|  |  |
| --- | --- |
| **Politechnika Białostocka**  **Wydział Informatyki** | Data: 10.10.2016 |
| **Przedmiot:** Techniki zapewniania poufności w Internecie.  **Sprawozdanie nr:** 6  **Temat:** Obfunkcje  **Autor:** Maciej Ziniewicz  **Studia:** stacjonarne II stopnia, semestr 2 | **Prowadzący:**  prof. dr hab. Vyacheslav Yarmolik  Ocena: |

Spis treści

[1. Treść zadania 2](#_Toc464143197)

[2. Część teoretyczna 2](#_Toc464143198)

[3. Rozwiązanie 3](#_Toc464143199)

# Treść zadania

Zaprojektuj aplikację do zaciemniania kodu w oparciu o trzy techniki zaciemniania: usuwaniu formatowania, usuwaniu komentarzy i zamianie identyfikatorów.

а) język oprogramowania – dowolny.

b) zamianie identyfikatora na losowy kod, naprzykład l01010100

# Część teoretyczna

Zaciemnianie kodu, czyli obfuskacja jest techniką przekształcania kodu źródłowego oprogramowania, zachowującą jego semantykę, ale w znaczący sposób utrudniającą jego analizę i tym samym zrozumienie. Jest to więc zamierzone działanie, które ma na celu ochronę własności intelektualnej z zachowaniem pełnej funkcjonalności. Mówiąc najprościej jest to próba ukrycia sposobu działania zabezpieczanego programu Obfuskacja kodu służy głównie inżynierii wstecznej (ang. reverse engineering) czyli procesowi badania programu komputerowego w celu ustalenia sposobu jego działania, a także sposobu i kosztu jego wykonania, zazwyczaj prowadzonemu w celu zdobycia informacji niezbędnych do skonstruowania odpowiednika. Upowszechnienie się oprogramowania dystrybuowanego w postaci kodu pośredniego jak Java lub języki platformy .NET znacząco zwiększa ryzyko wystąpienia negatywnych następstw nieautoryzowanego dostępu.

Zaciemnianie kodu źródłowego w celu przeciwdziałania ewentualnym próbom analizy programu to jedna z technik zarządzania ryzykiem nieautoryzowanego dostępu [4]. Następstwem takiego zdarzenia oprócz wyżej wymienionego zagrożenia własności intelektualnej może być również ułatwienie znalezienia luk bezpieczeństwa lub obniżenie zysku w przypadku aplikacji, która została zmodyfikowana w celu obejścia jej zabezpieczeń chroniących przed kopiowaniem. Zaciemnianie kodu służy w takim przypadku łagodzeniu strat związanych z tym ryzykiem. Do głównych zastosowań obfuskacji możemy zaliczyć:

a) zaciemnianie kodu binarnego programów, np. przez usuwanie wskaźnika rekordu aktywacji (ang.frame pointer ) i informacji o symbolach,

b) zmniejszanie rozmiaru plików wynikowych i poprawianie ich wydajności

c) stosowanie w walce z niechcianymi wiadomościami zwanymi spamami

Zaciemnianie układu strony opiera się na:

a) usuwaniu komentarzy,

b) zamianie identyfikatorów (ang. scrambling identifiers)

c) usuwaniu formatowania (ang. change formating) –

d) zmianie kolejności wyrażeń, komend, pętli – programiści wykazują tendencję do umieszczania razem powiązanego ze sobą kodu,

e) wyłączaniu metod – wyłanianie części kodu i tworzenie na ich podstawie nowych funkcji, procedur lub komponentów,

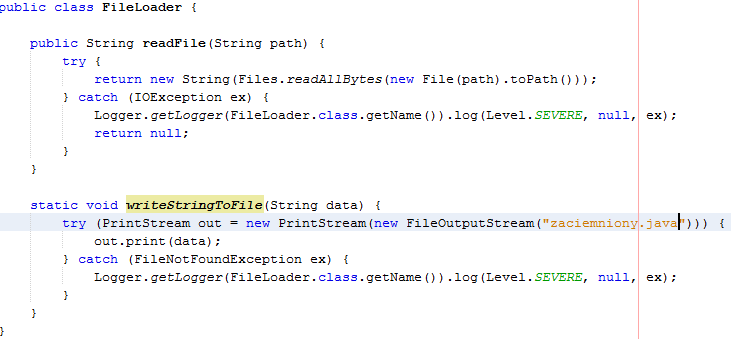
f) rozszerzaniu warunków pętli – dodawanie nieistotnych dla pętli warunków, zmiana warunków logicznych pętli na bardziej rozbudowane.

Trzecią grupą technik zaciemniania kodu jest zaciemnianie danych.

# Rozwiązanie

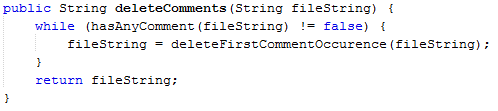
Aplikacja skada się z dwóch klas FileLoader oraz Obfuscator.

Pierwsza FileLoader odpowiada za załadowanie pliku do programu i zapisanie wyniku do pliku.

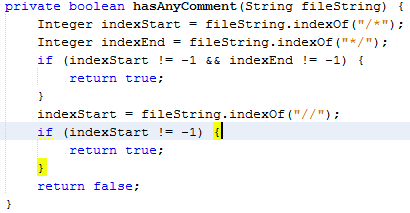


Kolejna czyli Obfuscator to właściwa klasa zaciemniająca. Skłąda się ona z 3 głownych metod:

* deleteComments(String fileString) – czyli metoda usuwająca komentarze,



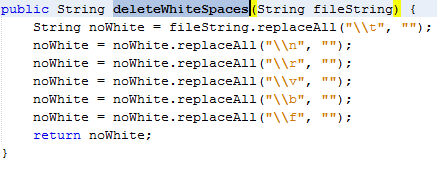
Składa się ona z dwóch pomniejszych metod, pierwsza hasAnyComment(String fileString) sprawdza czy podany tekst zawiera jakikolwiek komentarz:



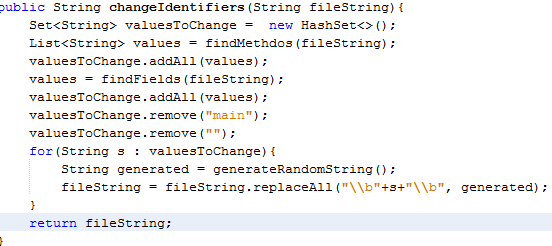
Druga deleteFirstCommentOccurence(String fileString) usuwa pierwszy wysąpienie komentarza, to znaczy że jak podamy do niej jakiś tekst z komentarzem usunie ona pierwszy napotkany komentarz, a reszę pozostawi.

Metoda deleteComments wykonuje w pętli obie te metody aż do momentu gdy metoda hasAnyComments zwróci wartość false czyli wykaże że podany tekst nie zawiera więcej komentarzy.

* Druga metoda to deleteWhiteSpaces(String fileString) – metoda usuwająca puste znaki, co z tym się wiąże usuwająca formatowanie pliku.

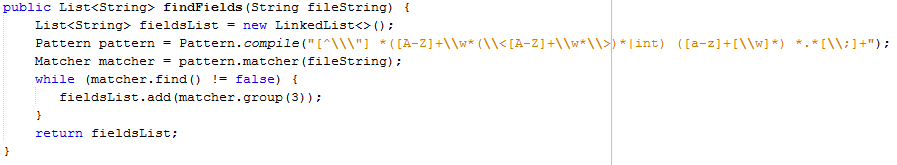


* Trzecia metoda changeIdentifiers(String fileString) - podmienia identyfikatory, podając do niej text zamieni ona zmienne i nazwy metod na losowe ciągi znaków.

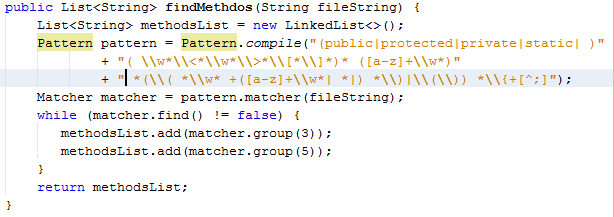


Wykorzystuje ona 4 metody które zapewniaja jej poprawne działanie:

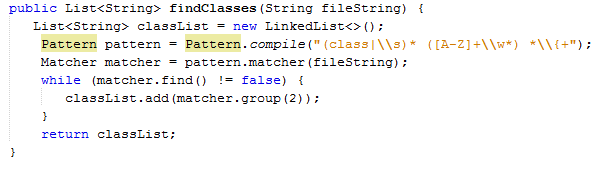
findFields(String fileString) – metoda odnajdująca deklaracie zmiennych w podanym tekscie



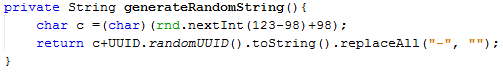
findMethods(String fileString) – metoda odnajdująca nazwy metod i zmiennych przekazywanych przez te metody



findClasses(String fileString) –metoda odnajdująca nazwy klas



generateRandomString()-metoda generująca losowy ciąg znaków na ktory podmieniane są wyszukane przez metody wyżej nazwy.



Metoda changeIdentifiers poczkowo wykoystuje 3 pierwsze klasy do wyszukania nazw wyst changeIdentifiersujh w programie a następnie w pętli dla każdej wyszukanej nazwy generuje unikatowy ciąg znaków i podmienia.