

Oprogramowanie systemowe

Projekt – sprawozdanie

Mateusz Kowalczyk, s188717

1. Wprowadzenie

Celem projektu było stworzenie aplikacji umożliwiającej ukrywanie danych na dysku. Wykorzystałem trzy sposoby umożliwiające osiągnięcie założonego rezultatu:

- ukrywanie między logicznym a fizycznym końcem pliku (w tzw. obszarze *slack* nazywanym dalej również *luźnym obszarem*),
- ukrywanie w alternatywnych strumieniach danych (ADS) NTFS,
- ukrywanie poza partycjami,

każdemu poświęcając jeden moduł programu.

Aplikację stworzyłem za pomocą środowiska Visual Studio 2022. Kod napisany został w języku C++ z wykorzystaniem interfejsu *Windows API*. Aby uchronić fizyczny komputer przed skutkami ewentualnych błędów program uruchamiałem w maszynie wirtualnej z systemem operacyjnym *Microsoft Windows Server 2022* stworzonej w programie *Oracle VirtualBox 7.0*.

2. Ukrywanie między logicznym a fizycznym końcem pliku

Dyski oferują blokowy sposób odczytu i zapisu danych. System operacyjny, aby zapisać plik na dysku, alokuje na nim obszar o wielkości stanowiącej wielokrotność jednostki alokacji (klastra). Mechanizm blokowy pozwala na zaadresowanie większej ilości pamięci dyskowej z użyciem tej samej liczby bajtów adresu oraz usprawnia korzystanie z dysków, oferując przepływ większej ilości danych przy jednej operacji. Same dane zapisywane na dysku oczywiście nie zawsze posiadają rozmiar będący wielokrotnością klastra. Z tego powodu między logicznym końcem pliku a rzeczywistym (fizycznym) końcem obszaru dla tego pliku zaalokowanego ma szansę pozostać wolna przestrzeń (luźnym obszar, *slack*). W niej właśnie dane ukrywa pierwsza część programu stworzonego w ramach projektu.

Aplikacja, korzystając z funkcji `GetSystemInfo`, pobiera strukturę `SYSTEM_INFO`, która zawiera pole `dwPageSize` informujące o rozmiarze jednostki alokacji pamięci dyskowej (ze względu na nazwę tego pola jednostkę tę dalej nazywać będę również stroną dyskową). Funkcją `GetFileSize` pobrany zostaje logiczny rozmiar pliku, w którego luźnym obszarze mają zostać ukryte bajty. Na podstawie powyższych dwóch liczb program oblicza przesunięcie bajtowe (*offset*) początku luźnego obszaru względem początku pliku. Koniec luźnego obszaru, znajduje się w oczywisty sposób w miejscu, w którym kończy się strona dyskowa, do której należy początek luźnego obszaru.

Zasadniczym etapem działania tego modułu programu jest stworzenie odwzorowania pliku do pamięci operacyjnej. Aplikacja osiąga to, wykorzystując funkcję `MapViewOfFile`. Zwraca ona wskaźnik na początek odwzorowania w pamięci operacyjnej, który program wykorzystuje do zapisania lub odczytania ukrywanych bajtów. Jeśli użytkownik zdecydował się to pierwsze, zmieniona zawartość pliku zostaje do niego wczytana z powrotem z pamięci głównej dzięki wykorzystaniu funkcji `FlushViewOfFile`.

3. Ukrywanie w ADS NTFS

W systemie plików NTFS z każdy plik posiada co najmniej jeden strumień danych. Podstawowy, nienazwany jest dostępny przez zwyczajne otwarcie pliku, np. w *Eksploratorze plików*. Inne, alternatywne, posiadają nazwy zapisywane po dwukropku. Przykładowo strumień *ads0* w pliku o ścieżce *myFolder\myFile.txt* dostępny byłby pod adresem *myFolder\myFile.txt:ads0*.

Przed programowaniem modułu aplikacji obsługującego alternatywne strumienie danych eksperymentowałem, tworząc je ręcznie. Aby utworzyć przykładowy ADS skorzystałem z polecenie wiersza poleceń:

```
D:\Studia\Semestr V\Oprogramowanie systemowe\Projekt\OS_Proj\datafiles\ads_datafiles\report>dir
Volume in drive D is Dysk główny
Volume Serial Number is 3EF2-82AA

Directory of D:\Studia\Semestr V\Oprogramowanie systemowe\Projekt\OS_Proj\datafiles\ads_datafiles\report

22.01.2024  23:32    <DIR>        .
22.01.2024  23:32    <DIR>        ..
               0 File(s)                0 bytes
               2 Dir(s)  299 813 838 848 bytes free

D:\Studia\Semestr V\Oprogramowanie systemowe\Projekt\OS_Proj\datafiles\ads_datafiles\report>echo ads0 > file0.txt:ads0

D:\Studia\Semestr V\Oprogramowanie systemowe\Projekt\OS_Proj\datafiles\ads_datafiles\report>dir
Volume in drive D is Dysk główny
Volume Serial Number is 3EF2-82AA

Directory of D:\Studia\Semestr V\Oprogramowanie systemowe\Projekt\OS_Proj\datafiles\ads_datafiles\report

22.01.2024  23:32    <DIR>        .
22.01.2024  23:32    <DIR>        ..
22.01.2024  23:32                0 file0.txt
               1 File(s)                0 bytes
               2 Dir(s)  299 813 838 848 bytes free


D:\Studia\Semestr V\Oprogramowanie systemowe\Projekt\OS_Proj\datafiles\ads_datafiles\report>dir /r
Volume in drive D is Dysk główny
Volume Serial Number is 3EF2-82AA

Directory of D:\Studia\Semestr V\Oprogramowanie systemowe\Projekt\OS_Proj\datafiles\ads_datafiles\report

22.01.2024  23:32    <DIR>        .
22.01.2024  23:32    <DIR>        ..
22.01.2024  23:32                0 file0.txt
                7 file0.txt:ads0:$DATA
               1 File(s)                0 bytes
               2 Dir(s)  299 813 838 848 bytes free
```

Aby wypisać również alternatywne strumienie danych, do polecenia *dir* należy dodać opcję */r*. ADS można otworzyć programem *Notatnik* w następujący sposób:

```
D:\Studia\Semestr V\Oprogramowanie systemowe\Projekt\OS_Proj\datafiles\ads_datafiles\report>notepad file0.txt:ads0
D:\Studia\Semestr V\Oprogramowanie systemowe\Projekt\OS_Proj\datafiles\ads_datafiles\report>
```



The screenshot shows a Notepad window titled 'file0.txt:ads0 — Notatnik'. The menu bar includes 'Plik', 'Edycja', 'Format', 'Widok', and 'Pomoc'. The text area contains the string 'ads0'.

W rzeczywistości alternatywnych strumieni danych używa się na przykład do oznaczania pochodzenia pobieranego pliku. Taka informacja zapisywana zostaje w strumieniu o nazwie *Zone.Identifier*. Umieszcza się w nim jeden z poniższych identyfikatorów:

Zone	Zoneld
Local machine	0
Local intranet	1
Trusted sites	2
Internet	3
Restricted sites	4

Przed uruchomieniem programu system operacyjny odczytuje tę daną i podejmuje decyzję, czy rozpoczęcie wykonywania tak oznaczonego programu niesie niebezpieczeństwo.

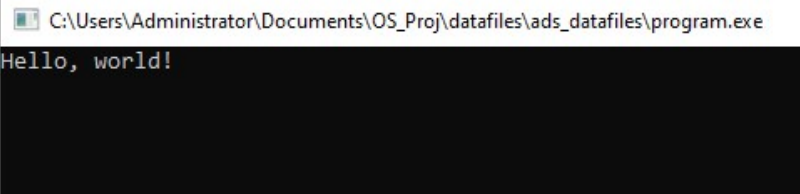
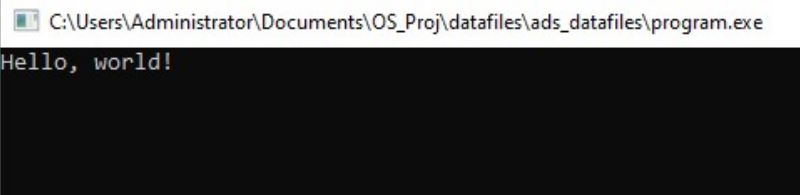
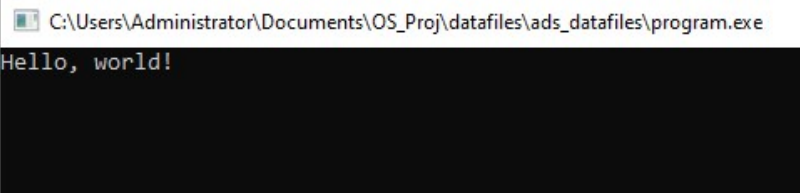
Moduł aplikacji projektowej potrafi, oprócz zapisywania i odczytywania dowolnych alternatywnych strumieni danych (za pomocą standardowych funkcji klasy *ifstream/ofstream*), upraszczać dodawanie do plików strumieni *Zone.Identifier* o następującej postaci:


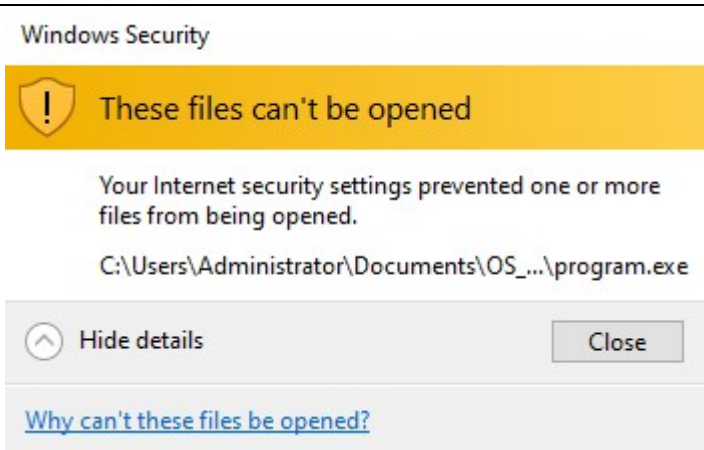
```

program.exe:Zone.Identifier - Notepad
File Edit Format View Help
[ZoneTransfer]
ZoneId=0

```

Próba uruchomienia programu oznaczonego odpowiednimi identyfikatorami stref kończy się następująco:

Zoneld	Rezultat	Komentarz
0		Program uruchomiony bez problemu
1		Program uruchomiony bez problemu
2		Program uruchomiony bez problemu

3		Wyświetlone ostrzeżenie z możliwością podjęcia decyzji o uruchomieniu programu
4		Brak możliwości uruchomienia programu

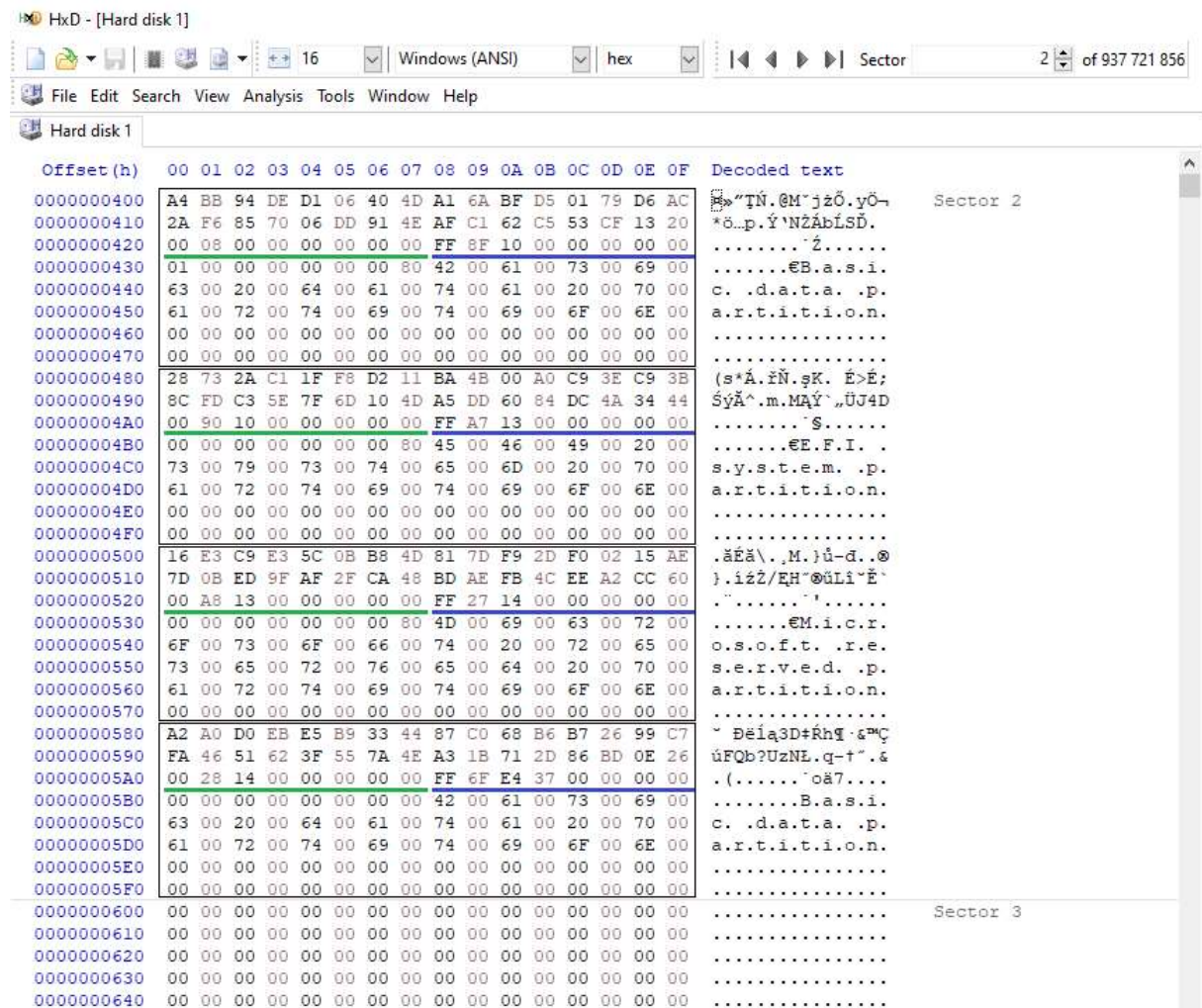
4. Ukrywanie poza partycjami

Podczas podziału dysku na partycje nie cała jego przestrzeń musi zostać wykorzystana. Pewien obszar za ostatnią partycją pozostaje wolny (dalej nazywam go również obszarem niezaalokowanym). W tym obszarze można zapisać dane, które nie będą widoczne dla programów czytających wyłącznie bajty z partycji dyskowych.

Aby móc podejrzeć zawartość całego dysku, łącznie z obszarem niezaalokowanym, skorzystałem z programu *HxD*, w którym otworzyłem dysk fizyczny. Początkowo przeglądałem jego zawartość ręcznie, obliczając miejsce, w którym kończy się ostatnia partycja.

W przypadku dysku o schemacie GPT należało odszukać sektor o logicznym adresie bloku (LBA) równym 2, a następnie, zaczynając od jego początku, odczytać 128-bajtowe wpisy o obecnych na dysku partycjach. W każdym wpisie pod przesunięciem 32 bajtów znajdowało się 8-bajtowe LBA początkowe (podkreślone na rysunku kolorem zielonym), a pod

przesunięciem 24 bajtów – 8-bajtowe LBA końcowe (podkreślone na niebiesko) danej partycji.



```
Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F Decoded text
0000000400 A4 BB 94 DE D1 06 40 4D A1 6A BF D5 01 79 D6 AC 0000000410 2A F6 85 70 06 DD 91 4E AF C1 62 C5 53 CF 13 20 0000000420 00 08 00 00 00 00 00 00 FF 8F 10 00 00 00 00 00 0000000430 01 00 00 00 00 00 00 80 42 00 61 00 73 00 69 00 0000000440 63 00 20 00 64 00 61 00 74 00 61 00 20 00 70 00 0000000450 61 00 72 00 74 00 69 00 74 00 69 00 6F 00 6E 00 0000000460 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0000000470 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0000000480 28 73 2A C1 1F F8 D2 11 BA 4B 00 A0 C9 3E C9 3B 0000000490 8C FD C3 5E 7F 6D 10 4D A5 DD 60 84 DC 4A 34 44 00000004A0 00 90 10 00 00 00 00 00 FF A7 13 00 00 00 00 00 00000004B0 00 00 00 00 00 00 00 80 45 00 46 00 49 00 20 00 00000004C0 73 00 79 00 73 00 74 00 65 00 6D 00 20 00 70 00 00000004D0 61 00 72 00 74 00 69 00 74 00 69 00 6F 00 6E 00 00000004E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00000004F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0000000500 16 E3 C9 E3 5C 0B B8 4D 81 7D F9 2D F0 02 15 AE 0000000510 7D 0B ED 9F AF 2F CA 48 BD AE FB 4C EE A2 CC 60 0000000520 00 A8 13 00 00 00 00 00 00 FF 27 14 00 00 00 00 00 0000000530 00 00 00 00 00 00 00 80 4D 00 69 00 63 00 72 00 0000000540 6F 00 73 00 6F 00 66 00 74 00 20 00 72 00 65 00 0000000550 73 00 65 00 72 00 76 00 65 00 64 00 20 00 70 00 0000000560 61 00 72 00 74 00 69 00 74 00 69 00 6F 00 6E 00 0000000570 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 0000000580 A2 A0 D0 EB E5 B9 33 44 87 C0 68 B6 B7 26 99 C7 0000000590 FA 46 51 62 3F 55 7A 4E A3 1B 71 2D 86 BD 0E 26 00000005A0 00 28 14 00 00 00 00 00 FF 6F E4 37 00 00 00 00 00 00000005B0 00 00 00 00 00 00 00 42 00 61 00 73 00 69 00 00000005C0 63 00 20 00 64 00 61 00 74 00 61 00 20 00 70 00 00000005D0 61 00 72 00 74 00 69 00 74 00 69 00 6F 00 6E 00 00000005E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00000005F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
000000600 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 000000610 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 000000620 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 000000630 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 000000640 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

Pierwsze LBA następujące po ostatnim LBA ostatniej partycji stanowi początek obszaru niezaalokowanego. Należy jednak pamiętać, iż ostatnie 33 sektory dysku zajmuje obszar *Secondary GPT*.

W przypadku dysku o schemacie MBR należało odczytać 64-bajtową tablicę partycji znajdującą się w sektorze o LBA = 0. Należą do niej bajty bezpośrednio przed ostatnimi dwoma bajtami (55 AA) tego sektora. W każdym 16-bajtowym wpisie przedostatnie 4 bajty (podkreślone kolorem zielonym) zajmuje LBA początkowe, natomiast ostatnie 4 bajty (zaznaczone na niebiesko) – liczba sektorów danej partycji.

Na podstawie liczby sektorów zajmowanych przez ostatnią partycję można obliczyć początek obszaru niezaalokowanego.

Po zebraniu tych informacji i wykonaniu ręcznych eksperymentów rozpocząłem pracę nad ostatnim modułem aplikacji projektowej. Celem było zaprogramowanie zestawu funkcji, które otwierają dysk fizyczny jako plik, odczytują informacje o nim oraz zapisują dane poza partycją lub odczytują dane spoza partycji.

W celu otwarcia dysku fizycznego jako plik aplikacja używa funkcji `CreateFileA` z podanym jako nazwa pliku napisem w formacie `\\.\PhysicalDriveX`, gdzie *X* to numer dysku fizycznego. Następnie program za pomocą funkcji `DeviceIoControl` pobiera strukturę `DISK_GEOMETRY_EX`, która zawiera informację o geometrii dysku. Odczytana zostaje z niej liczba bajtów przypadająca na sektor oraz rozmiar dysku. Kolejnym wywołaniem funkcji `DeviceIoControl` zostaje pobrana struktura `DRIVE_LAYOUT_INFORMATION_EX`. Ta zawiera informację o schemacie partycjonowania. W przypadku wykrycia schematu GPT od rozmiaru dysku odejmowana jest liczba sektorów przypadająca na *Secondary GPT*, tak aby w późniejszym etapie nie nadpisać tego obszaru przy zapisywaniu danych za ostatnią partycją aż do końca dysku. Ze struktury `DRIVE_LAYOUT_INFORMATION_EX` program odczytuje w kolejnym kroku dla każdej partycji jej początek i długość. Na tej podstawie określa, pod jakim przesunięciem bajtowym kończy się ostatnia. Za tym punktem rozpoczyna się obszar niezaalokowany.

Na podstawie rozmiaru danych do zapisu/odczytu aplikacja tworzy w pamięci operacyjnej bufor o rozmiarze stanowiącym wielokrotność rozmiaru sektora na dysku. Jeśli użytkownik zdecydował się na zapisanie danych na dysk, zostają one wpisane do tego bufora. Następnie za pomocą funkcji `SetFilePointerEx` wskaźnik odczytu/zapisu ustawiany jest w odpowiednie miejsce, a z użyciem funkcji `ReadFile/WriteFile` dane zostają zapisane w obszarze za partycjami lub z niego odczytane.