen | Visok |
| Poštovanje *hidden partition type* u Windows OS | Ignoriše se na USB uređajima | Poštuje se na fiksnim i eksternim diskovima |
| Ponašanje u praktičnom delu rada | Particija ostaje vidljiva u File Explorer-u | Particija postaje skrivena |

Tabela 2 - Uporedni prikaz FAT (FAT32) i NTFS datotečnih sistema

Ponašanje FAT (FAT32) i NTFS datotečnih sistema, prikazano u ovom poglavlju, dodatno je analizirano i potvrđeno u praktičnom delu rada kroz direktnu izmenu MBR particione tabele i pokušaje sakrivanja particija, uz posmatranje reakcije operativnih sistema različitih generacija.

# **SAKRIVANJE PARTICIJE TVRDOG DISKA -PRAKTIČNA IMPLEMENTACIJA**

### Radno okruženje i metodologija

U ovom delu rada prikazan je postupak sakrivanja particije tvrdog diska, sa posebnim fokusom na NTFS datotečni sistem. Eksperiment je realizovan u kontrolisanom okruženju, korišćenjem operativnog sistema Windows 11, koji predstavlja savremeni operativni sistem sa unapređenim bezbednosnim mehanizmima.

Sakrivanje particije realizovano je na dva različita načina:

1. direktnom izmenom struktura diska pomoću **HeX editora**,
2. korišćenjem **programa razvijenog u programskom jeziku C/C++**, koji automatski izvršava potrebne izmene.

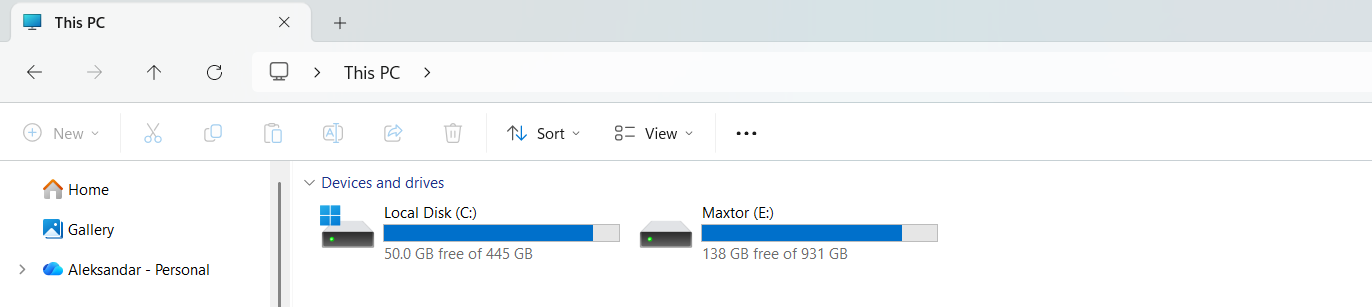
Cilj ovog dela rada je da se pokaže kako se manipulacijom *low-level* struktura diska može uticati na vidljivost particija u operativnom sistemu, kao i da se identifikuju ograničenja savremenih operativnih sistema u odnosu na starije verzije.

### Sakrivanje NTFS particije pomoću HeX editora

*HxD heksadecimalni editor* predstavlja alat koji omogućava direktan pristup RAW podacima na fizičkom disku, nezavisno od operativnog sistema i datotečnog sistema. Za razliku od standardnih alata operativnog sistema, koji pristupaju podacima kroz logičke strukture (volumene i datotečne sisteme), HxD omogućava pregled i izmenu sadržaja diska na bajt-nivou, uključujući i sektore koji nisu vidljivi korisniku.

Zahvaljujući mogućnosti otvaranja fizičkog diska i prikaza sektora prema LBA adresama, HxD je pogodan za analizu i izmenu struktura kao što su Master Boot Record (MBR) i particiona tabela. Ove karakteristike čine HxD odgovarajućim alatom za forenzičku analizu i eksperimentalne demonstracije manipulacije *low-level* strukturama diska.

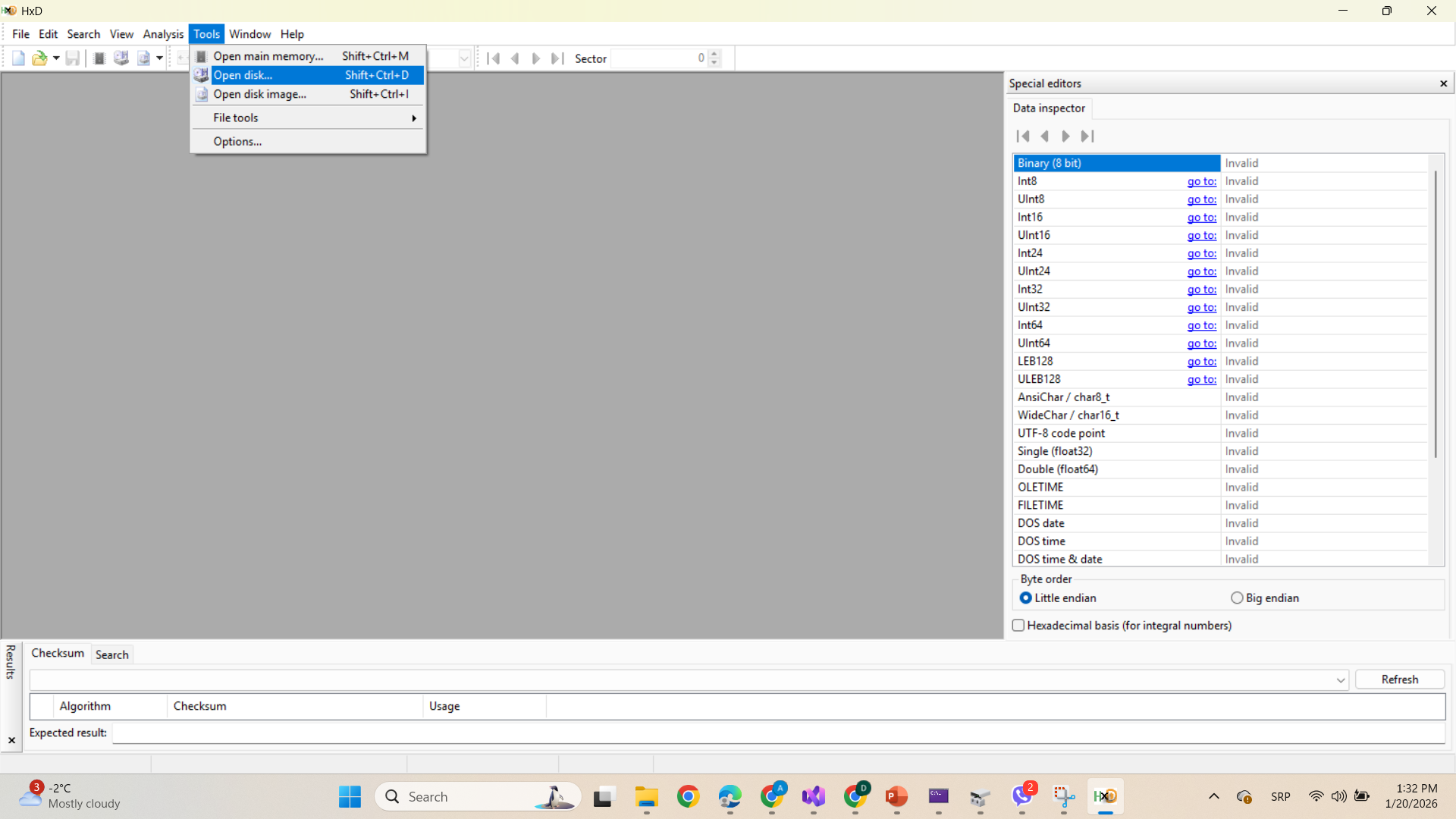
Sakrivanje NTFS particije pomoću heksadecimalnog editora realizovano je direktnom izmenom zapisa u Master Boot Record-u, tačnije promenom vrednosti polja Partition Type u tabeli particija. Izmena ovog bajta dovodi do toga da operativni sistem particiju ne prepoznaje kao standardnu, te je samim tim ne prikazuje korisniku.



Slika 11 - Prikaz particija u File Exploreru pre izvršenog postupka sakrivanja

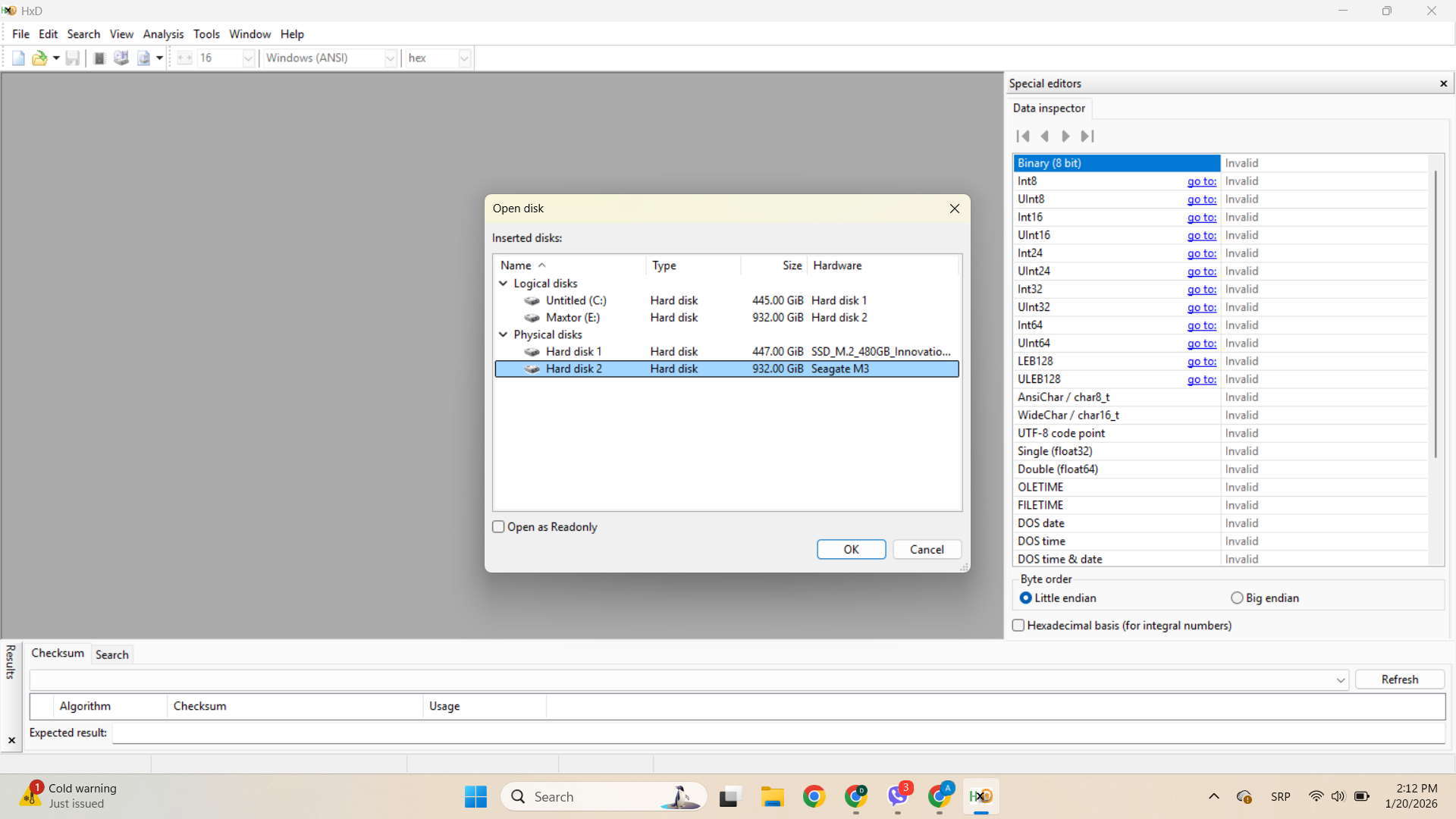
Na slici 11 prikazane su particije tvrdog diska koji se nalazi u samom računaru. U okviru ovog eksperimenta izvršeno je sakrivanje particije Maxtor (E:) korišćenjem heksadecimalnog editora HxD.

Program HxD je pokrenut sa administratorskim privilegijama, kako bi bio omogućen direktan pristup fizičkom disku. U glavnom meniju, izborom kartice *Tools*, a zatim opcije *Open disk…*, omogućeno je otvaranje fizičkog diska, kao što je prikazano na slici 12.



Slika 12 – Kartica Tools u HxD programu

Nakon toga prikazan je dijalog za izbor diska kome se direktno pristupa. Prilikom otvaranja diska neophodno je onemogućiti opciju *Open as Readonly*, jer je u okviru eksperimenta potrebno izvršiti upis podataka na disk (slika 13).



Slika 13 – Kartica Tools u HxD programu

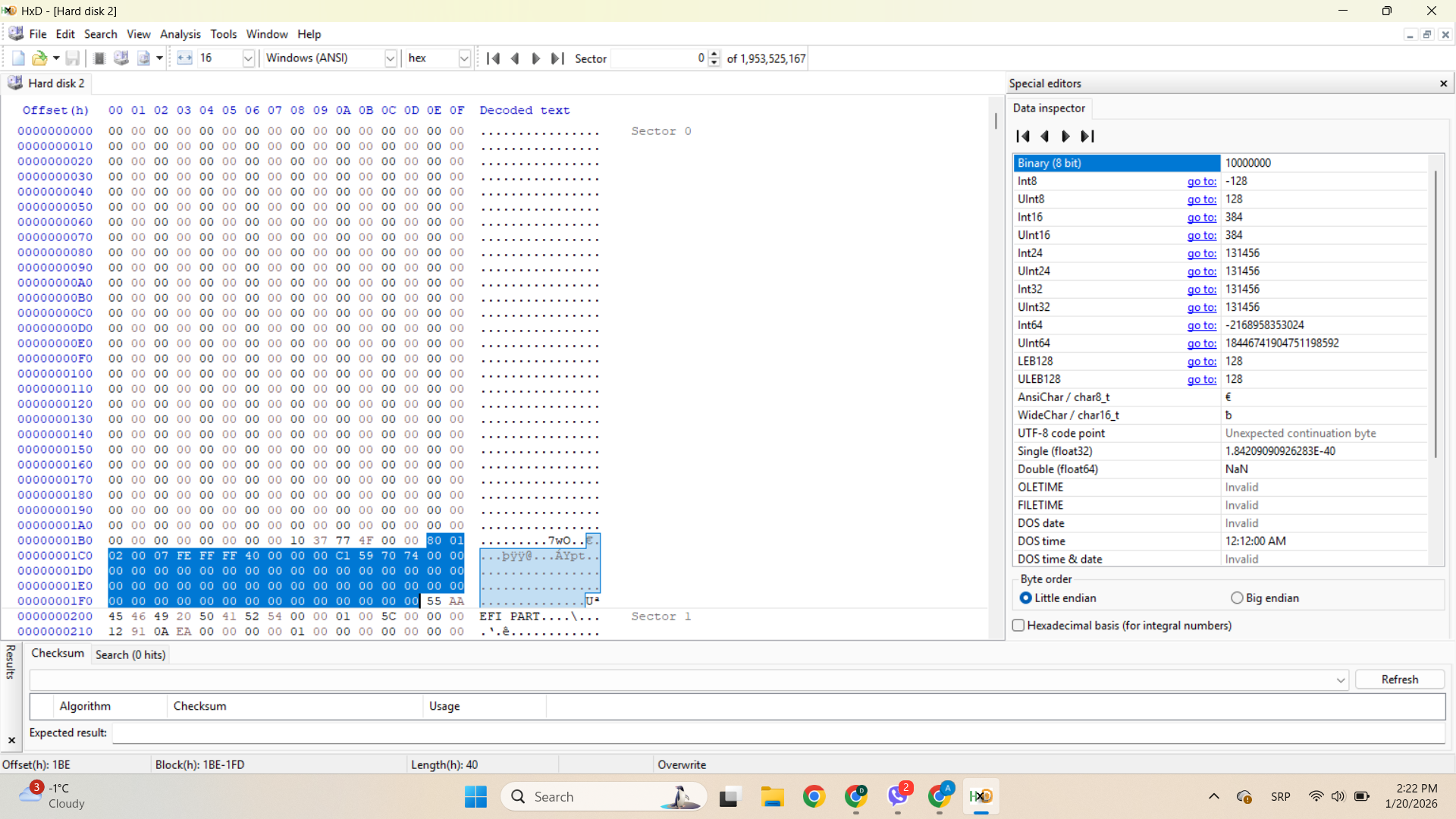
Pre izvođenja bilo kakvih izmena nad *low-level* strukturama diska, neophodno je izvršiti izrada rezervne kopije (backup) relevantnih podataka. Direktna manipulacija RAW sektorima diska, uključujući izmene Master Boot Record-a i particione tabele, nosi rizik od gubitka podataka ili neispravnog rada sistema u slučaju greške.

Zbog toga je pre početka eksperimenta kreirana rezervna kopija podataka, čime je obezbeđena mogućnost povratka sistema u prvobitno stanje. Ova mera predstavlja standardnu bezbednosnu praksu u forenzičkim i eksperimentalnim postupcima koji podrazumevaju direktan pristup fizičkom disku. Izrada rezervne kopije pre početka eksperimenta omogućava očuvanje integriteta originalnih podataka i predstavlja osnovni princip rada u digitalnoj forenzici. Izrada rezervne kopije realizovana je korišćenjem opcije *Save as* u okviru menija *File*, čime je sačuvano originalno stanje podataka pre izvođenja izmena (slika 14).



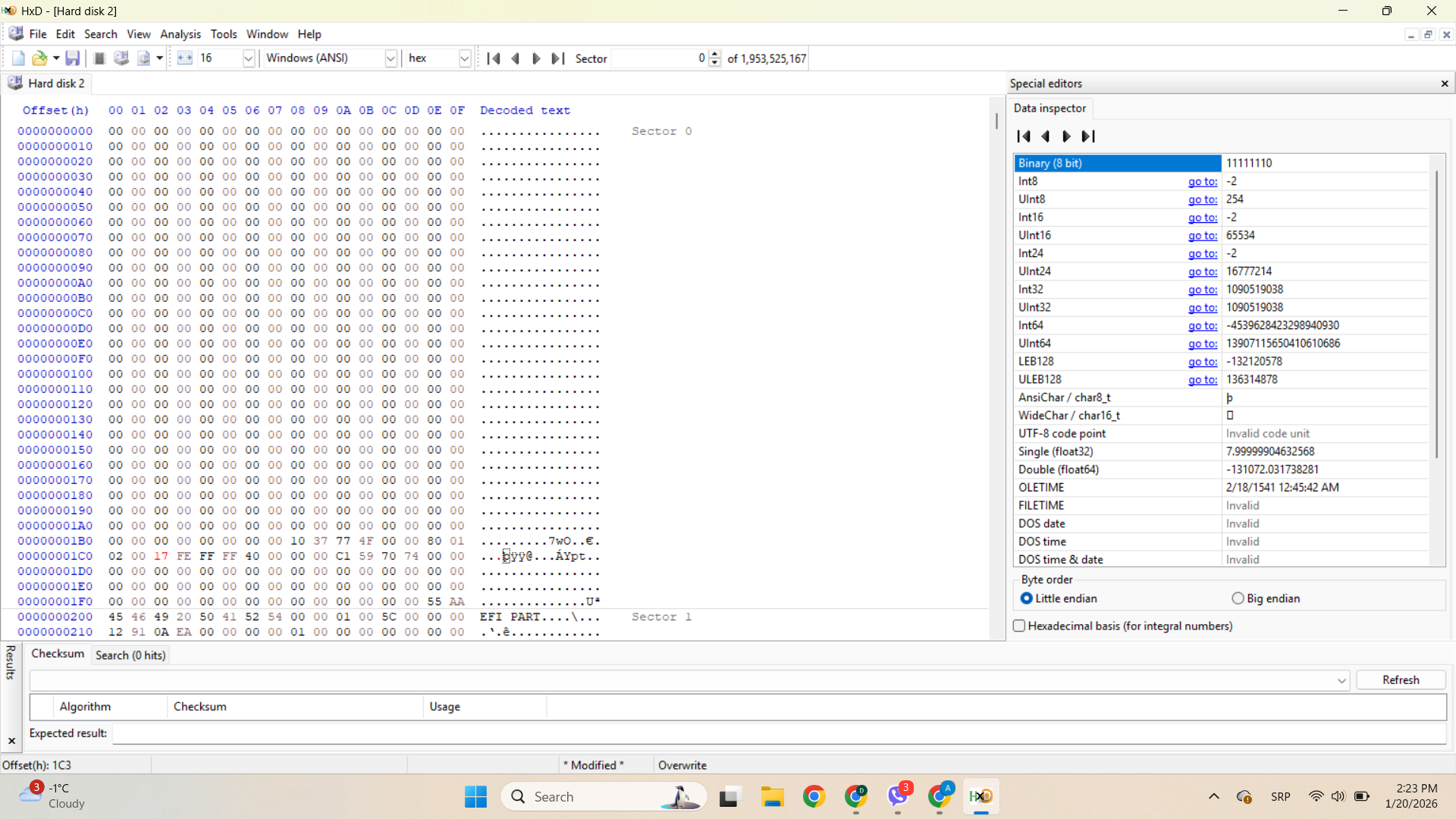
Slika 14 - Kreiranje rezervne kopije sektora diska pomoću HxD heksadecimalnog editora

Potvrdom izbora prikazuje se sadržaj sektora tvrdog diska u heksadecimalnom obliku, uključujući i sektor *LBA 0*, u kojem se nalazi *Master Boot Record* (slika 15).



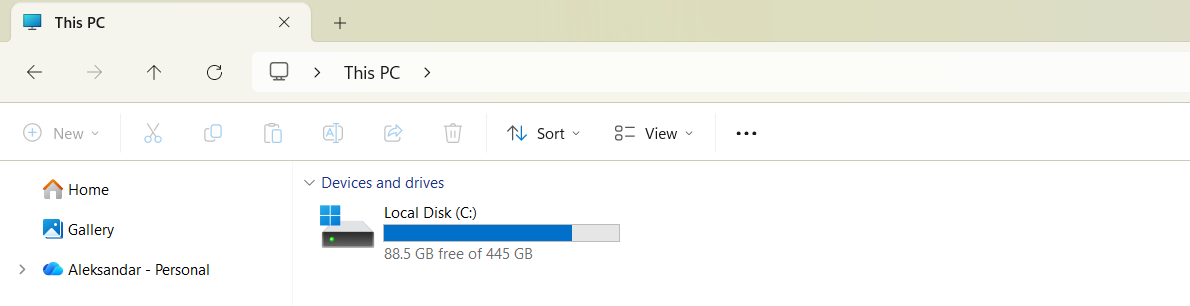
Slika 15 – Kartica Prikaz sektora tvrdog diska u HxD editoru

U opsegu adresa od 0x1BE do 0x1FD nalazi se particiona tabela (*Master Partition Table*), koja sadrži zapise o karakteristikama particija na disku. U okviru particionog unosa koji odgovara NTFS particiji Maxtor (E:), na adresi 0x1E2, izvršena je izmena vrednosti polja Partition Type. Standardna vrednost 0x07, koja označava NTFS particiju, zamenjena je vrednošću 0x17, koja predstavlja oznaku za skrivenu NTFS particiju.



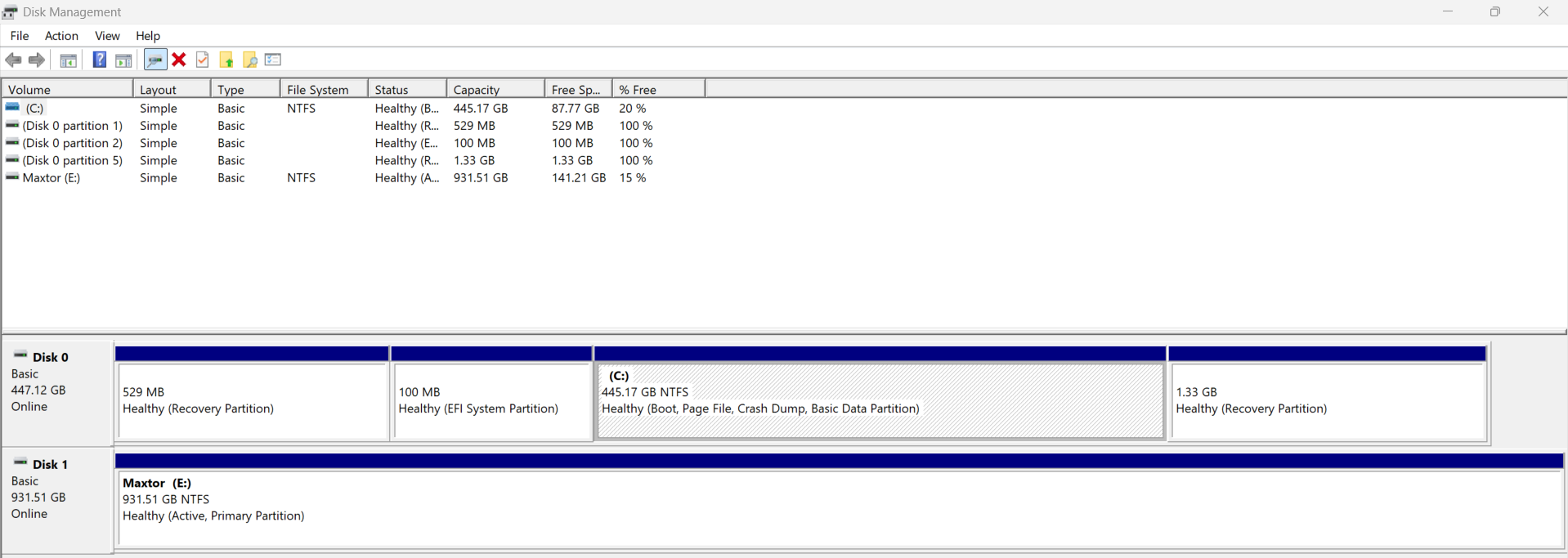
Slika 16 – Heksadecimalni prikaz Master Boot Record-a sa izmenom particione tabele

Nakon izvršene izmene i restartovanja računara, particija čija je karakteristika promenjena više nije vidljiva u File Explorer-u, što je prikazano na slici 17.



Slika 17 – Prikaz particija u File Exploreru nakon izvršenog postupka sakrivanja

Nakon restartovanja računara, particija *Maxtor (E:)* više se ne prikazuje u *File Explorer-u.* Međutim, u alatu *Disk Management* može se uočiti da particija i dalje fizički postoji na disku, ali nije dostupna korisniku kroz standardni prikaz datoteka (slika 18).

Slika 18 - Prikaz particije u Disk Management alatu nakon sakrivanja

Postupak vraćanja prvobitne karakteristike particije identičan je postupku njenog sakrivanja, s tim što se u polje Partition Type ponovo upisuje heksadecimalna vrednost 0x07. Nakon ponovnog restartovanja sistema, particija postaje vidljiva i dostupna korisniku.

### Programska implementacija sakrivanja NTFS particije

Program za sakrivanje NTFS particije razvijen je u programskom jeziku C/C++, korišćenjem razvojnog okruženja Microsoft Visual Studio. Razvoj i testiranje programa izvršeni su na operativnom sistemu Windows 11, uz pokretanje programa sa administratorskim privilegijama, koje su neophodne za direktan pristup fizičkom disku.

Glavni cilj razvijenog programa jeste automatizacija postupka sakrivanja NTFS particije, koji je prethodno realizovan manuelno pomoću heksadecimalnog editora. Program omogućava prikaz boot sektora pre i nakon izvršene izmene, kao i promenu karakteristike NTFS particije izmenom tipa particije u Master Boot Record-u. Na ovaj način obezbeđena je automatizovana i kontrolisana realizacija postupka sakrivanja particije.

U okviru programske implementacije, posebna pažnja posvećena je i bezbednosti podataka. Program automatski kreira poseban direktorijum za rezervne kopije, u okviru kojeg se čuva backup originalnog stanja diska pre izvođenja bilo kakvih izmena. Na taj način obezbeđena je mogućnost povratka sistema u prethodno stanje u slučaju greške tokom izvršavanja programa.

U razvoju programa korišćene su sledeće Windows API funkcije:

* *ReadFile()* - omogućava čitanje podataka sa fizičkog diska u memorijski bafer. U okviru programa koristi se za učitavanje boot sektora (MBR sektora) u memoriju, kako bi se omogućio njegov prikaz i dalja analiza pre izvršavanja izmene,
* *CreateFile()* - koristi se za otvaranje fizičkog diska kao ulazno-izlaznog uređaja,
* *WriteSect()* – koristi se za upis podataka iz memorijskog bafera nazad na fizički disk. Nakon izmene sadržaja boot sektora u memoriji, ova funkcija omogućava trajno zapisivanje izmenjenog sektora na disk, čime se ostvaruje sakrivanje particije,
* *SetFilePointerEx()* - koristi se za pozicioniranje pokazivača datoteke na tačno određenu lokaciju na disku. U programu se koristi za pomeranje pokazivača na početak ciljanog sektora (LBA adresu), pri čemu se pomeranje vrši u koracima od 512 bajtova, što odgovara veličini jednog sektora,
* *CloseHandle() -* koristi se za zatvaranje prethodno otvorenog rukovaoca (handle) nad fizičkim diskom. Njena upotreba je neophodna radi pravilnog oslobađanja sistemskih resursa nakon završetka operacija čitanja ili upisa,
* *CreateDirectoryA() -* koristi se za kreiranje direktorijuma na disku. U programu se koristi za automatsko kreiranje direktorijuma namenjenog čuvanju rezervnih kopija (backup), pri čemu se funkcija bezbedno izvršava i u slučaju da direktorijum već postoji,
* *fopen\_s() -* predstavlja bezbednu verziju funkcije za otvaranje datoteka. U okviru programa koristi se za kreiranje binarne datoteke u koju se upisuje rezervna kopija originalnog boot sektora, čime se obezbeđuje mogućnost vraćanja sistema u početno stanje,
* *fwrite() -* koristi se za upis binarnih podataka u datoteku. U programu se koristi za upis 512 bajtova boot sektora u backup datoteku, čime se čuva tačna kopija originalnog stanja MBR sektora,
* *fclose() -* koristi se za zatvaranje prethodno otvorene datoteke i pravilno završavanje procesa upisa rezervne kopije.

Prva faza implementacije obuhvata otvaranje fizičkog diska i očitavanje boot sektora (LBA 0). Putanja do fizičkog diska (npr. \\.\PhysicalDriveX) definiše se manuelno u izvornom kodu programa, u skladu sa konfiguracijom sistema na kome se program izvršava. Program zatim direktno pristupa tvrdom disku, očitava boot sektor i prikazuje ga u konzolnom prozoru. Ručno definisanje putanje do fizičkog diska omogućava potpunu kontrolu nad ciljnim uređajem i sprečava nenamerni pristup sistemskom disku. Na slici 19 dat je prikaz boot sektora pre skrivanja particije, onako kako ga prikazuje sam program.



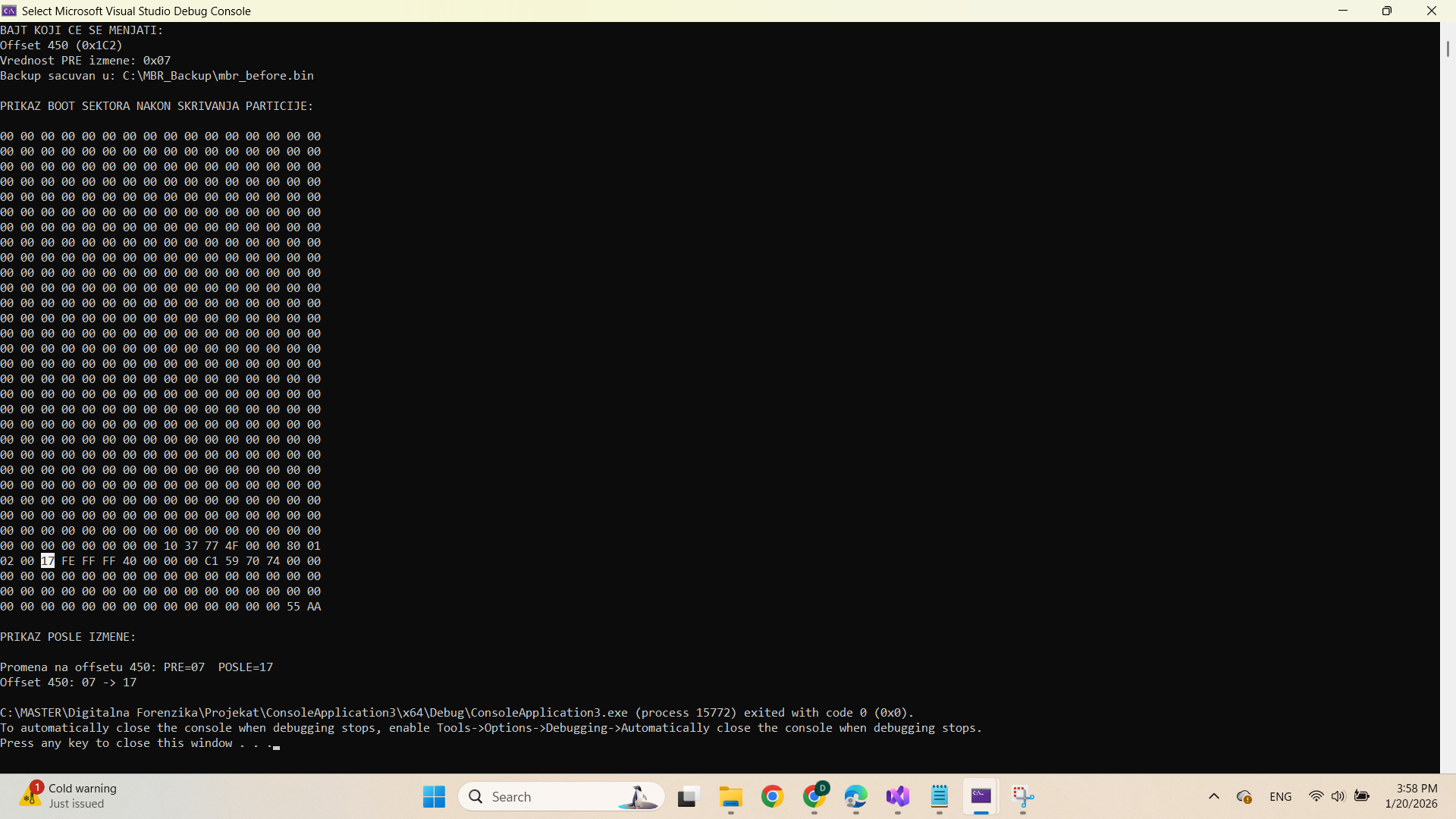
Slika 19 - Boot sektor pre sakrivanja particije (prikaz programa)

Poređenjem slike 19 i slike 15, na kojoj je prikazan boot sektor tvrdog diska korišćenjem HxD programa, može se zaključiti da u prikazu boot sektora nema razlike. Sa slike 19 se vidi da particiona tabela tvrdog diska sadrži heksadecimalne zapise, koji ukazuju na postojanje NTFS particija na disku.

Nakon uspešnog očitavanja boot sektora, u narednoj fazi program vrši kreiranje posebnog direktorijuma namenjenog čuvanju rezervne kopije podataka. Direktorijum pod nazivom *MBR\_Backup* automatski se kreira na sistemskom disku ukoliko prethodno ne postoji. Na ovaj način obezbeđuje se prostor za smeštanje backup datoteka pre izvođenja bilo kakvih izmena nad sadržajem diska.

Kreiranje direktorijuma predstavlja bezbednosnu meru, jer omogućava čuvanje originalnog stanja boot sektora i kasniji povratak sistema u početno stanje u slučaju greške tokom izvršavanja programa.

Nakon očitavanja i prikaza boot sektora u konzolnom prozoru, program vrši sakrivanje particije Maxtor (E:) tako što upisuje heksadecimalnu vrednost 0x17, koja je manuelno definisana u izvornom kodu programa i označava skrivenu NTFS particiju, umesto heksadecimalne vrednosti 0x07 (NTFS particija). Izmena se vrši na *offsetu 450*[[6]](#footnote-6) koji predstavlja decimalni prikaz heksadecimalne vrednosti 0x1C2, u okviru particione tabele, nakon čega se prikazuje izmenjeni boot sektor (slika 20).



Slika 20 - Boot sektor nakon sakrivanja particije (prikaz programa)

Nakon restartovanja računara, particija *Maxtor (E:)* više se ne prikazuje u File Explorer-u. Dobijeni rezultat identičan je rezultatu ostvarenim korišćenjem HxD programa, čime je potvrđeno da je postupak sakrivanja particije uspešno realizovan i programskim putem.

### Ograničenja sakrivanja particija i ponašanje operativnog sistema

Eksperiment je dodatno sproveden nad FAT32 datotečnim sistemom na USB flash uređajima. Izmena MBR particione tabele izvršena je korektno promenom vrednosti polja *Partition Type* na offsetu 450, gde se heksadecimalna vrednost 0x0C (FAT32 LBA particija) zamenjuje vrednošću 0x1C, koja označava skrivenu FAT32 particiju. Ova izmena predstavlja ispravnu i standardnu modifikaciju zapisa u particionoj tabeli.

Međutim, prilikom izvođenja eksperimenta uočen je niz ograničenja koja onemogućavaju uspešnu realizaciju istog postupka u savremenim Windows operativnim sistemima. Na USB uređajima koje Windows operativni sistem prepoznaje kao *removable media*, oznaka skrivene particije se ignoriše i operativni sistem automatski dodeljuje slovo diska, zbog čega particija ostaje vidljiva u File Explorer-u, uprkos korektno izvršenoj izmeni MBR zapisa.

Kako bi se proverilo da li su uočena ograničenja posledica samog datotečnog sistema ili unapređenih bezbednosnih mehanizama operativnog sistema, eksperiment je dodatno testiran u virtuelnom okruženju *VirtualBox*. U tu svrhu kreirane su dve virtuelne mašine sa operativnim sistemima Windows 7 i Windows XP.

Razvijeni program je kompajliran u izvršni (.exe) fajl i pokretan iz *Command Prompt-a* sa administratorskim privilegijama, kako bi se obezbedio direktan pristup fizičkom disku.

Prilikom pokretanja izvršne (.exe) datoteke na Windows 7 operativnom sistemu uočeno je isto ponašanje kao i na Windows 11 sistemu. Nakon izmene MBR particione tabele i restartovanja sistema, particija ostaje vidljiva u File Explorer-u, što ukazuje na to da se ne radi o grešci u implementaciji programa, već o ponašanju samog operativnog sistema.

Na Windows operativnim sistemima verzije 7 i novijim postoje i drugi načini da se particija ukloni iz prikaza u File Explorer-u, kao što su:

* uklanjanje slova diska korišćenjem opcije *Change Drive Letter and Paths* u Disk Management alatu,
* postavljanje diska u *Offline* režim,
* korišćenje sistemskih politika (*Group Policy*) ili izmena u Windows Registry-ju.

Međutim, ove metode nisu korišćene u okviru ovog rada, jer cilj eksperimenta nije bio sakrivanje particije korišćenjem logičkih ili administrativnih mehanizama operativnog sistema, već analiza efekata direktne izmene MBR particione tabele.

Dodatnim testiranjem utvrđeno je da primena oznake skrivene particije zavisi ne samo od datotečnog sistema, već i od tipa skladišnog uređaja. Iako Windows operativni sistem poštuje *hidden partition type* za NTFS particije na eksternim hard diskovima, isto ponašanje nije uočeno kod USB flash uređaja, čak i kada su oni formatirani u NTFS datotečni sistem.

U okviru eksperimenta, USB flash uređaj je formatiran u NTFS datotečni sistem i izvršena je izmena vrednosti polja *Partition Type* u MBR particionoj tabeli sa **0x07** na **0x17**. Međutim, uprkos korektno izvršenoj izmeni, particija je i dalje ostajala vidljiva u File Explorer-u. Ovo ponašanje potvrđuje da Windows operativni sistem USB flash uređaje tretira kao *removable media*, pri čemu ignoriše oznaku skrivene particije bez obzira na korišćeni datotečni sistem.

Nasuprot tome, eksterni hard diskovi, iako se u korisničkom interfejsu često prikazuju kao prenosivi uređaji, u okviru Windows kernel-a se tretiraju kao uređaji klase *fixed disk*. Zbog toga se na njima primenjuju standardna pravila interpretacije MBR particione tabele, uključujući i poštovanje oznake skrivene NTFS particije.

Na osnovu sprovedenih testova može se zaključiti da je sakrivanje particije izmenom *Partition Type* polja u MBR tabeli efikasno isključivo za NTFS particije na fiksnim i eksternim hard diskovima, dok se na USB flash uređajima, bez obzira na datotečni sistem (FAT32 ili NTFS), ova oznaka ignoriše od strane Windows operativnog sistema.

### Forenzičke implikacije manipulacije MBR-a

Direktna izmena MBR particione tabele predstavlja tehniku koja može imati značajne implikacije u kontekstu digitalne forenzike. Iako promena vrednosti Partition Type polja ne utiče na fizičko postojanje podataka na disku, ona menja način na koji operativni sistem interpretira i prikazuje particije korisniku.

U forenzičkom postupku, skrivena particija može biti identifikovana analizom RAW sektora diska, nezavisno od interpretacije operativnog sistema. Upoređivanjem sadržaja MBR sektora sa logičkim prikazom diskova u sistemskim alatima moguće je detektovati neslaganja koja ukazuju na manipulaciju.

Savremeni forenzički alati (npr. FTK, EnCase, Autopsy) vrše analizu fizičke strukture diska i ne oslanjaju se isključivo na informacije koje pruža operativni sistem. Zbog toga izmena Partition Type polja ne predstavlja pouzdanu tehniku prikrivanja podataka u profesionalnom forenzičkom okruženju.

Dodatno, u slučaju diskova inicijalizovanih GPT (GUID Partition Table) šemom particionisanja, ovakav pristup manipulaciji nije primenljiv, jer GPT koristi drugačiju strukturu zapisa i redundantne kopije particione tabele.

Ove činjenice potvrđuju da manipulacija MBR-a može zavarati korisnički interfejs operativnog sistema, ali ne predstavlja efikasnu tehniku prikrivanja podataka u kontekstu detaljne forenzičke analize.

# **ZAKLJUČAK**

U ovom radu *“Forenzika storage medijuma: Hex editor sa RAW pristupom podacima na disku i sakrivanje particije”* analizirana je organizacija podataka na tvrdom disku sa posebnim fokusom na strukturu Master Boot Record-a (MBR) i način na koji operativni sistem interpretiše informacije o particijama. Teorijski deo rada obuhvatio je fizičku i logičku organizaciju diska, kao i ulogu datotečnih sistema u procesu pristupa podacima.

Eksperimentalni deo rada obuhvatio je sakrivanje particija direktnom izmenom MBR particione tabele, primenom heksadecimalnog editora HxD i sopstvenog programa razvijenog u programskom jeziku C/C++. Poređenjem rezultata dobijenih korišćenjem oba pristupa potvrđeno je da se identične izmene nad boot sektorom mogu ostvariti i ručno i programskim putem, što potvrđuje ispravnost implementacije i korektnost postupka.

Daljom analizom utvrđeno je da efikasnost sakrivanja particije zavisi od kombinacije datotečnog sistema, tipa skladišnog uređaja i ponašanja operativnog sistema. Izmena polja Partition Type u MBR tabeli pokazala se kao efikasna metoda za sakrivanje NTFS particija na fiksnim i eksternim hard diskovima, dok se na USB flash uređajima ova oznaka ignoriše od strane Windows operativnog sistema, bez obzira na korišćeni datotečni sistem.

Rezultati testiranja na različitim verzijama Windows operativnog sistema ukazuju na to da savremeni sistemi primenjuju dodatne bezbednosne mehanizme prilikom montiranja prenosivih medijuma, što ima direktan uticaj na forenzičku interpretaciju podataka. Ovakvo ponašanje ne predstavlja grešku u implementaciji, već posledicu dizajna operativnog sistema.

Na osnovu sprovedenih analiza može se zaključiti da je izmena MBR particione tabele validna tehnika u kontekstu digitalne forenzike, ali da njena primena mora biti razmatrana u zavisnosti od tipa uređaja i okruženja u kojem se analiza sprovodi. Razumevanje ovih ograničenja ključno je za pravilno tumačenje dokaza i pouzdanu analizu podataka na nivou fizičkog skladišta.

# **LITERATURA**

[1] <https://racunarskekomponenteblog.wordpress.com/2016/05/10/tvrdi-disk/>

[2] <https://www.linfo.org/raw_disk.html>

[3] <https://android.eng.ankara.edu.tr/wp-content/uploads/sites/656/2018/02/2-Bolum-1.pdf>

[4] [Cory Altheide, Harlan Carvey, "Digital Forensics with Open Source Tools", Syngress; 1 edition (April 28, 2011)](http://www.amazon.com/Digital-Forensics-Open-Source-Tools/dp/1597495867/ref=sr_1_2?ie=UTF8&qid=1385124010&sr=8-2&keywords=digital+forensics)

[5] “Investigating Hard Disks, File and Operating Systems, EC-Council Press”, Cengage Learning, 2010.

[6] “Informatika”, Univerzitet Singidunum, Prof. dr Milan Milosavljević, Prof. dr Mladen Veinović, Doc. dr Gojko Grubor, Beograd 2009.

[7] The Winn L. Rosch Hardware Bible, 6th Edition

[8] “File System Forensic Analysis” by Brian Carrier, 2015.

[9] <https://mh-nexus.de/en/hxd/>

[10] <https://www.virtualbox.org/>

[11] <https://www.geeksforgeeks.org/cpp/how-to-read-from-a-file-in-cpp/>

[12] <https://learn.microsoft.com/en-us/windows/win32/api/fileapi/nf-fileapi-readfile>

[13] <https://learn.microsoft.com/en-us/windows/win32/api/fileapi/nf-fileapi-createfilea>

[14] <https://learn.microsoft.com/en-us/windows/win32/api/fileapi/nf-fileapi-setfilepointerex>

[15] <https://learn.microsoft.com/en-us/windows/win32/api/fileapi/nf-fileapi-writefile>

[16] <https://learn.microsoft.com/en-us/windows/win32/api/handleapi/nf-handleapi-closehandle>

[17] <https://learn.microsoft.com/en-us/windows/win32/api/fileapi/nf-fileapi-createdirectorya>

[18]<https://learn.microsoft.com/en-us/cpp/c-runtime-library/reference/fopen-s-wfopen-s?view=msvc-170>

[19] <https://learn.microsoft.com/en-us/cpp/c-runtime-library/reference/fwrite?view=msvc-170>

[20]<https://learn.microsoft.com/nl-nl/cpp/c-runtime-library/reference/fclose-fcloseall?view=msvc-150>

1. *RAW pristup podacima* - Skup standarda za fizičku konekciju i prenos podataka između računara i perifernih uređaja, razvijen od strane Američkog instituta za nacionalne standarde (engl. American National Standards Institute), koji omogućava direktan (RAW) pristup podacima [↑](#footnote-ref-1)
2. *Master Boot Record (MBR) -* predstavlja prvi sektor na particionisanom disku (LBA 0) koji sadrži osnovne informacije o rasporedu particija. [↑](#footnote-ref-2)
3. *IDE - Integrated Development Environment) -* poznatiji kao ATA ili PATA, predstavlja standardni interfejs za konektovanje matične ploče računara i perifernih uređaja. [↑](#footnote-ref-3)
4. *SCSI - Small Computer System Interface -* Skup standarda za fizičku konekciju i prenos podataka između računara i perifernih uređaja, razvijen od strane Američkog instituta za nacionalne standarde (engl. American National Standards Institute). [↑](#footnote-ref-4)
5. *Master Partition Table (MPT)* predstavlja strukturu podataka na disku koja sadrži informacije o rasporedu particija i omogućava operativnom sistemu da identifikuje i pristupi pojedinačnim particijama. [↑](#footnote-ref-5)
6. **Offset 450** odgovara polju *Partition Type* u prvom particionom unosu MBR particione tabele. Particiona tabela u MBR-u počinje na offsetu 446 bajtova, a polje Partition Type predstavlja 5. bajt unutar svakog 16-bajtnog particionog unosa, tj. offset +4. Zbog toga je apsolutni offset ovog polja jednak zbiru 446 B + 4 B = 450 B, što predstavlja decimalni zapis heksadecimalne vrednosti 0x1C2. [↑](#footnote-ref-6)