

Министерство образования Республики Беларусь

Учреждение образования
БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Информационные сети. Основы безопасности

ОТЧЕТ
к лабораторной работе №7
на тему

ЗАЩИТА ПО ОТ НЕСАНКЦИОНИРОВАННОГО ИСПОЛЬЗОВАНИЯ

Выполнил

П.Н. Носкович

Проверил

А.В. Герчик

Минск 2025

СОДЕРЖАНИЕ

Введение.....	3
1 Краткие теоретические сведения.....	4
2 Результаты выполнения лабораторной работы.....	5
Заключение	6
Приложение А (обязательное) Листинг кода.....	7

ВВЕДЕНИЕ

Задача данной работы состоит в демонстрации обфускации исходного кода программы с целью повышения безопасности и защиты от анализа злоумышленниками. Для этого были предприняты следующие шаги:

- 1 Замена имен переменных и функций на случайные, трудночитаемые строки.

- 2 Удаление лишних пробелов и переносов строк в коде, что затрудняет его восприятие, но не изменяет функциональность.

Обфускация не должна изменять работу программы, а лишь усложнять процесс анализа исходного кода. Программа должна продолжать выполнять те же функции, что и до обфускации, но ее код становится сложным для понимания.

1 КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

Обфускация или запутывание кода – это приведение исходного текста или исполняемого кода программы к виду, сохраняющему её функциональность, но затрудняющему анализ, понимание алгоритмов работы и модификацию при декомпиляции.

Обфускация производится в следующих целях:

1 Затруднение декомпиляции/отладки и изучения программ с целью обнаружения функциональности.

2 Затруднение декомпиляции программ с целью предотвращения обратной разработки или обхода *DRM* и систем проверки лицензий.

3 Оптимизация программы с целью уменьшения размера работающего кода и (если используется не компилируемый язык) ускорения работы.

4 Демонстрация неочевидных возможностей языка и квалификации программиста (если производится вручную, а не инструментальными средствами).

«Запутывание» кода может осуществляться на уровне алгоритма, исходного текста и/или ассемблерного текста. Для создания запутанного ассемблерного текста могут использоваться специализированные компиляторы, использующие неочевидные или недокументированные возможности среды исполнения программы. Существуют также специальные программы, производящие обфускацию, называемые обфускаторами (англ. *obfuscator*).

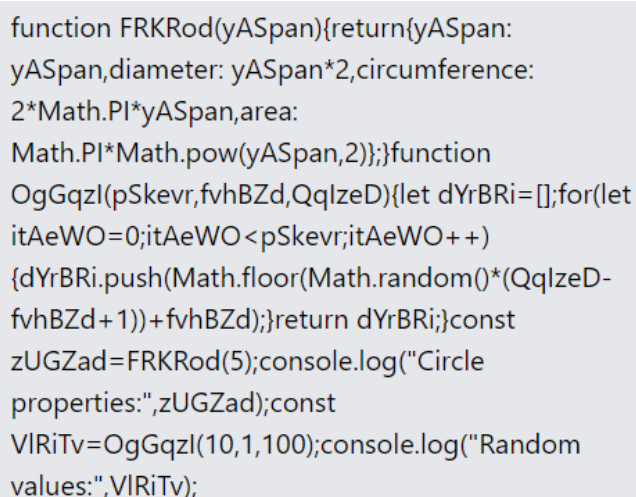
2 РЕЗУЛЬТАТЫ ВЫПОЛНЕНИЯ ЛАБОРАТОРНОЙ РАБОТЫ

В результате разработки программы был реализован метод обфускации программного кода приложения. Исходный код представлен на рисунке 2.1, а обфусцированный код – на рисунке 2.2.



```
main.js
1- function calculateCircleProperties(radius) {
2-   return {
3-     radius: radius,
4-     diameter: radius * 2,
5-     circumference: 2 * Math.PI * radius,
6-     area: Math.PI * Math.pow(radius, 2)
7-   };
8- }
9-
10- function generateRandomNumbers(count, min, max) {
11-   let numbers = [];
12-   for (let i = 0; i < count; i++) {
13-     numbers.push(Math.floor(Math.random() * (max - min + 1)) + min);
14-   }
15-   return numbers;
16- }
17-
18- const circle = calculateCircleProperties(5);
19- console.log("Circle properties:", circle);
20-
21- const randomNumbers = generateRandomNumbers(10, 1, 100);
22- console.log("Random values:", randomNumbers);
23-
```

Рисунок 2.1 – Исходный код



```
function FRKRod(yASpan){return{yASpan:
yASpan,diameter: yASpan*2,circumference:
2*Math.PI*yASpan,area:
Math.PI*Math.pow(yASpan,2)};}function
OgGqzl(pSkevr,fvhBZd,QqlzeD){let dYrBRi=[];for(let
itAeWO=0;itAeWO<pSkevr;itAeWO++)
{dYrBRi.push(Math.floor(Math.random()*(QqlzeD-
fvhBZd+1))+fvhBZd);}return dYrBRi;}const
zUGZad=FRKRod(5);console.log("Circle
properties:",zUGZad);const
VIRiTv=OgGqzl(10,1,100);console.log("Random
values:",VIRiTv);
```

Рисунок 2.2 – Код после обфускации

После обфускации код программы продолжает выполнять ту же задачу, что и до обфускации, но теперь его сложнее анализировать. Все переменные и функции имеют случайные имена, и вся логика программы скрыта за этими идентификаторами.

ЗАКЛЮЧЕНИЕ

В ходе работы была успешно выполнена обфускация программы, написанной на языке *JavaScript*. Результаты показали, что даже базовые методы обфускации, такие как замена имен переменных и функций, могут значительно усложнить анализ кода и повысить безопасность приложения.

Обфускация доказала свою эффективность как метод защиты программного обеспечения от несанкционированного доступа. Этот метод затрудняет задачу для злоумышленников, которые пытаются изучить исходный код с целью его эксплуатации. Важно подчеркнуть, что обфускация является важным элементом комплексной стратегии защиты программного обеспечения, позволяющим защитить код от обратного инжиниринга и атак.

Проведенная работа демонстрирует значимость обфускации для повышения безопасности программного обеспечения в условиях современных угроз и подтверждает, что даже простые методы защиты, такие как изменение имен переменных и функций, могут существенно затруднить несанкционированное использование программы.

ПРИЛОЖЕНИЕ А

(обязательное)

Листинг кода

```
import React, { useState } from "react";

function App() {
  // Исходный код
  const [sourceCode, setSourceCode] = useState(`
function calculateCircleProperties(radius) {
  return {
    radius: radius,
    diameter: radius * 2,
    circumference: 2 * Math.PI * radius,
    area: Math.PI * Math.pow(radius, 2)
  };
}

function generateRandomNumbers(count, min, max) {
  let numbers = [];
  for (let i = 0; i < count; i++) {
    numbers.push(Math.floor(Math.random() * (max - min + 1)) + min);
  }
  return numbers;
}

const circle = calculateCircleProperties(5);
console.log("Circle properties:", circle);

const randomNumbers = generateRandomNumbers(10, 1, 100);
console.log("Random values:", randomNumbers);
`);

  const [obfuscatedCode, setObfuscatedCode] = useState("");

  // Обфускация кода
  const obfuscateCode = (code) => {
    function generateRandomString(length) {
      const chars = "ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz";
      let result = "";
      for (let i = 0; i < length; i++) {
        result += chars.charAt(Math.floor(Math.random() * chars.length));
      }
      return result;
    }

    let varPattern = /[a-zA-Z_][a-zA-Z0-9_]*\s*=\s*[^\s;]+/g;
    let funcPattern = /function\s+([a-zA-Z_][a-zA-Z0-9_]*\s*\(/g;
    let paramPattern = /function\s+[a-zA-Z_][a-zA-Z0-9_]*\s*\((([^\s]*)*)\)/g;

    let replacements = {};

    code = code.replace(varPattern, (match, p1) => {
      let newName = generateRandomString(6);
      replacements[p1] = newName;
      return match.replace(p1, newName);
    });

    code = code.replace(funcPattern, (match, p1) => {
      let newName = generateRandomString(6);
      replacements[p1] = newName;
      return match.replace(p1, newName);
    });
  };
}
```

```

});

code = code.replace(paramPattern, (match, p1) => {
  let params = p1.split(",").map((param) => param.trim()).filter((param)
=> param);
  let newParams = params.map((param) => {
    let newName = generateRandomString(6);
    replacements[param] = newName;
    return newName;
  });
  return match.replace(p1, newParams.join(","));
});

for (let oldName in replacements) {
  let newName = replacements[oldName];
  let regExp = new RegExp(`\\b${oldName}\\b`, "g");
  code = code.replace(regExp, newName);
}

let lines = code.split("\n");
lines = lines.filter((line) => line.trim() !== "");

let obfuscatedCode = lines.join(" ");

obfuscatedCode = obfuscatedCode
  .replace(/\s*([+\\-*/=<>!&|(){}[\],;.])\s*/g, "$1")
  .replace(/\s{2,}/g, " ")
  .replace(/; /g, ";")
  .replace(/,/g, ",")
  .replace(/\\ /g, "\\ ")
  .replace(/\\{/g, "{")
  .replace(/\\}/g, "}");

return obfuscatedCode;
};

// Обработчик обфускации
const handleObfuscate = () => {
  const result = obfuscateCode(sourceCode);
  setObfuscatedCode(result);
};

return (
  <div className="min-h-screen bg-gray-100 flex flex-col items-center p-6">
    <h1 className="text-2xl font-bold mb-4">Лабораторная №7. Обфускация
кода</h1>
    <p className="text-gray-700 mb-4 max-w-2xl text-center">
      Обфускация кода – это процесс преобразования исходного кода в менее
читаемый вид, чтобы усложнить его анализ.
      Здесь вы можете вставить JavaScript-код, обфусцировать его и увидеть
результат.
    </p>
    <div className="flex flex-col md:flex-row gap-4 w-full max-w-4xl">
      <div className="flex-1">
        <h2 className="text-lg font-semibold mb-2">Исходный код:</h2>
        <textarea
          className="w-full h-60 p-3 border border-gray-300 rounded-md"
          value={sourceCode}
          onChange={(e) => setSourceCode(e.target.value)}
        ></textarea>
      </div>
      <div className="flex-1">
        <h2 className="text-lg font-semibold mb-2">Обфусцированный
код:</h2>

```



```

        <textarea
            className="w-full h-60 p-3 border border-gray-300 rounded-md bg-
gray-200"
            value={obfuscatedCode}
            readOnly
        ></textarea>
    </div>
</div>
<button
    onClick={handleObfuscate}
    className="mt-4 px-6 py-2 bg-blue-600 text-white rounded-md hover:bg-
blue-700"
    >
        Обфусцировать код
    </button>
    <div className="mt-8 max-w-3xl text-gray-700">
        <h3 className="text-lg font-semibold mb-2">Как работает
обфускация:</h3>
        <p className="mb-2">
            Обфускация заменяет имена функций, переменных и параметров на
случайные строки. Это делает код менее читаемым,
            но он продолжает выполнять ту же функцию. Например:
        </p>
        <ul className="list-disc list-inside mb-4">
            <li>Переменные <code>radius</code> могут стать
<code>AbCdEf</code>.</li>
            <li>Функции <code>calculateCircleProperties</code> могут стать
<code>XyZ123</code>.</li>
        </ul>
        <p>
            Такой процесс полезен для защиты интеллектуальной собственности, но
его следует использовать с осторожностью.
        </p>
    </div>
</div>
    );
}

export default App;

```