02

Obfuskacja kodu

ZACIEMNIANIE KODU

hacking, cyberbezpieczeństwo i sztuczna inteligencja

opracowanie: Michał Suchoń

NEXT

Przypomnienie...

I. Co to funkcja skrótu?

II. Gdzie wykorzystujemy funkcje skrótu?

III. Podaj najpopularniejsze funkcje skrótu

*IV. Co to crc / suma kontrolna?

Przypomnienie...

I. Co to funkcja skrótu?

Inaczej: funkcja hashująca – zamienia ciąg jawny na nieodwracalną pseudolosową zbitkę znaków (hash)

II. Gdzie wykorzystujemy funkcje skrótu?

III. Podaj najpopularniejsze funkcje skrótu

*IV. Co to crc / suma kontrolna?

Przypomnienie...

I. Co to funkcja skrótu?

Inaczej: funkcja hashująca – zamienia ciąg jawny na nieodwracalną pseudolosową zbitkę znaków (hash)

II. Gdzie wykorzystujemy funkcje skrótu?

Przy zapisanie hasła, np. w bazie danych, w celu uniknięcia przechowywania haseł w postaci jawnej (co podczas wycieku stanowi pożywkę dla cyberprzestępców)

III. Podaj najpopularniejsze funkcje skrótu

*IV. Co to crc / suma kontrolna?

Przypomnienie...

I. Co to funkcja skrótu?

Inaczej: funkcja hashująca – zamienia ciąg jawny na nieodwracalną pseudolosową zbitkę znaków (hash)

II. Gdzie wykorzystujemy funkcje skrótu?

Przy zapisanie hasła, np. w bazie danych, w celu uniknięcia przechowywania haseł w postaci jawnej (co podczas wycieku stanowi pożywkę dla cyberprzestępców)

III. Podaj najpopularniejsze funkcje skrótu

SHA-256, MD-5, BCRYPT

*IV. Co to crc / suma kontrolna?

Przypomnienie...

I. Co to funkcja skrótu?

Inaczej: funkcja hashująca – zamienia ciąg jawny na nieodwracalną pseudolosową zbitkę znaków (hash)

II. Gdzie wykorzystujemy funkcje skrótu?

Przy zapisanie hasła, np. w bazie danych, w celu uniknięcia przechowywania haseł w postaci jawnej (co podczas wycieku stanowi pożywkę dla cyberprzestępców)

III. Podaj najpopularniejsze funkcje skrótu

SHA-256, MD-5, BCRYPT

*IV. Co to crc / suma kontrolna?

(CRC – *Cyclic Redundancy Code*); **cykliczny kod nadmiarowy** - zestaw sum kontrolnych (wartość zapewniająca integralność danych), używany do wykrywania przypadkowych błędów powstałych podczas przesyłania danych lub ich przechowywania w postaci binarnej

Małe wyzwanie na start...

Napisz prosty program w C#, obliczający pole trójkąta. Podstawa trójkąta ma 10 cm, wysokość 5 cm.

Wskazówki:

- Wzór na pole trójkąta: $P = \frac{1}{2}(a * h)$
- Wypisanie w konsoli: **Console.WriteLine(…)**
- Wczytanie danych od użytkownika: **Console.ReadLine(…)**
- Zatrzymanie programu dopiero po kliknięciu klawisza: **Console.ReadKey(…)**
- Parsowanie wczytywanego stringa na typ liczbowy: <typ liczbowy>.Parse(Console.ReadLine())
- Rzutowanie z jednego typu liczbowego na drugi: (<typ, do którego dążymy>) <zmienna typu użytego na początku>;
  np. *int a = 5; float b; b = (float) a;*

CZAS NA ZADANIE: 5-10 MINUT

Możliwe rozwiązania:

```
float pole, a, h;

Console.WriteLine("1. Podaj długość podstawy [cm]:");
a = float.Parse(Console.ReadLine());


Console.WriteLine("1. Podaj długość wysokości [cm]:");
h = float.Parse(Console.ReadLine());

pole = 0.5f * (a * h);

Console.WriteLine($"1 WYNIK: Pole trójkąta wynosi: {pole} cm");

Console.ReadKey();
```

```
try
{
    Console.WriteLine("4. Podaj długość podstawy [cm]:");
    a = float.Parse(Console.ReadLine());


    Console.WriteLine("4. Podaj długość wysokości [cm]:");
    h = float.Parse(Console.ReadLine());

}
catch (FormatException)
{
    Console.WriteLine("4. Błąd: podana wartość nie jest liczbą zmiennoprzecinkową.");
}
catch (OverflowException)
{
    Console.WriteLine("4. Błąd: podana liczba jest za duża lub za mała.");
}

pole = 0.5f * (a * h);

Console.WriteLine($"4 WYNIK: Pole trójkąta wynosi: {pole} cm");

Console.ReadKey();
```

```
Console.WriteLine("3. Podaj długość podstawy [cm]:");
a = float.TryParse(Console.ReadLine(), out a) ? a : 0f;


Console.WriteLine("3. Podaj długość wysokości [cm]:");
h = float.TryParse(Console.ReadLine(), out h) ? h : 0f;

pole = 0.5f * (a * h);

Console.WriteLine($"3 WYNIK: Pole trójkąta wynosi: {pole} cm");


Console.ReadKey();
```

```
a = 5;
h = 8;

pole = 0.5f * (a * h);

Console.WriteLine($"2 WYNIK: Pole trójkąta wynosi: {pole} cm");


Console.ReadKey();
```

Czy wszystkie rozwiązania są dobre?

Czy każdy program robi to samo?

Czy są jakieś różnice w programie?
Jeśli tak, jakie?

Czy dla komputera programy są identyczne?

Z czego wynikają te różnice?
Czy można to jakoś wykorzystać w hackingu?

ZACIEMNIANIE KODU OBFUSKACJA

Technika przekształcania programów w taki sposób, że zachowuje się ich semantykę (wszystkie funkcje i działanie programu pozostają zachowane), natomiast utrudnia się jego zrozumienie i analizę.

Jak zaciemnić kod?

TECHNIKI ZACIEMNIANIA KODU

TRANSFORMACJA WYGLĄDU

TECHNIKI ZACIEMNIANIA KODU

TRANSFORMACJA WYGLĄDU

zmiana nazw identyfikatorów (np. zmiennych), formatowania, usuwanie komentarzy

TRANSFORMACJA DANYCH

TECHNIKI ZACIEMNIANIA KODU

TRANSFORMACJA WYGLĄDU

zmiana nazw identyfikatorów (np. zmiennych), formatowania, usuwanie komentarzy

TRANSFORMACJA KONTROLI

TRANSFORMACJA DANYCH

rozdzielanie zmiennych, konwersja statycznych danych do procedury, zmiana kodowania, zmiana długości życia zmiennej, łączenie zmiennych skalarnych, zmiana relacji dziedziczenia, rozłam bądź łączenie tablic, zmiana porządku instancji zmiennych, metod, tablic...

TECHNIKI ZACIEMNIANIA KODU

TRANSFORMACJA WYGLĄDU

zmiana nazw identyfikatorów (np. zmiennych), formatowania, usuwanie komentarzy

TRANSFORMACJA KONTROLI

zmiana przebiegu, rozszerzanie warunków pętli, zmiana kolejności poleceń / pętli / wyrażeń, metody inline, ogólnikowe wyrażenia, klonowanie metod

TRANSFORMACJA DANYCH

rozdzielanie zmiennych, konwersja statycznych danych do procedury, zmiana kodowania, zmiana długości życia zmiennej, łączenie zmiennych skalarnych, zmiana relacji dziedziczenia, rozłam bądź łączenie tablic, zmiana porządku instancji zmiennych, metod, tablic...

Obfuskacja - pros and cons

ZALETY (+)

- szybszy czas ładowania
- zmniejszone zużycie pamięci
- ochrona tajemnic biznesowych
- zapobieganie obchodzeniu zabezpieczeń

WADY (-)

- jedyne zabezpieczenie
- debugowanie
- przenośność
- mechanizm refleksji

Obfuskacja - pros and cons

ZALETY (+)

- szybszy czas ładowania – skrypty przesyłane przez sieć (klient-serwer) im są mniejsze, tym szybsze ich pobieranie
- zmniejszone zużycie pamięci – skrypty starszych języków jak BASIC wykonywały się szybciej i wymagały mniej RAM'u, jeśli nazwy zmiennych były jednoliterowy, nie było komentarze a puste znaki były używane tylko tam, gdzie były niezbędne
- ochrona tajemnic biznesowych – utrudnione zrozumienie kodu sprzyja zapobiegnięciu jego modyfikacjom
- zapobieganie obchodzeniu zabezpieczeń – dzięki temu zakrywanie lub przerabianie mechanizmów licencyjnych jest cięższe. Utrudnia to też hackowanie gier wieloosobowych

WADY (-)

- jedyne zabezpieczenie – zaciemnianie nie zastąpi nam mechanizmów kryptograficznych przy hashujących!
- debugowanie – jest niezwykle trudne przy zaciemnionym kodzie; nazwy zmiennych mogą wydawać się bez sensu, składnia i struktura nie do poznania (debugguje się wówczas dwie wersje kompilacji – przed i po obfuskacji)
- przenośność – silnie zależy od specyficznych cech dla danego kompilatora czy środowiska uruchomieniowego. Jakakolwiek zmiana w tym zakresie utrudnia proces obfuskacji
- mechanizm refleksji – pozwala na inspekcję klas i wywoływanie ich metod wyłącznie po nazwach. W systemach z tym mechanizmem po zaciemnieniu staje się to niemożliwe, gdyż nazwy nie mogą zostać zmieniony przez obfuskator

Przykłady obfuskacji kodu

SPAM

Dawniej antywirusy nie były tak rozwinięte jak dzisiaj i poprzez prostą manipulację treści za pomocą HTML’a i CSS’a można było obejść zabezpieczenia czy filtry.

Powiedzmy, że nasz filtr antyspamowy na poczcie uznaje słowo „OMG” jako niebezpieczne. Oznacza to, że linijka w stylu ** OMG ** zostanie zablokowana.

Jednak jeśli zapisze się ją w taki sposób: ** O M G ** - filtr nie wykrywał zakazanej frazy!

Podobnie z użyciem CSS’a:

```
<p style="font-size: 0px;">
    <span style="font-size: 20px;">O</span>"laf"
    <span style="font-size: 20px;">M</span>"ichał"
    <span style="font-size: 20px;">G</span>"osia"
</p>
```

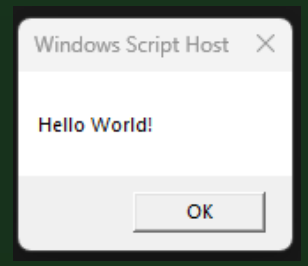
SPAM

Dzisiaj natomiast wiele algorytmów jest na to uodporniona, dlatego dużo skuteczniejsze okazało się **modyfikowanie czcionek**.

Zazwyczaj, gdy widzimy w mailu w załączniku plik z rozszerzeniem **.exe** – wiemy by go nie klikać. **A co z nietypowymi rozszerzeniami, jak np. VBE?**

Gdyby w mailu zostało wysłane niebezpieczne oprogramowanie w postaci zwykłego skryptu VBScript, to byłoby widać jego kod.

Dlatego też korzysta się z kodowanego pliku VBS, właśnie z rozszerzeniem VBE (VBScript Encoded), gdyż zamiast kodu zobaczymy zbitek losowych znaków. **W ten sposób Microsoft zabezpieczał skrypty VBS przed ich niepożądaną redystrybucją.**




```
Wscript.Echo "Hello World!"
```

Prosta instrukcja w Visual Basic'u, wyświetlającą napis „Hello World" -> zapisana w **.vbs**

```
#@~^.*:d{:?VX~<V.;WX==^#~@
```

Ta sama instrukcja, utworzona i zapisana w **.vbe**

WAŻNE! DO ZROBIENIA PLIKU .VBE Z .VBS POTRZEBUJEMY PROGRAMU -> screnc.exe (niewspierany już przez Microsoft, ale wciąż dostępny w sieci) bądź SKRYPT W POWERSHELLU

Przykładowy skrypt:
https://pastebin.com/LayH5S4j

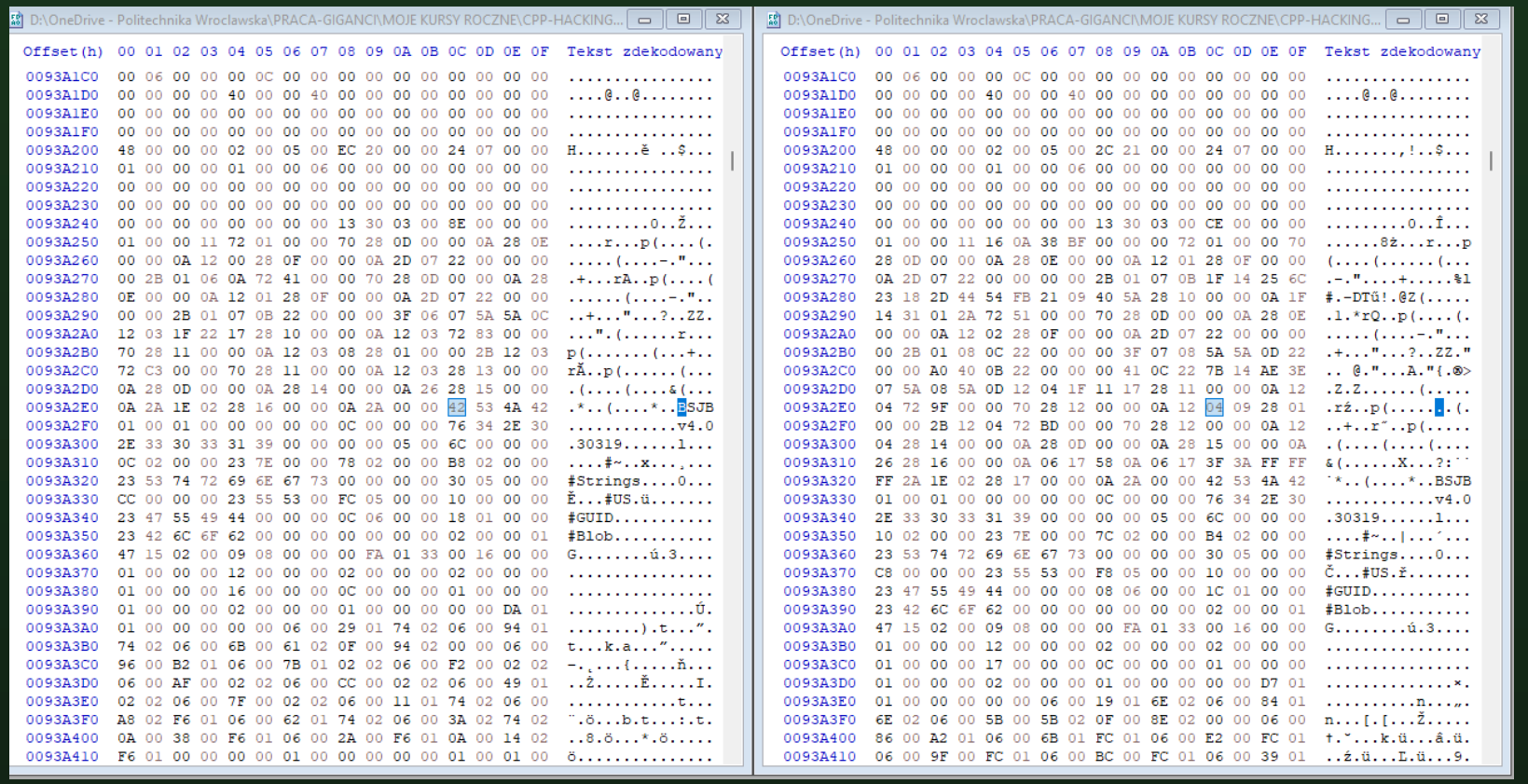
Task 1 | Analiza Pliku .exe apki

- Pobieramy program o nazwie HxD (HexEdytor) -> https://mh-nexus.de/downloads/HxDPortableSetup.zip
- Tworzymy build'a naszej aplikacji z początku zajęć (wykonywalny .exe) lub jeśli nie wiemy jak, to można skorzystać z wersji w katalogu bin > Debug
- Otwieramy nasz plik .exe przez HxD (Dobrą praktyką jest przenieść builda jeśli mamy w inne miejsce by nie uległ żadnej modyfikacji)
- Modyfikujemy nasz program, zmieniając nazwy zmiennych, dodać śmieciowy kod, zamknąć program w pętli, która wykona się raz, zapiszemy, że podstawa nie ma 10 cm a 2 * 5 itd.
- Ponownie kompilujemy aplikację i wrzucamy do HxD – porównujemy!

Polecenie do tworzenia builda:

```
dotnet publish -r win-x64 -p:PublishSingleFile=true --self-contained true -o <nazwa katalogu do builda>
```

Task 1 - rozwiązanie



D:\OneDrive - Politechnika Wroclawska\PRACA-GIGANCI\MOJE KURSY ROCZNE\CPP-HACKING...

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Tekst zdekodowany
0093A1C0	00	06	00	00	00	0C	00	00	00	00	00	00	00	00	00	00	.....
0093A1D0	00	00	00	00	40	00	00	40	00	00	00	00	00	00	00	00	....@..@.....
0093A1E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0093A1F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0093A200	48	00	00	00	02	00	05	00	EC	20	00	00	24	07	00	00	H.....ě ..$...
0093A210	01	00	00	00	01	00	00	06	00	00	00	00	00	00	00	00	.....
0093A220	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0093A230	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0093A240	00	00	00	00	00	00	00	00	13	30	03	00	8E	00	00	00	.....0..Ž...
0093A250	01	00	00	11	72	01	00	00	70	28	0D	00	00	0A	28	0E	....r...p(....(.
0093A260	00	00	0A	12	00	28	0F	00	00	0A	2D	07	22	00	00	00	.....(....-."...
0093A270	00	2B	01	06	0A	72	41	00	00	70	28	0D	00	00	0A	28	.+...rA..p(....(
0093A280	0E	00	00	0A	12	01	28	0F	00	00	0A	2D	07	22	00	00	......(....-."...
0093A290	00	00	2B	01	07	0B	22	00	00	00	3F	06	07	5A	5A	0C	..+..."...?..ZZ.
0093A2A0	12	03	1F	22	17	28	10	00	00	0A	12	03	72	83	00	00	...".(......r...
0093A2B0	70	28	11	00	00	0A	12	03	08	28	01	00	00	2B	12	03	p(.......(...+..
0093A2C0	72	C3	00	00	70	28	11	00	00	0A	12	03	28	13	00	00	rĂ..p(......(...
0093A2D0	0A	28	0D	00	00	0A	28	14	00	00	0A	26	28	15	00	00	.(....(....&(...
0093A2E0	0A	2A	1E	02	28	16	00	00	0A	2A	00	00	42	53	4A	42	.*..(....*..BSJB
0093A2F0	01	00	01	00	00	00	00	00	0C	00	00	00	76	34	2E	30	............v4.0
0093A300	2E	33	30	33	31	39	00	00	00	00	05	00	6C	00	00	00	.30319......l...
0093A310	0C	02	00	00	23	7E	00	00	78	02	00	00	B8	02	00	00	....#~..x...¸...
0093A320	23	53	74	72	69	6E	67	73	00	00	00	00	30	05	00	00	#Strings....0...
0093A330	CC	00	00	00	23	55	53	00	FC	05	00	00	10	00	00	00	Ě...#US.ü.....
0093A340	23	47	55	49	44	00	00	00	0C	06	00	00	18	01	00	00	#GUID..........
0093A350	23	42	6C	6F	62	00	00	00	00	00	00	00	02	00	00	01	#Blob..........
0093A360	47	15	02	00	09	08	00	00	00	FA	01	33	00	16	00	00	G........ú.3....
0093A370	01	00	00	00	12	00	00	00	02	00	00	00	02	00	00	00	.....
0093A380	01	00	00	00	16	00	00	00	0C	00	00	00	01	00	00	00	.....
0093A390	01	00	00	00	02	00	00	00	01	00	00	00	00	00	DA	01	.....Ú.
0093A3A0	01	00	00	00	00	00	06	00	29	01	74	02	06	00	94	01	........).t...".
0093A3B0	74	02	06	00	6B	00	61	02	0F	00	94	02	00	00	06	00	t...k.a..."....
0093A3C0	96	00	B2	01	06	00	7B	01	02	02	06	00	F2	00	02	02	-.¸...{.....ň...
0093A3D0	06	00	AF	00	02	02	06	00	CC	00	02	02	06	00	49	01	..Ž.....Ě.....I.
0093A3E0	02	02	06	00	7F	00	02	02	06	00	11	01	74	02	06	00	..........t....
0093A3F0	A8	02	F6	01	06	00	62	01	74	02	06	00	3A	02	74	02	¨.ö...b.t...:.t.
0093A400	0A	00	38	00	F6	01	06	00	2A	00	F6	01	0A	00	14	02	..8.ö...*.ö.....
0093A410	F6	01	00	00	00	00	01	00	00	00	00	00	01	00	01	00	ö..............

D:\OneDrive - Politechnika Wroclawska\PRACA-GIGANCI\MOJE KURSY ROCZNE\CPP-HACKING...

Offset(h)	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	Tekst zdekodowany
0093A1C0	00	06	00	00	00	0C	00	00	00	00	00	00	00	00	00	00	.....
0093A1D0	00	00	00	00	40	00	00	40	00	00	00	00	00	00	00	00	....@..@.....
0093A1E0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0093A1F0	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0093A200	48	00	00	00	02	00	05	00	2C	21	00	00	24	07	00	00	H.....,!..$...
0093A210	01	00	00	00	01	00	00	06	00	00	00	00	00	00	00	00	.....
0093A220	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0093A230	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	.....
0093A240	00	00	00	00	00	00	00	00	13	30	03	00	CE	00	00	00	.....0..Î...
0093A250	01	00	00	11	16	0A	38	BF	00	00	00	72	01	00	00	70	......8ż...r...p
0093A260	28	0D	00	00	0A	28	0E	00	00	0A	12	01	28	0F	00	00	(....(......(...
0093A270	0A	2D	07	22	00	00	00	2B	01	07	0B	1F	14	25	6C		.-."....+.....%l
0093A280	23	18	2D	44	54	FB	21	09	40	5A	28	10	00	00	0A	1F	#.-DTű!.@Z(.....
0093A290	14	31	01	2A	72	51	00	00	70	28	0D	00	00	0A	28	0E	.1.*rQ..p(....(.
0093A2A0	00	00	0A	12	02	28	0F	00	00	0A	2D	07	22	00	00	00	.....(....-."...
0093A2B0	00	2B	01	08	0C	22	00	00	00	3F	07	08	5A	5A	0D	22	.+..."...?..ZZ."
0093A2C0	00	00	A0	40	0B	22	00	00	00	41	0C	22	7B	14	AE	3E	.. @."...A."{.®>
0093A2D0	07	5A	08	5A	0D	12	04	1F	11	17	28	11	00	00	0A	12	.Z.Z......(.....
0093A2E0	04	72	9F	00	00	70	28	12	00	00	0A	12	04	09	28	01	.rź..p(.....(.
0093A2F0	00	00	2B	12	04	72	BD	00	00	70	28	12	00	00	0A	12	..+..r˝..p(.....
0093A300	04	28	14	00	00	0A	28	0D	00	00	0A	28	15	00	00	0A	.(....(....(....
0093A310	26	28	16	00	00	0A	06	17	58	0A	06	17	3F	3A	FF	FF	&(......X...?:˙˙
0093A320	FF	2A	1E	02	28	17	00	00	0A	2A	00	00	42	53	4A	42	˙*..(....*..BSJB
0093A330	01	00	01	00	00	00	00	00	0C	00	00	00	76	34	2E	30	............v4.0
0093A340	2E	33	30	33	31	39	00	00	00	00	05	00	6C	00	00	00	.30319......l...
0093A350	10	02	00	00	23	7E	00	00	7C	02	00	00	B4	02	00	00	....#~..|...´...
0093A360	23	53	74	72	69	6E	67	73	00	00	00	00	30	05	00	00	#Strings....0...
0093A370	C8	00	00	00	23	55	53	00	F8	05	00	00	10	00	00	00	Č...#US.ř.....
0093A380	23	47	55	49	44	00	00	00	08	06	00	00	1C	01	00	00	#GUID..........
0093A390	23	42	6C	6F	62	00	00	00	00	00	00	00	02	00	00	01	#Blob..........
0093A3A0	47	15	02	00	09	08	00	00	00	FA	01	33	00	16	00	00	G........ú.3....
0093A3B0	01	00	00	00	12	00	00	00	02	00	00	00	02	00	00	00	.....
0093A3C0	01	00	00	00	17	00	00	00	0C	00	00	00	01	00	00	00	.....
0093A3D0	01	00	00	00	02	00	00	00	01	00	00	00	00	00	D7	01	.....×.
0093A3E0	01	00	00	00	00	00	00	06	00	19	01	6E	02	06	00	84	.....n...„.
0093A3F0	6E	02	06	00	5B	00	5B	02	0F	00	8E	02	00	00	06	00	n...[.[...Ž.....
0093A400	86	00	A2	01	06	00	6B	01	FC	01	06	00	E2	00	FC	01	†.˘...k.ü...â.ü.
0093A410	06	00	9F	00	FC	01	06	00	BC	00	FC	01	06	00	39	01	..ź.ü...L.ü...9.

Przed obfuskacją

Algorytm	Suma kontrolna
Checksum-32	40231756

Po obfuskacji

Algorytm	Suma kontrolna
Checksum-32	40231469

Task 2 | Antywirus a aplikacje

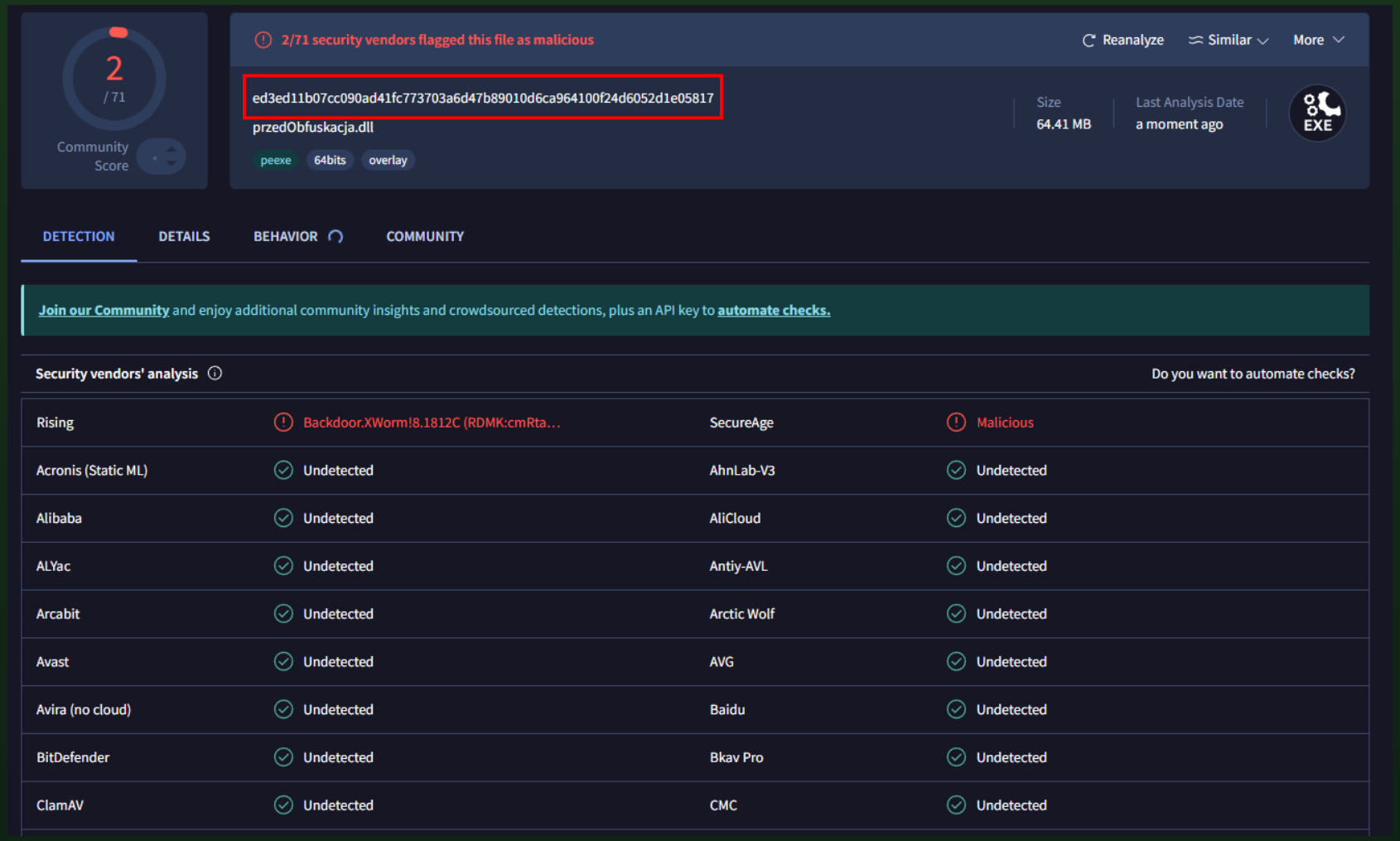
Przez zaciemnienie kodu, utrudniamy działanie antywirusa (potrzebuje on więcej wyliczeń, aby zweryfikować program).

Zwykle antywirusy identyfikują złośliwe oprogramowanie na dwa sposoby: używając tzw. sygnatur lub na podstawie zachowania podejrzanego programu w systemie operacyjnym. Wykrywanie za pomocą sygnatur działa podobnie do systemu odpornościowego człowieka. Antywirus skanuje system w poszukiwaniu charakterystycznych elementów, które zawiera złośliwe oprogramowanie. W celu ich odnalezienia, korzysta z bazy danych złośliwych próbek. Jeżeli na dysku komputera zostanie odnalezione coś, co pasuje do którejkolwiek z nich, antywirus stara się to zneutralizować.

Wrzućmy więc obie aplikacje (przed i po obfuskacji) na stronie **https://www.virustotal.com/gui/home/upload** i porównajmy je ze sobą

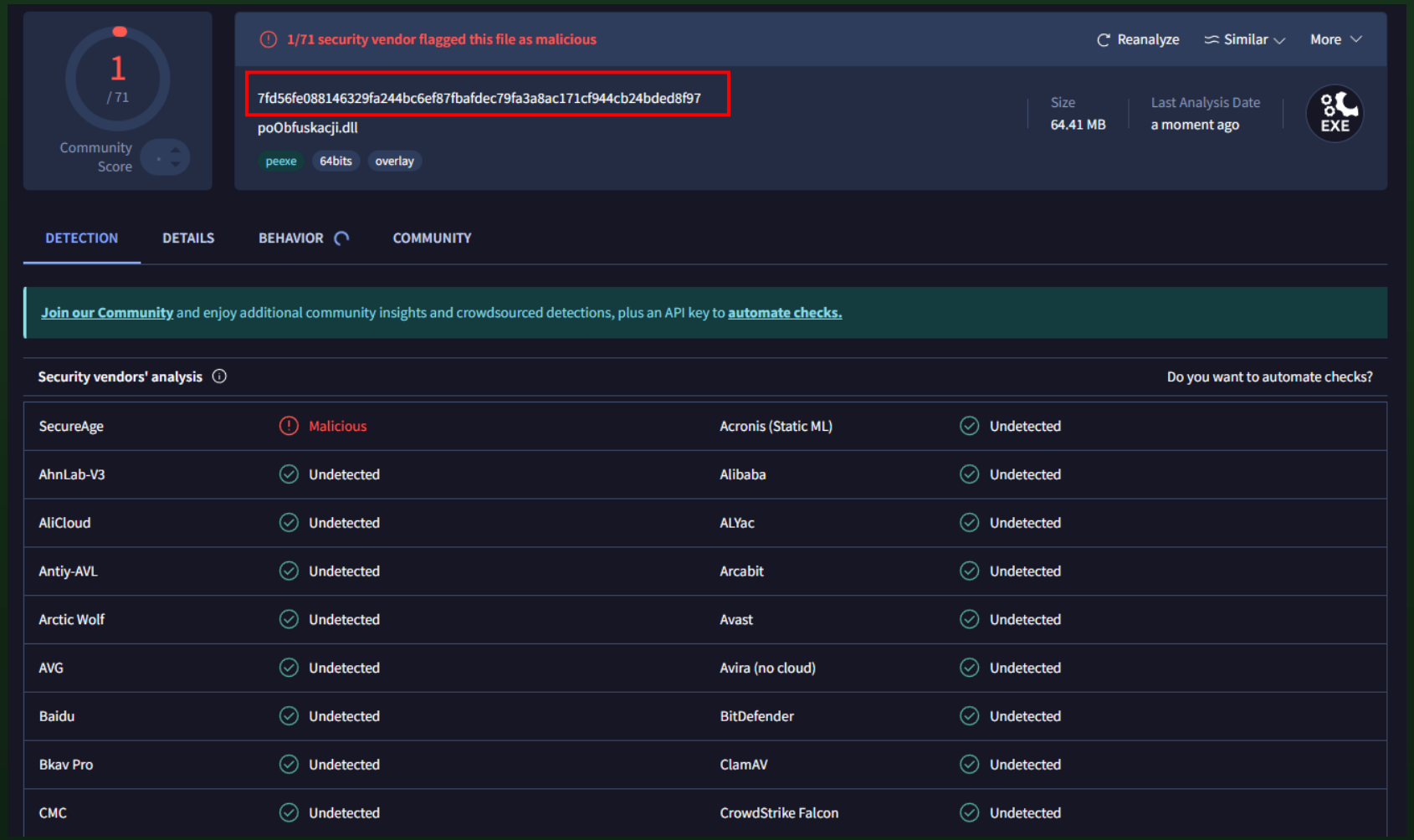
Task 2 - rozwiązanie

Przed obfuskacją



Task 2 - rozwiązanie

Po obfuskacji



Podsumowanie

I. Jak nazywa się program do przeglądania plików exe?

II. Jakimi sposobami możemy zmienić aplikacje dla antywirusa, ale pozostawiając je działanie bez zmian?

III. Jak działa antywirus?

EXIT