ISSN 2658-5014 (Print) СИИТ ISSN 2686-7044 (Online)
2023. Т. 5, № 5 (14). С. 68–77 http://siit.ugatu.su
СИСТЕМНАЯ ИНЖЕНЕРИЯ И ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ

УДК 004.4 DOI 10.54708/2658-5014-SIIT-2023-no5-p68

ПОДХОД К ОБЕСПЕЧЕНИЮ БЕЗОПАСНОСТИ ПРОГРАММНОГО КОДА В ВЕБ-ОРИЕНТИРОВАННОЙ СРЕДЕ

Г. О. ОРЛОВ

Аннотация. Цель работы – обеспечить защиту программного обеспечения (ПО) от несанкционированного копирования. Объектом исследования является система защиты от несанкционированного копирования ПО. Предмет исследования – программная реализация защиты кода ПО от взлома и создание несанкционированных копий посредством разработки веб-приложения с защитным модулем. Проведен анализ наиболее актуальных и востребованных методов защиты ПО от несанкционированного копирования. Предложена структура системы защиты на основе разработанного обфускатора кода, был описан алгоритм работы обфускатора кода, после чего на его основе была разработана и протестирована программа, написанная на языке Python с использованием фреймворка Django, реализующая обфускацию кода.

Ключевые слова: защита от пиратства; обфускация; защита ПО; шифрование кода; разработка веб-приложений.

ВВЕДЕНИЕ

В данной статье рассматривается использование веб-приложения для защиты кода программы от несанкционированного копирования, в частности от пиратства [1]. Основной целью является обеспечение защиты ПО от нелегального копирования, а основной задачей – разработка модуля веб-приложения, который обеспечит эту защиту посредством шифрования кода.

Как известно, идеальной защиты от взлома ПО не существует. Имея в своем распоряжении достаточно высокую вычислительную мощность, злоумышленник может подобрать необходимые ключи шифрования на любом этапе алгоритма при использовании, к примеру, реверс-инжиниринга.

Поэтому с ростом технических возможностей ЭВМ совершенствовать и усложнять для злоумышленника алгоритмы шифрования данных необходимо, а это значит, что и новые методы защиты ПО от нелегального копирования должны появляться регулярно, в том числе использующие веб-сервисы, и разработчику необходимо будет их внедрять, в случае если он заинтересован в минимизации потерь прибыли вследствие пиратства.

АНАЛИЗ МЕТОДОВ

1. Обфускация кода

Несмотря на то, что существует заблуждение о том, что преобразование кода на языке высокого уровня в машинный код при исполнении программы может быть достаточной помехой для злоумышленников, крайне рекомендуется использовать обфускацию (запутывание, усложнение) кода. Предполагается, что в код добавляется множество по факту лишних команд, процедур, переменных и др., которые никак не влияют на конечный результат и служат исключительно инструментами для того, чтобы запутать злоумышленника, не дать возможность понять, где конкретно в коде происходит привязка программы, например, к цифровому ключу [2–4].

Применение обфускации (запутывания) кода является крайне востребованным методом защиты из-за такой фундаментальной проблемы, как тот факт, что написанная человеком программа может быть человеком и понята, проанализирована, разобрана.

2. Псевдокод, полиморфные технологии

Данный метод защиты является более сложным в реализации, чем обфускация кода. Использование псевдокода не меняет структуру программы, вместо этого изменяется язык. Всем необходимым командам из текущего языка в соответствие создаются новые команды, на так называемом псевдоязыке. Их суть от этого не изменяется, однако если злоумышленник не знает логики псевдоязыка, код окажется для него неразборчивым.

Псевдокоманды воспроизводятся через специальный эмулятор при запуске приложения для того, чтобы код воспроизводился корректно. Если после применения данного метода надежности защиты будет недостаточно, то можно скомбинировать его с полиморфными технологиями, которые предполагают, что каждый раз при запуске программы будут генерироваться новые случайные команды на псевдоязыке, соответственно каждый раз параметры у эмулятора будут различаться [5].

Чтобы определить, какой из двух этих методов является наиболее пригодным для реализации защитного модуля, был произведен сравнительный анализ с учетом преимуществ и недостатков каждого из них, представленный в таблице.

Таблица

Сравнительный анализ методов

Решаемая проблема	Метод	Достоинства	Недостатки
Скрытие API-интерфейса в коде ПО	Обфускация	API сложнее обнаружить в запутанном коде; метод прост в исполнении; вариативен; достаточно надежен	Может снизить производительность ПО
	Псевдокод	API сложнее распознать из-за псевдокоманд; надежен; не замедляет ПО лишними командами	Трудно реализовать; эмулятор для псевдокода может создавать лишнюю нагрузку
Снижение уязвимости к реверс-инжинирингу	Обфускация	Запутывает код, затрудняя понимание логики работы	Может снизить производительность ПО
	Псевдокод	При незнании логики псевдоязыка понять логику ПО невозможно	Логика ПО просчитывается, в случае понимания того, что означают псевдокоманды

Учитывая вышеописанные достоинства и недостатки, можно прийти к выводу, что для относительного простого и эффективного решения проблемы нелегального копирования ПО можно прибегнуть к такому решению, как использование обфускации кода. Таким образом будет получена достаточно надежная, но в то же время не слишком сложная в реализации и ресурсоемкая система защиты.

Места, в которых верифицируется ключ, будут скрыты от злоумышленника ложными/видоизмененными строками.

Предлагаемый алгоритм обфускации

Далее представлена блок-схема разработанного для защитного модуля алгоритма обфускации кода (рисунок 1).

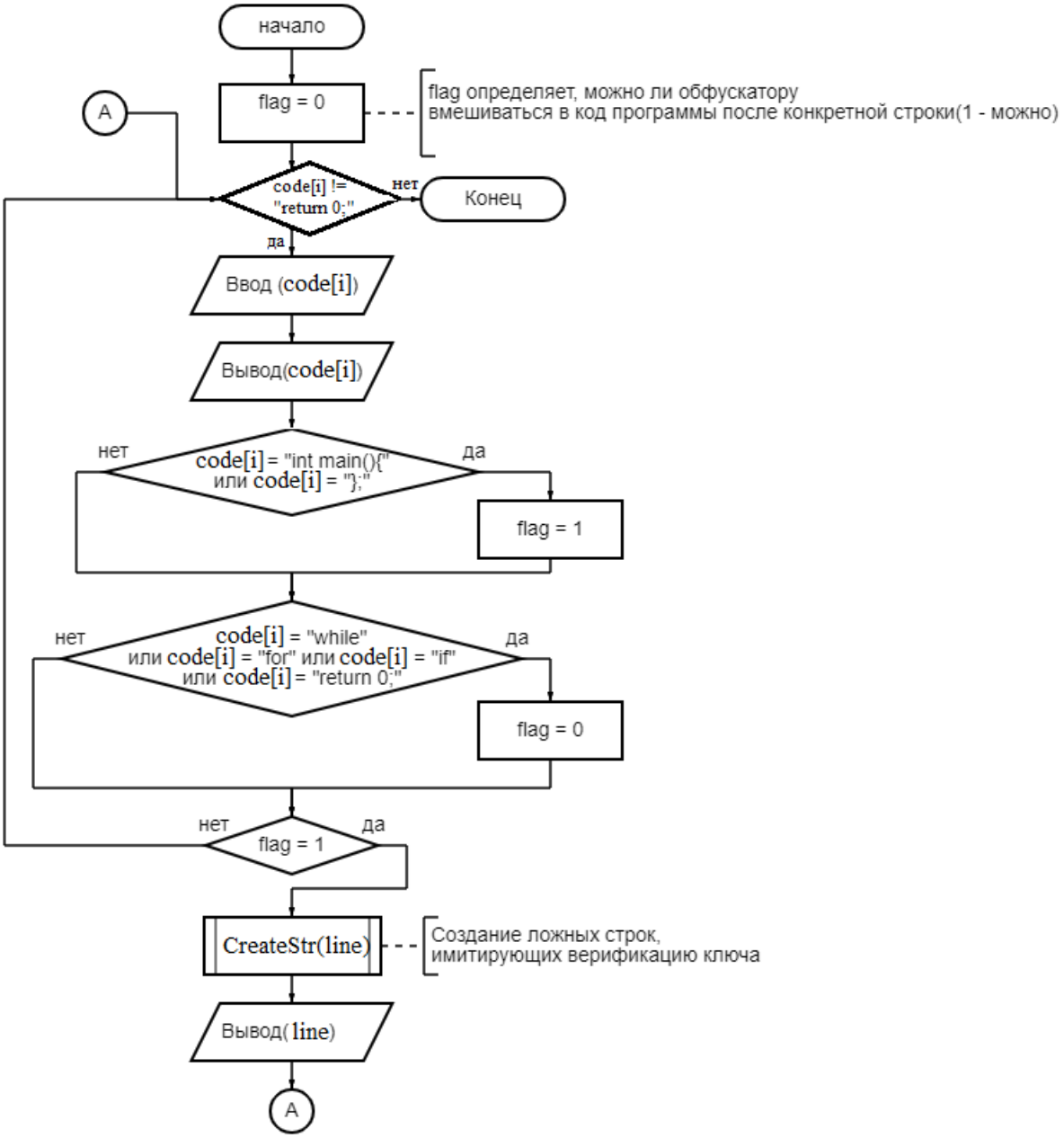

Рис. 1 Блок-схема алгоритма обфускации кода.

Алгоритм составлялся таким образом, чтобы стремиться к «обфускации неразличимости» («Indistinguishability Obfuscation» [6]). Как утверждается в статье «Способ построения неразличимого программного кода с использованием ключа» [7], данный класс обфускаторов можно применить к программного коду, имея в основе определенный ключ, причем сделать это на высокоуровневом языке программирования, что существенно облегчает внедрение данной защиты в оборот.

Code[i] обозначает переменные, в которые считываются изначальные строки кода. Затем они передаются в шаблон с добавлением ложных строк, которые имитируют код модуля верификации ключа. То есть весь код разделяется построчно, и при его анализе рассматривается

каждая отдельная строка, происходит детализация информации. Принцип детализации данных для обеспечения безопасности уже использовался в статье СИИТ «Детализация пространственной информации для обеспечения защищенности баз данных в распределенных информационных системах» [8]. Это мешает злоумышленнику обойти защиту, изменив определенные части исходного кода.

Ветвления в алгоритме для различных значений code[i] нужны, чтобы не позволить обфускатору вставить лишние строки в тех местах, где это нарушит логику программы или внутри тела цикла. Таким образом, сложность компрометации снижается в пользу производительности ПО, что рекомендовано в трудах других ученых, занимающихся защитой программного кода [9–11].

Подфункции CreateStr1 и CreateStr2 генерируют лишние строки в программе, имея заготовленный набор названий переменных, по смыслу похожих на те, которые использовались бы при верификации ключа. Каждый раз названия для них выбираются случайным образом, поскольку, как показывают исследования других авторов, использование случайных параметров при обфускации значительно повышает эффективность защиты [12–14].

В результате на выходе имеется код с тем же функционалом, но запутанный таким образом, что злоумышленнику сложно будет найти необходимые ему области программного кода. Эффективность защитного алгоритма проверена с помощью методики, разработанной С. С. Лебедевым [15].

Используемое программное обеспечение

Защитное приложение было разработано с использованием фреймворка Django на языке программирования Python. В качестве среды разработки использовалась PyCharm.

Реализация защиты в виде веб-приложения

За основу для выстраивания подхода к созданию архитектуры приложения был взяты идеи из работ [16–18].

Сперва, при помощи фреймворка Django, был запущен локальный сервер (рисунок 2).

```
PS C:\Users\Gleb\PycharmProjects\Obfuscator\obfuscator> python manage.py runserver
Watching for file changes with StatReloader
Performing system checks...

System check identified no issues (0 silenced).
June 09, 2022 - 23:42:45
Django version 4.0.3, using settings 'obfuscator.settings'
Starting development server at http://127.0.0.1:8000/
Quit the server with CTRL-BREAK.
```

Рис. 2 Запуск сервера с помощью Django.

Затем создается форма для передачи исходного кода программы на сервер для его дальнейшей обфускации.

Для этого необходимо в файле `forms.py` проекта импортировать `forms`, и создать соответствующий класс:

```
from django import forms
class CodeForm(forms.Form):
    code_field = forms.CharField(widget=forms.Textarea)
```

Далее необходимо обратиться к файлу `views.py`. В нем создаются необходимые функции для обработки защищаемого кода. Сперва подключаются `render` для передачи обработанного кода в шаблон, импортируется ранее созданная форма и `randint` для генерации случайных названий переменных ложных строк:

```
from django.shortcuts import render
from .forms import CodeForm
from random import randint
```

Далее создаются функции, которые составляют случайные ложные строки:

```
def CreateVarName():
    case = randint(1, 3)
    if (case == 1):
        trick = "user_key" + str(randint(1, 100))
    if (case == 2):
        trick = "valid_key" + str(randint(1, 100))
    if (case == 3):
        trick = "license_key" + str(randint(1, 100))
    return trick
def CreateStr1(var):
    trick = "int " + var + "=" + str(randint(1000, 5000)) + ";"
    return trick
def CreateStr2(var1, var2):
    trick = "if(" + var1 + " == " + var2 + "){bool valid = 1;}"
    return trick
```

Затем код, полученный из формы, передается в список из строк. Список изменяется в соответствии с описанным ранее алгоритмом обфускации с помощью функций CreateStr:

```
def index(request):
    submitbutton = request.POST.get("submit")
    code_raw = ''
    code = ''

    form = CodeForm(request.POST or None)
    if form.is_valid():
        code_raw = form.data.get("code_field")
        code = code_raw.split("\r\n")
        flag = 0
        i = 0
        while code[i] != "return 0;":
            if code[i] == "int main(){" or code[i] == "};":
                flag = 1

            if code[i] == "while" or code[i] == "for" or
                          code[i] == "if" or code[i] == "return 0;":
                flag = 0

            if flag == 1:
                line = CreateVarName()
                line2 = CreateVarName()
                code.insert(i + 1, CreateStr1(line))
                code.insert(i + 2, CreateStr1(line2))
                code.insert(i + 3, CreateStr2(line, line2))
                i += 3
            i += 1

    context = {'form': form, 'code': code,
                                      'submitbutton': submitbutton}

    return render(request, 'test.html', context)
```

После обработки измененный код передается в шаблон test.html и пользователю выдается обфусцированный код:

```
<head>
<meta charset="UTF-8">
<title>C++ Obfuscator</title>
</head>Enter some C++ code to obfuscate:</head>
<body></body>
<form action="" method="POST">
{% csrf_token %}
{{ form.as_p }}
<input type="Submit" name="submit" value="Submit"/>
</form>

{% if submitbutton == "Submit" %}
 <h2>Obfuscated code:</h2>
 </h3>{%for i in code%}
  {{i|linebreaks}}
{%endfor%}</h3>
{% endif %}
```

Апробация защитного модуля

Модуль обфускации при запуске сайта предлагает вставить защищаемый код в форму. Далее его можно будет отправить на сервер для дальнейшего шифрования (рисунок 3).

Enter some C++ code to obfuscate:

Code field:

```
#include <iostream>
#include <string>
#include <fstream>
#include <cstdlib>
#include <time.h>

using namespace std;

int main(){
    srand(time(0));
    string str;
    string license;
    ifstream key("keys.txt");
    ofstream key2("keys_new.txt");
    bool flag = 0;
    cout << "Enter your license key, please:" << endl;
    cin >> license;

while
(getline(key, str)){
        if(str == license)
        {
            flag = 1;
            str = "NULL";
        }

    key2 << str << endl;
};
    key.close();
```

Submit

Рис. 3 Обфускатор кода.

При нажатии на кнопку отправки в шаблон передается обфусцированный код, и разработчик может получить его запутанную версию (рисунок 4).

В результате, в коде появляется множество ложных строк, похожих на те, которые реализуют верификацию ключа, не нарушающих функционал программы, но запутывающих при этом злоумышленника.

Снижение быстродействия ПО после обфускации сведено к минимуму, так как алгоритм не допускает большого количества повторений ложных команд в циклах.

Obfuscated code:

```
#include <iostream>
#include <string>
#include <fstream>
#include <cstdlib>
#include <time.h>
using namespace std;
int main(){
int license_key96=1191;
int license_key89=3771;
if(license_key96 == license_key89){bool valid = 1;}
srand(time(0));
int user_key6=3570;
int user_key5=3460;
if(user_key6 == user_key5){bool valid = 1;}
string str;
int user_key24=4027;
int license_key64=3318;
if(user_key24 == license_key64){bool valid = 1;}
string license;
int license_key6=4477;
int user_key62=4572;
if(license_key6 == user_key62){bool valid = 1;}
ifstream key("keys.txt");
int license_key72=4939;
int license_key57=3134;
if(license_key72 == license_key57){bool valid = 1;}
ofstream key2("keys_new.txt");
int license_key31=3601;
int valid_key61=1388;
if(license_key31 == valid_key61){bool valid = 1;}
bool flag = 0;
int valid_key77=1955;
int license_key75=1511;
if(valid_key77 == license_key75){bool valid = 1;}
cout << "Enter your license key, please:" << endl;
```

Рис. 4 Исходный код после обфускации.

Заключение

В данной статье представлен основной модуль программы для защиты ПО от пиратства, а именно – веб-обфускатор программного кода.

В результате тестирования защитного модуля не было выявлено ошибок или существенного замедления работы в защищаемой программе, а код в результате обфускации оказывается существенно запутанным.

Был разработан обфускатор, который видоизменяет программный код таким образом, что для злоумышленника найти необходимую для него функцию или участок кода крайне проблематично из-за множества ложных строк, имитирующих область кода, в которой происходит верификация ключа.

Поскольку вышеописанные изменения в коде серьезно усложняют взлом программы для злоумышленника, защита от нелегального копирования ПО обеспечена, следовательно, основную цель данной работы можно считать достигнутой, а основные задачи – выполненными.

Недостатки обычных обфускаторов, устраненные в предлагаемой системе защиты:

1) Существенное замедление работы программы. Поскольку предлагаемый обфускатор не использует ресурсоемкие команды, а также не располагает ложные строки внутри длинных циклов, данная проблема решена.

2) Простота деобфускации. Программный код в результате работы предлагаемой системы защиты оказывается заполненным строками, похожими на те, что нужны злоумышленнику, следовательно, деобфусцировать такой код сложнее. В аналогичном обфускаторе от StarForce [19] легко отличить строки, добавленные обфускатором от исходного кода.

В ходе разработки данного веб-приложения были протестированы возможности фреймворка Django, используемого для создания веб-приложений на языке Python, а именно – было продемонстрировано создание форм, шаблонов и функций обработки данных из форм с последующей передачей в шаблон результатов, что помогло эффективно разрабатывать обфускатор кода именно в форме веб-приложения.

Благодарности

Автор выражает признательность научному руководителю д-ру техн. наук Г. Р. Воробьевой за помощь в составлении алгоритма защитного ПО, постановку задачи и полезное обсуждение вопросов.

Список литературы / References

1. Что такое компьютерное пиратство? [Электронный ресурс]. URL: https://www.cloudav.ru/mediacenter/tips/software-piracy/ (Дата обращения: 01.08.2023). [[What is Software Piracy? [Electronic resource]. URL: https://www.cloudav.ru/mediacenter/tips/software-piracy/ (Date of access: 01/08/2023). (In Russian).]]

2. Современные технологии защиты ПО от нелегального копирования» [Электронный ресурс]. URL: http://lib.itsec.ru/articles2/Oborandteh/sovremennie-tehnologii-zashiti-po-ot-nelegalnogo-kopirovaniya-chto-vibrat-razrabotchiky (Дата обращения: 04.08.2023). [[Modern Technologies for Protecting Software from Illegal Copying" [Electronic resource]. URL: http://lib.itsec.ru/articles2/Oborandteh/sovremennie-tehnologii-zashiti-po-ot-nelegalnogo-kopirovaniya-chto-vibrat-razrabotchiky (Date of access: 08/04/2023). (In Russian).]]

3. Обфускация программ: [Электронный ресурс]. URL: https://habr.com/ru/ёpost/255871/ (Дата обращения: 07.08.2023). [[Obfuscation of Programs: [Electronic resource]. URL: https://habr.com/ru/epost/255871/ (Date of access: 08/07/2023). (In Russian).]]

4. Обфускация (программное обеспечение): [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Обфускация_(программное_обеспечение) (Дата обращения: 13.08.2023). [[Obfuscation (software): [Electronic resource]. URL: https://ru.wikipedia.org/wiki/Obfuscation_(software_ware) (Date of access: 08/13/2023). (In Russian).]]

5. Псевдокод и полиморфные технологии: [Электронный ресурс]. URL: https://www.aktiv-company.ru/press-center/publication/2010-08-13.html (Дата обращения: 07.09.2023). [[Pseudocode and Polymorphic Technologies: [Electronic resource]. URL: https://www.aktiv-company.ru/press-center/publication/2010-08-13.html (Access date: 09/07/2023). (In Russian).]]

6. Garg S., Gentry C., Halevi S., Raykova M., Sahai A. and Waters B. Candidate indistinguishability obfuscation and functional encryption for all circuits. 2013// IEEE 54th Annual Symposium on Foundations of Computer Science, Berkeley, CA, USA, 2013. Pp. 40–49, doi: 10.1109/FOCS.2013.13.

7. Способ построения неразличимого программного кода с использованием ключа: [Электронный ресурс]. URL: https://cyberleninka.ru/article/n/sposob-postroeniya-nerazlichimogo-programmnogo-koda-s-ispolzovaniem-klyucha/viewer (Дата обращения: 10.08.2023). [[A method for Constructing an Indistinguishable Program Code Using a Key: [Electronic resource]. URL: https://cyberleninka.ru/article/n/sposob-postroeniya-nerazlichimogo-programmnogo-koda-s-ispolzovaniem-klyucha/viewer (Date of access: 08/10/2023). (In Russian).]]

8. Закирова Э. Ф., Павлов С. В., Трубин В. Д., Христодуло О. И. Детализация пространственной информации для обеспечения защищенности баз данных в распределенных информационных системах // Системная инженерия и информационные технологии. 2022. Т. 4. № 1 (8). С. 20–26. [[Zakirova E. F., Pavlov S. V., Trubin V. D., Christodoulo O. I. "Detailing spatial information to ensure the security of databases in distributed information systems" // System Engineering and Information Technologies. 2022. Vol. 4, No. 1 (8), pp. 20–26. (In Russian).]]

9. Аранов В. Ю. Метод и средства защиты исполняемого программного кода от динамического и статического анализа: дис. … канд. техн. наук: 05.13.19: защищена 23.12.2014. Санкт-Петербургский государственный политехнический университет, СПб., 2014. [[Aranov V. Yu. Method and Means of Protecting Executable Program Code from Dynamic and Static Analysis: dis.

...cand. tech. sciences: 13/05/19: protected 12/23/2014. St. Petersburg State Polytechnic University, St. Petersburg, 2014. (In Russian).]]

10. Щелкунов Д. А. Разработка методик защиты программ от анализа и модификации на основе запутывания кода и данных: дис. … канд. техн. наук: 05.13.19: защищена 18.02.2009 МГТУ им. Н. Э. Баумана. М., 2009. [[Shchelkunov D. A. Development of Methods for Protecting Programs from Analysis and Modification Based on Obfuscation of Code and Data: dis. ... cand. tech. sciences: 13/05/19: protected 02/18/2009 MSTU im. N. E. Bauman, Moscow, 2009. (In Russian).]]

11. Краснопевцев А. А. Защита от несанкционированного копирования приложений, компилируемых в промежуточное представление: дис. … канд. техн. наук: 05.13.19: защищена 24.06.2011. Национальный исследовательский ядерный университет «МИФИ». М., 2011. [[Krasnopevtsev A. A. Protection against Unauthorized Copying of Applications Compiled into an Intermediate Representation: dis. ... cand. tech. sciences: 13/05/19: protected 06/24/2011. National Research Nuclear University "MEPhI", Moscow, 2011. (In Russian).]]

12. Буинцев Д. Н. Метод защиты программных средств на основе запутывающих преобразований: дис. … канд. техн. наук: 05.13.19: защищена 09.03.2006. Томский государственный университет систем управления и радиоэлектроники. Томск, 2006. [[Buintsev D. N. Method of Software Protection based on Entangling Transformations: dis. ...cand. tech. sciences: 13/05/19: protected 03/09/2006. Tomsk State University of Control Systems and Radioelectronics, Tomsk, 2006. (In Russian).]]

13. Цивин С. В. Методы защиты программных средств вне доверенной вычислительной среды: дис. … канд. техн. наук: 05.13.19: защищена 25.11.2000. Пензенский государственный университет. Пенза, 2000. [[Tsivin S.V. Methods for Protecting Software outside a Trusted Computing Environment: dis. ...cand. tech. sciences: 13/05/19: protected 11/25/2000. Penza State University, Penza, 2000. (In Russian).]]

14. Машевский Ю. В. Исследование существующих и разработка новых программных средств защиты информации от динамического и статического анализа: дис. … канд. техн. наук: 05.13.11: защищена 21.05.2004. Московский энергетический институт. М., 2004. [[Mashevsky Yu. V. Research of Existing and Development of New Software Tools for Protecting Information from Dynamic and Static Analysis: dis. ...cand. tech. sciences: 13/05/11: protected 05/21/2004. Moscow Energy Institute, Moscow, 2004. (In Russian).]]

15. Лебедев С. С. Разработка методов и средств комплексной оценки качества систем защиты программного обеспечения: дис. … канд. техн. наук: 05.02.23: защищена 05.12.2007. Московский авиационный институт. М., 2007. [[Lebedev S. S. Development of Methods and Means for Comprehensive Assessment of the Quality of Software Protection Systems: dis. ... cand. tech. sciences: 02/05/23: protected 12/05/2007. Moscow Aviation Institute, Moscow, 2007. (In Russian).]]

16. Воробьев А. В., Пилипенко В. А., Еникеев Т. А., Воробьева Г. Р. Геоинформационная система для анализа динамики экстремальных геомагнитных возмущений по данным наблюдений наземных станций // Компьютерная оптика. 2020. Т. 44. С. 782–790. [[Vorobyov A. V., Pilipenko V. A., Enikeev T. A., Vorobyova G. R. “Geoinformation system for analyzing the dynamics of extreme geomagnetic disturbances based on observational data from ground stations” // Computer Optics, 44, pp. 782–790, 2020. (In Russian).]]

17. Воробьева Г. Р. Методологические основы обработки неоднородной пространственно-временной информации в системах поддержки принятия решений на основе технологий больших данных (на примере геомагнитных данных) // Системная инженерия и информационные технологии. 2023. Т. 5. № 3 (12). С. 3–26. [[Vorobyova G. R. “Methodological foundations for processing heterogeneous spatio-temporal information in decision support systems based on big data technologies (using the example of geomagnetic data)” // System Engineering and Information Technologies. 2023. Vol. 5, No. 3 (12), pp. 3–26. (In Russian).]]

18. Воробьев А. В. Методологические основы обработки пространственной информации для поддержки принятия решений на основе агрегированных цифровых двойников (на примере высокоширотных геомагнитных данных) // Системная инженерия и информационные технологии. 2023. Т. 5. № 4 (13). С. 3–27. [[Vorobyov A. V. “Methodological foundations for processing spatial information to support decision-making based on aggregated digital twins (using the example of high-latitude geomagnetic data)” // System Engineering and Information Technologies. 2023. Vol. 5, No. 4 (13), pp. 3-27. (In Russian).]]

19. StarForce C++ Obfuscator: [Электронный ресурс]. URL: https://www.star-force.ru/products/starforce-obfuscator/ (Дата обращения: 17.08.2023).

Поступила в редакцию 19 сентября 2023 г.

МЕТАДАННЫЕ / METADATA

Title: An approach to ensuring software code security in a web-based environment.

Abstract: Purpose of the work: to ensure software protection from unauthorized copying. The object of the study is a system for protecting against unauthorized copying of software. The subject of the research is the software implementation of software code protection from hacking and the creation of unauthorized copies through the development of a web application with a security module. An analysis of the most relevant and popular methods of protecting software from unauthorized copying has been carried out. The structure of the protection system based on the developed code obfuscator was proposed, the algorithm of the code obfuscator was described, after which, based on it, a program written in Python using the Django framework was developed and tested, implementing code obfuscation.

Key words: protection against piracy; obfuscation; software protection; code encryption; web application development.

Язык статьи / Language: русский / Russian.

Об авторе / About the author:

ОРЛОВ Глеб Олегович
ФГБОУ ВО «Уфимский университет науки и технологий», Россия.
Аспирант ин-та информатики, математики и робототехники. Иссл. в обл. информационной безопасности, защиты программного кода
E-mail: orlovgleb99@mail.ru
ORCID: https://orcid.org/0009-0003-5123-3859

ORLOV Gleb Olegovich
Ufa University of Science and Technologies, Russia.
Postgraduate student, Institute of Informatics, Mathematics, and Robotics. Research in the field of information security, software code protection.
E-mail: orlovgleb99@mail.ru
ORCID: https://orcid.org/0009-0003-5123-3859