ZESZYTY NAUKOWE POLITECHNIKI ŚLĄSKIEJ 2012
Seria: ORGANIZACJA I ZARZĄDZANIE z. 61 Nr kol. 1873

Ireneusz J. JÓŹWIAK, Marcin MICHALSKI, Marcin PASIECZNY
Politechnika Wrocławska
Wydział Informatyki i Zarządzania

ZACIEMNIANIE KODU JAKO FORMA ZABEZPIECZENIA KODU ŹRÓDŁOWEGO

Streszczenie. W artykule przedstawiono wprowadzenie do techniki zabezpieczenia kodu źródłowego metodą zaciemniania kodu, zwaną obfuskacją. Obfuskacja to technika przekształcania programów, która zachowuje ich semantykę, ale znacząco utrudnia zrozumienie. W artykule zdefiniowano obfuskację, omówiono jej zastosowania oraz wady. Przedstawiono przegląd popularnych programów zwanych obfuskatorami, które pozwalają poddać kod źródłowy procesowi zaciemniania kodu dla popularnych języków programowania, takich jak: języki platformy .NET np. C#, Java, Flash (ActionScript).

CODE OBFUSCATION AS A FORM OF SOURCE CODE SECURITY

Summary. In the paper an introduction to the security source code method called obfuscation has been presented. Obfuscation is a technique of converting programs, which retains its semantics, but it is significantly difficult to understand. This article presents definitions of obfuscation, its uses and drawbacks. Paper provides an overview of popular software called obfuscator, that allows the source code subject to the process of code obfuscation, for popular programming languages such as: .NET platform languages eg. C#, Java, Flash (ActionScript).

1. Wprowadzenie

Szybki rozwój sprzętu komputerowego i ciągły wzrost jego mocy obliczeniowej obecnie wiąże się z coraz większą ilością coraz lepszego oprogramowania do jego obsługi. Większość takiego oprogramowania, jak programy użytkowe czy gry komputerowe, jest coraz częściej rozprowadzana w postaci kodu źródłowego. Rozpowszechniany w ten sposób kod podlega

ochronie praw autorskich. Niestety nie zawsze jest on odpowiednio zabezpieczony pod kątem nieautoryzowanego wykorzystania. Powstało wiele technik i narzędzi zabezpieczających kod źródłowy przed kradzieżą [1]. Jedną z metod zabezpieczenia kodu źródłowego, przedstawioną w niniejszym artykule, jest zaciemnianie kodu, zwane obfuskacją (ang. *obfuscation*).

2. Obfuskacja

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 [1].

Jako pierwszy definicję obfuskacji podał Christian Collberg w „A taxonomy of obfuscating transformations" [2], którą rozwijał w kolejnych publikacjach i jest ona następująca [1]:

$P \xrightarrow{T} P'$ jest transformatą obfuskacyjną, jeżeli:

a) program P przerywa swoją pracę lub kończy ją z błędami, program P' może też przerywać swoje działanie lub skończyć je z błędami,

b) w przeciwnym wypadku P' musi zakończyć swoje działanie i wygenerować identyczne wyniki co P.

Gaël Hachez w pracy „A Comparative Study of Software Protection Tools Suited for E-Commerce with Contributions to Software Watermarking and Smart Cards" [3] zauważa, że definicja przyjęta przez Collberga daje możliwość wprowadzenia w błąd użytkownika oprogramowania [1]. Cytując warunek (a): „...może też przerywać swoje działanie lub skończyć je z błędami", możliwe jest, że oryginalny program zostanie zakończony z błędami, natomiast zmodyfikowany (zabezpieczony) program zakończy się, nie zgłaszając użytkownikowi żadnych błędów. Tego rodzaju niepożądane działanie zostało zmienione przez modyfikację definicji obfuskacji na następującą [1]:

$P \xrightarrow{T} P'$ jest transformatą obfuskacyjną, jeżeli (a) program P przerywa swoją pracę lub kończy ją z błędami, program P' musi też przerywać swoje działanie lub skończyć je z błędami, (b) w przeciwnym wypadku P' musi zakończyć swoje działanie i wygenerować identyczne wyniki co P.

Program P' może mieć inne zachowania w stosunku do programu P, takie jak łączenie z Internetem, wysyłanie poczty elektronicznej czy tworzenie plików, jednak działania te nie będą miały żadnego wpływu na komunikację z użytkownikiem. Od zabezpieczonego programu P' nie jest wymagane, by jego wydajność była równa z wydajnością programu P. Należy

wspomnieć o ograniczeniach, jakim podlega obfuskacja. Trzeba pamiętać, że każdy program P będzie można odtworzyć do P', który swoją budową będzie bardzo zbliżony do programu P. Zawsze będzie to możliwe do wykonania, ale będzie się to wiązać z dużym nakładem finansowym i czasowym. Dlatego zadaniem obfuskacji nie jest zabezpieczenie programu przed jego dekompilacją, lecz w znaczącym stopniu wydłużenie czasu potrzebnego do jego analizy i zrozumienia jego działania [1].

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. W takim przypadku zaciemnianie kodu służy 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; z technik zaciemniania korzysta się również przy tworzeniu MIDletów na platformie Java ME w celu redukcji rozmiaru plików .jar, wygenerowanych w procesie kompilacji,
c) stosowanie w walce z niechcianymi wiadomościami zwanymi spamami; spamerzy zaciemniają treść elektronicznej korespondencji w celu ominięcia filtrów, natomiast użytkownicy poczty elektronicznej, próbujący uniknąć spamu, podają swoje adresy e-poczty w niejawnej postaci, aby utrudnić harvesterom (programom komputerowym służącym do zbierania adresów poczty elektronicznej) ich odczytanie,
d) organizowanie konkursów na najbardziej pomysłowo zaciemnione programy, takie jak: International Obfuscated C Code Contest, Obfuscated Perl Contest, International Obfuscated Ruby Code Contest oraz Obfuscated PostScript Contest.

W dalszej części przedstawiono, w języku C, przykład jednego z najsłynniejszych zaciemnionych kodów źródłowych, wypisujący do standardowego wyjścia dwanaście zwrotek tekstu angielskiej piosenki *The 12 Days of Christmas:*

```
#include <stdio.h>
main(t,_,a)char *a;{return!0<t?t<3?main(-79,-13,a+main(-87,1-_,
main(-86,0,a+1)+a)):1,t<_?main(t+1,_,a):3,main(-94,-27+t,a)&&t==2?_<13?
main(2,_+1,"%s %d %d\n"):9:16:t<0?t<-72?main(_,t,
"@n'+,#'/*{}w+/w#cdnr/+,{}r/*de}+,/*{*+,/w{%+,/w#q#n+,/#{l,+,/n{n+,/+#n+,/#\
;#q#n+,/+k#;*+,/'r :'d*'3,}{w+K w'K:'+}e#';dq#'l \
q#'+d'K#!/+k#;q#'r}eKK#}w'r}eKK{nl]'/#;#q#n'){}#}w'){){nl]'/+#n';d}rw' i;# \
){nl]!/n{n#'; r{#w'r nc{nl]'/#{l,+'K {rw' iK{;[{nl]'/w#q#n'wk nw' \
iwk{KK{nl]!/w{%'l##w#' i; :{nl]'/*{q#'ld;r'}{nlwb!/*de}'c \
;;{nl'-{}rw]'/+,}##*}#nc,',#nw]'/+kd'+e}+;#'rdq#w! nr'/ ') }+}{rl#'{n' ')# \
}'+}##(!!/")
:t<-50?_==*a?putchar(31[a]):main(-65,_,a+1):main((*a=='/')+t,_,a+1)
  :0<t?main(2,2,"%s"):*a=='/'||main(0,main(-61,*a,
"!ek;dc i@bK'(q)-[w]*%n+r3#l,{}:\nuwloca-O;m .vpbks,fxntdCeghiry"),a+1);}
```

Dalej przedstawiono, w języku C, zaciemniony kod źródłowy programu *anonimous.c,* zwycięzcy 1st International Obfuscated C Code Contest 1984 w kategorii Dishonorable mention:

```
int i;main(){for(;i["]<i;++i){--i;}"];read('-'-'-',i+++"hell\o,
world!\n",'/'/'/'));}read(j,i,p){write(j/p+p,i---j,i/i);}
```

a jego pierwotny kod źródłowy programu jest następujący:

```
#include<stdio.h>
int main(void)
{
printf("Hello World!\n");
return 0;
}
```

Przykłady te pokazują, w jakim stopniu można utrudnić analizę krótkich kodów [2, 7].

3. Techniki obfuskacji [1], [4]

Wyróżniamy 3 grupy transformacji obfuskacyjnych [4]. Pierwszą grupą technik zaciemniania kodu jest zbiór transformat odpowiedzialnych za układ i przejrzystość kodu źródłowego. Zaciemnianie układu strony (ang. *layout obfuscation*) polega na zmianie lub usunięciu informacji, które nie mają wpływu na wykonanie programu. Transformaty tego typu są najłatwiejsze do implementacji i zastosowania. Grupa często nazywana jest darmową, ponieważ jej zastosowanie w żadnym stopniu nie wpływa na zwiększenie objętości programu ani na prędkość działania po transformacji. Transformaty układu strony noszą też nazwę „jednokierunkowe" (ang. *one-way*) – raz usunięta informacja lub zmieniona nazwa zmiennej nie jest możliwa do odtworzenia. Swoim działaniem powodują zmniejszenie ilości informacji ułatwiających pracę osobie analizującej.

Zaciemnianie układu strony opiera się na:

a) usuwaniu komentarzy, które opisują działanie programu, oraz informacji dla debuggera, ponieważ często programiści wprowadzają dużą liczbę danych opisujących kod, np. działanie funkcji i procedur czy opis parametrów;
b) zamianie identyfikatorów (ang. *scrambling identifiers*) – wszystkie klasy, metody oraz pola są identyfikowane przez unikalną nazwę. Dobrą praktyką w programowaniu jest nazywanie elementów odpowiednio do ich przeznaczenia lub zastosowania. W ten sposób pomagają one analizującemu poznać rolę, jaką pełni dany element. Zamiana identyfikatorów może skutecznie utrudnić to zadanie;
c) usuwaniu formatowania (ang. *change formating*), to jest usuwaniu różnego rodzaju wcięć typu tabulacje, spacje, białe znaki, czego wynikiem są sklejone wiersze kodu.

Drugą grupą technik zaciemniania kodu jest zmiana kontroli (z ang. *control obfuscation*). Celem tego typu obfuskacji jest zmiana kontroli w programie przy niezmienianiu części wyliczeniowych programu. Wiele technik z tej grupy wywodzi się bezpośrednio z teorii optymalizacji, wykorzystywanej w kompilatorach. Zmiana kontroli opiera się na:

a) zmianie kolejności wyrażeń, komend, pętli – programiści wykazują tendencję do umieszczania razem powiązanego ze sobą kodu,
b) wyłączaniu metod – wyłanianie części kodu i tworzenie na ich podstawie nowych funkcji, procedur lub komponentów,
c) rozszerzaniu warunków pętli – dodawanie warunków nieistotnych dla pętli, zmiana warunków logicznych pętli na bardziej rozbudowane.

Trzecią grupą technik zaciemniania kodu jest zaciemnianie danych (ang. *data obfuscation*), które opiera się na:

a) rozdzielaniu zmiennych,
b) konwersji statycznych danych do procedury,
c) zmianie kodowania,
d) zmianie długości życia zmiennych,
e) łączeniu zmiennych skalarnych,
f) zmianie relacji dziedziczenia,
g) rozłamie lub łączeniu tablic,
h) zmianie porządku instancji zmiennych, metod lub tablic.

Mimo że zaciemnianie kodu jest dosyć dobrą techniką zabezpieczenia, z którą związane są liczne zalety, również i ta forma ochrony nie jest wolna od wad, mniej lub bardziej uciążliwych. Do najważniejszych wad obfuskacji możemy zaliczyć:

a) nie powinno być to jedyne zabezpieczenie kodu źródłowego – żadna ze znanych technik obfuskacji nie daje gwarancji znacznego utrudnienia analizy zaciemnionego programu. Obfuskacja nie zapewnia bezpieczeństwa porównywalnego ze współczesnymi systemami kryptograficznymi, powinna więc być stosowana jako ich dopełnienie;

b) debuggowanie – wykonywanie zaciemnionych programów pod kontrolą debuggera jest niezwykle trudne. Nazwy zmiennych nie mają sensu, a struktura programu może być zmieniona nie do poznania. Zmusza to programistów do utrzymywania przynajmniej dwóch, osobnych kompilacji oraz do porównywania ich zachowania. Ponadto istnienie niezaciemnionej wersji programu zwiększa ryzyko jego rozpowszechnienia m.in. na skutek kradzieży;

c) przenośność – zaciemnianie bardzo często zależy od specyficznych cech kompilatora bądź środowiska wykonania programu, co utrudnia obfuskację, jeśli jedno z nich się zmieni;

d) mechanizm refleksji – umożliwia inspekcję klas i wywoływanie ich metod wyłącznie na podstawie ich nazw. Stosowanie obfuskacji w systemach, których działanie wykorzystuje ten mechanizm, znacznie ogranicza swobodę i wymaga od programistów oznaczenia wszystkich identyfikatorów, których nazwy nie mogą zostać zmienione przez proces obfuskacji.

4. Obfuskatory dla języków platformy .NET

Obfuskator jest to program pozwalający poddać kod źródłowy procesowi obfuskacji. Często obok zaciemniania kodu umożliwia również przeprowadzenie procesu zwanego shrinkowaniem, który polega na usuwaniu niewykorzystywanych fragmentów kodu lub nawet całych klas, kompresji statycznych zmiennych przechowujących długie ciągi znaków lub bajtów itp. Tabela 1 przedstawia najbardziej znane i najczęściej używane programy do zaciemniania kodu dla wybranych, popularnych języków programowania.

Mimo że platforma .NET dostarcza programistom wiele nowych możliwości i ułatwień również w aspekcie bezpieczeństwa kodu źródłowego aplikacji, droga do jego pozyskania do nielegalnego użytku jest dosyć prosta. Można tutaj wspomnieć o programie .NET Reflector [12], który na życzenie użytkownika wygeneruje w pełni funkcjonalny projekt na podstawie dostarczonych plików .dll i .exe. Aby zapobiec takim sytuacjom, można zastosować takie obfuskatory, jak: Dotfuscator, Eazfuscator.NET oraz Macrobject Obfuscator.NET.

Tabela 1

Obfuskatory dla wybranych języków programowania

Język programowania lub platforma programistyczna	Obfuskatory
Platforma .NET	Dotfuscator [5], Eazfuscator.NET [6], Macrobject Obfuscator.NET [7]
Java	ProGuard [8], yGuard [9]
Flash (ActionScript)	SWF Encrypt [10], SecureSWF [11]

Źródło: opracowanie własne.

Dotfuscator firmy PreEmptive, którego darmowa wersja (Community Edition) jest dołączona do środowiska wytwarzania oprogramowania Visual Studio Professional, jest szczególnie polecany przez Microsoft [5]. Wersja darmowa stanowi idealne rozwiązanie dla prostych aplikacji, które wymagają podstawowego poziomu ochrony. Wspiera ono elementy zaciemniania układu strony, takie jak zmiana identyfikatorów. Niestety darmowa wersja ma pewne ograniczenie. Pozwala ona na obfuskacje jednego pliku .dll bez wiedzy, jak wygląda komunikacja z innymi plikami. Dlatego możliwe jest tylko obfuskowanie każdego projektu osobno z włączoną opcją, że obfuskujemy bibliotekę. Dotfuscator w wersji Professional wspiera developerów, którzy wymagają najwyższego poziomu ochrony dla swoich aplikacji. Obsługuje wiele typów aplikacji, takich jak aplikacje SQL, Silverlight, WPF czy Managed C++, co jest istotną przewagą nad wersją darmową, obsługującą tylko aplikacje biurowe. Dodatkowo oprócz zaciemniania układu strony wpiera elementy związane ze zmianą kontroli, oferuje szyfrowanie łańcuchów znakowych i optymalizację pod kątem pamięci.

Eazfuscator.NET jest bezpłatnym obfuskatorem dla platformy .NET, co jest jego atutem [6]. Najistotniejsze jego możliwości to: zmiana nazw identyfikatorów, szyfrowanie i kompresja łańcuchów znakowych, linearyzacja hierarchii klas oraz zmiana kontroli. Dodatkowo umożliwia on automatyczną optymalizację kodu, obsługę błędów oraz integrację ze środowiskiem Visual Studio. Oprócz aplikacji biurowych obsługuje również aplikacje: Silverlight, XNA oraz Windows Mobile 7. Wszystkie jego cechy wskazują na to, że może być ciekawą alternatywą dla płatnej wersji Dotfuscatora.

Macrobject Obfuscator.NET jest komercyjnym programem do zaciemniania kodu dla platformy .NET [7]. Zastosowane w nim mechanizmy uniemożliwiają pokazanie prawidłowej struktury programu po poddaniu go dekompilacji np. za pomocą programu .NET Reflector. Najważniejsze jego możliwości to zaciemnianie układu strony przez zmianę nazw klas, metod, pól i identyfikatorów oraz zmiana kontroli. Istotną jego wadą w stosunku do wyżej przedstawionych obfuskatorów jest brak wsparcia dla .NET Framework w wersji 4.0, co może mieć istotne znaczenie dla developerów wykorzystujących jego nowe mechanizmy w swoich aplikacjach.

5. Obfuskatory dla języka Java

Problem ochrony kodu źródłowego nie jest również obcy developerom stosującym język Java w swoich aplikacjach. Kod źródłowy po zamianie w trakcie kompilacji na b-code (z ang. *byte code*) nie stanowi żadnego problemu dla użytkowników, którzy chcą go nielegalnie wykorzystać. Dlatego możemy również zastosować obfuskatory, które pozwolą chociaż na częściowe rozwiązanie problemu bezpieczeństwa kodu źródłowego. Do najczęściej wykorzystywanych obfuskatorów dla języka Java należą ProGuard i yGuard [8], [9].

ProGuard jest bezpłatnym programem, dystrybuowanym na licencji GNU, do zaciemniania kodu oraz optymalizacji skompilowanych programów w języku Java. Pierwszym procesem, jakim poddaje b-code, jest kompaktowanie, czyli proces analizowania *byte code'u*, czego wynikiem jest usuwanie zbędnego kodu w postaci nieużywanych klas, zmiennych i metod. Oczywiście działanie programu po tej operacji pozostaje niezmienione. Po kompaktowaniu b-code'u nadal znajdują się w nim informacje dla debuggera, które nie są wymagane do poprawnego działania programu. Są to na przykład nazwy plików źródłowych, numery linii czy też nazwy klas, metod i argumentów. Te informacje umożliwiają łatwą dekompilację programu. W drugim procesie ProGuard usuwa te dane, potrzebne debuggerowi, a wszystkie nazwy zmienia na nic nieznaczące ciągi znaków. Działanie programu nie zmienia się, z pewnymi drobnymi wyjątkami. W wypadku wystąpienia błędu w programie, w raporcie (*exception stack trace*) nie uzyskamy poprawnych informacji o numerach linii oraz nazwach, które obfuskator pozamieniał. Jest to cena, jaką płacimy za zaciemnianie kodu. Zastosowanie takich metod w znaczącym stopniu utrudnia inżynierię wsteczną. Jeszcze jedną wadą tego obfuskatora mogą być problemy w programie bądź bibliotece poddawanej zaciemnianiu kodu, jeśli korzystamy w nich z mechanizmu refleksji. Istotną zaletą tego programu jest jego multiplatformowość.

yGuard jest również bezpłatnym programem do obfuskacji kodu bajtowego Javy i jego optymalizacji chroniących kod źródłowy przed niepożądanym dostępem. Wysoko konfigurowalny, automatyczny mechanizm zaciemniania kodu chroni pliki .class przez zmianę nazw pakietów, klas, metod i pól na nic nieznaczące ciągi znaków. Ważnym skutkiem ubocznym tego procesu jest zmniejszenie rozmiaru aplikacji w zależności od wybranego schematu mapowania. Zawarte w nim mechanizmy w znaczącym stopniu utrudniają proces dekompilacji. Istotnymi zaletami tego obfuskatora są poprawne zaciemnianie programów, które zależą od zewnętrznych bibliotek, oraz tworzenie specjalnych plików podczas każdego procesu zaciemniania kodu, które mogą być wykorzystane do wprowadzenia poprawek do już rozwiniętego systemu zaciemniania kodu aplikacji.

6. Obfuskatory dla Flasha

Podobnie jak aplikacje Javy czy .NETa, które działają w wirtualnych maszynach, również aplikacje przygotowane we Flashu, a szczególnie flashowe gry, które ostatnio zdobyły popularność, działają we Flash Playerze. Skoro odtwarzacz potrafi analizować pliki .swf, równie dobrze może to zrobić taki program, jak dekompilator. Użycie takiego dekompilatora automatycznie prowadzi do uzyskania dostępu do kodu źródłowego aplikacji, dlatego również tutaj warto korzystać z obfuskatorów. Najbardziej popularnymi obfuskatorami dla Flasha są SWF Encrypt oraz SecureSWF [10], [11], [13], [14].

SWF Encrypt jest komercyjnym programem przygotowanym przez firmę Amayeta, zaprojektowanym specjalnie dla programu Adobe Flash w wersji 6-10, wykorzystującym zaawansowane techniki zaciemniania kodu (obok sprawdzonej technologii szyfrowania) w celu zapewnienia bezpieczeństwa przed inżynierią wsteczną plików .swf i kradzieżą kodu ActionScript. Dodatkowo oferuje możliwość szyfrowania zasobów wykorzystywanych przez pliki .swf, takich jak obrazy czy grafiki. Wspiera maskowanie plików programów w technologii Adobe AIR. Według opinii producenta SWF Encrypt, zastosowane w nim mechanizmy obfuskacji kodu pozwalają na skuteczne zabezpieczenie nawet jednolinijkowego pliku .swf, w znaczącym stopniu utrudniając jego dekompilację. Wadą tego programu, o której warto wspomnieć, jest w większości przypadków kilka razy większy rozmiar pliku zabezpieczonego względem pliku oryginalnego. Ciekawą funkcją stanowiącą duża zaletę tego obfuskatora jest to, że każda zaciemniona funkcja w porównaniu z jej oryginalną postacią ma dodane na początku i na końcu pewne instrukcje, które mogą spowodować zawieszenie się dekompilatora. Dostępność tego programu na platformy Windows i Mac OSX jest jeszcze jedną jego zaletą.

SecureSWF firmy Kindisoft jest płatnym programem do obfuskacji kodu, wspierającym wszystkie wersje języka ActionScript na każdej platformie. Wspiera wszystkie formaty plików Flash: .swf, .swc oraz .air. Stosuje metody zaciemniania układu strony, polegające na agresywnej zmianie identyfikatorów klas, pól i metod oraz szyfrowaniu łańcuchów znakowych, co stanowi ważną zaletę tego programu. Zapewnia optymalizację kodu przez m.in. zmianę nazw dłuższych identyfikatorów na krótkie, 1–2-znakowe, co powoduje, że zaszyfrowane pliki mają mniejszy rozmiar i mogą się wykonywać szybciej. Program uniemożliwia nieuprawnionym osobom kopiowanie aplikacji Flash i wykorzystywania ich w trybie offline lub osadzanie ich na innych stronach internetowych. Dostępny jest w trzech wersjach dostosowanych do różnych potrzeb, tak aby każdy użytkownik mógł wybrać wersję dla niego najlepszą.

7. Podsumowanie

Obfuskacja kodu nie jest i nie powinna być jedyną formą zabezpieczenia kodu źródłowego. Nawet żadna kombinacja zabezpieczeń nie da nam całkowitej ochrony przed osobami, które chcą wykorzystać nasz kod do własnych celów. Można wymyślać coraz nowsze sposoby zabezpieczania, zaskakiwać atakujących, utrudniać im życie, jednak prędzej czy później i te sposoby zostaną rozszyfrowane. Wszystkie przedstawione programy do zaciemniania kodu mają swoje zalety i wady, ale warto z tych programów korzystać, ponieważ niektóre z nich dodatkowo umożliwiają optymalizację kodu źródłowego, dzięki czemu ma on mniejszy rozmiar i szybciej się wykonuje. Trzeba zaznaczyć, co bardzo ważne, że istnieje co najmniej kilka deobfuscatorów, czyli programów odwracających proces obfuskacji, które potrafią przywrócić kod zaciemniony przez przedstawione programy do całkiem czytelnej formy. Należy mieć to na uwadze, projektując rozwiązania z wykorzystaniem tych technologii.

Bibliografia

1. Yarmolik V.N., Brzozowski M.: Obfuskacja w zabezpieczaniu praw autorskich do projektów sprzętowych. Politechnika Białostocka, Przegląd Telekomunikacyjny, R. LXXI, nr 6, 2008.
2. Collberg C., Thomborson C., Low D.: A Taxonomy of Obfuscating Transformations. Technical Report 148, Department of Computer Science, The University of Auckland, July 1997.
3. Hachez G.: A Comparative Study of Software Protection Tools Suited for E-Commerce with Contributions to Software Watermarking and Smart Cards. PhD thesis, Universite Catholique de Louvain, March 2003.
4. Zaciemnianie kodu, http://pl.wikipedia.org/wiki/Zaciemnianie_kodu.
5. Dotfuscator, http://www.preemptive.com/products/dotfuscator/ [dostęp: 25.05.2011].
6. Eazfuscator.NET, http://www.foss.kharkov.ua/g1/projects/eazfuscator/dotnet/Default.aspx [dostęp: 10.05.2011].
7. Macrobject Obfuscator.NET, http://www.macrobject.com/en/obfuscator/ [dostęp: 10.05.2011].
8. ProGuard, http://proguard.sourceforge.net/ [dostęp: 10.05.2011].
9. yGuard, http://www.yworks.com/en/products_yguard_about.htm [dostęp: 10.05.2011].
10. SWF Encrypt, http://www.amayeta.com/software/swfencrypt/ [dostęp: 10.05.2011].
11. SecureSWF, http://www.kindisoft.com/ [dostęp: 10.05.2011].

12. .NET Reflector, http://reflector.red-gate.com/download.aspx [dostęp: 10.05.2011].
13. Test SWF Encrypt, http://www.konkursy-hack.pl/articles.php?article_id=13 [dostęp: 10.05.2011].
14. Zacharczyk R.: Zabezpieczanie gier w technologii Flash. Software Developer's Journal, No. 6, 2011.

Abstract

Software is more and more frequently distributed in form of source code. Unprotected code is easy to alter and build in others projects. One of the method for protection against attacks on intellectual rights is obfuscation. Obfuscation is a technique of converting programs, which retains its semantics, but it significantly difficult to understand. As a first definition of obfuscation reported Christian Collberg in „A taxonomy of obfuscating transformations". Gaël Hachez in paper „A Comparative Study of Software Protection Tools Suited for E-Commerce with Contributions to Software Watermarking and Smart Cards" noted that the definition adopted by Collberg gives the possibility of misrepresentation and modified the definition of obfuscation. Process of obfuscation is also subject to restrictions. Each program can be converted to a state before obfuscation. Code obfuscation is mainly used in reverse engineering and is one of risk management techniques of unauthorized access. The most important applications of code obfuscation is optimization and protection against spam. There are three transformation groups of obfuscation: layout obfuscation, control obfuscation and data obfuscation. Layout obfuscation is e.q. comments removing, scrambling identifiers and change formatting. Control obfuscation is e.q. reordering of expressions and extension of the loop conditions. Data obfuscation is e.q. separation of variables and changing the encoding. This article describes also popular software called obfuscator, that allow the source code subject to the process of code obfuscation, for popular programming languages such as: .NET platform languages e. g. C#, Java, Flash (ActionScript). In the paper, the list of obfuscator for popular programming languages has been presented.