Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет   
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина: Информационные сети. Основы безопасности

ОТЧЁТ

к лабораторной работе №7

на тему

**ЗАЩИТА ПО ОТ НЕСАНКЦИОНИРОВАННОГО ИСПОЛЬЗОВАНИЯ**

Выполнил: студент гр.253505 Сенько Н. С.

Проверил: ассистент кафедры информатики Герчик А.В.

Минск 2025

**СОДЕРЖАНИЕ**

[1 Цель работы 3](#_Toc190108682)

[2 Ход работы 4](#_Toc190108683)

[Заключение 6](#_Toc190108684)

# 1 ЦЕЛЬ РАБОТЫ

Целью исследования стало изучение и практическое применение методов обфускации JavaScript-кода на примере реализации анимации «матричного дождя» (эффект 404). Основные задачи включали: преобразование читаемого кода в запутанную версию с сохранением функциональности, замену осмысленных идентификаторов на бессмысленные комбинации символов, внедрение шестнадцатеричных значений, разбиение строковых констант, а также усложнение логики для противодействия реверс-инжинирингу. Работа направлена на демонстрацию того, как обфускация затрудняет анализ алгоритмов, защищает интеллектуальную собственность и повышает устойчивость кода к модификациям.

# 2 ХОД РАБОТЫ

Исходный код, реализующий анимацию падения символов на элементе canvas, был проанализирован для выявления ключевых компонентов: настройки холста, генерации символов, управления кадрами через setInterval. На первом этапе проведена минификация: удалены пробелы, комментарии, сокращены имена переменных (например, context → ctx). Далее выполнена обфускация: идентификаторы заменены на комбинации вида \_0x5a7b54, строковые литералы разбиты через split('') и закодированы в шестнадцатеричные значения (например, getElementById → ['getElementById']), а функции переписаны с использованием анонимных выражений и замыканий. Для усложнения логики добавлены фиктивные операции, бессмысленные проверки условий и избыточные вызовы методов. Ключевые части кода, такие как генерация символов и обновление позиций, были замаскированы через динамическое обращение к свойствам объекта (например, ctx['fillText'] вместо ctx.fillText). После преобразований проведено тестирование: проверена корректность отрисовки анимации, частота кадров, обработка изменений размеров окна. Результаты подтвердили, что функциональность сохранена, но анализ кода стал значительно сложнее из-за отсутствия понятных имен переменных и нагромождения ложной логики.

# ЗАКЛЮЧЕНИЕ

Практическое применение методов обфускации к JavaScript-коду продемонстрировало их эффективность для защиты алгоритмов от анализа. Замена идентификаторов, маскировка строк, внедрение избыточных операций и динамическое выполнение методов позволили сохранить работоспособность анимации «матричного дождя», но сделали код практически нечитаемым без дополнительной декомпиляции. Работа подтвердила, что обфускация является важным инструментом в веб-разработке для защиты клиентской логики, усложнения кражи или модификации кода, особенно в проектах с требованием к безопасности. Однако подчеркивается необходимость баланса между уровнем запутывания и производительностью: избыточные вычисления могут замедлить выполнение скрипта, что требует оптимизации на этапе тестирования.

<html><head><meta http-equiv="Content-Type" content="text/html; charset=windows-1252">

<style>

\*{

margin: 0;

padding: 0;

}

html {

width: 100%;

height: 100%;

background: radial-gradient(circle, #fff 0%, #aaa 100%) no-repeat;

overflow-x: hidden;

overflow-y: hidden;

}

body {

text-align: center;

display: table;

background: black;

width: 100%;

height: 100%;

overflow-x: hidden;

overflow-y: hidden;

}

canvas {display:block;}

#website {

position: absolute;

bottom: 10px;

left: 10px;

color : #0F0;

z-index : 1;

box-sizing: border-box;

vertical-align: middle;

}

span {

font-family: monospace;

font-size: 1.5em;

}

span:after {

content:" Page Not Found";

opacity: 0;

animation: cursor 1s infinite;

}

@keyframes cursor {

0% {

opacity: 0;

}

40% {

opacity: 0;

}

50% {

opacity: 1;

}

90% {

opacity: 1;

}

100% {

opacity: 0;

}

}

</style>

</head>

<body>

<canvas id="c" height="1025" width="1752"></canvas>

<span id="website">METANIT.COM</span>

<script>

var \_0x1e70c3 = document['getElementById']('c'), \_0xb2df2 = \_0x1e70c3['getContext']('2d');

\_0x1e70c3['height'] = window['innerHeight'], \_0x1e70c3['width'] = window['innerWidth'];

var \_0x18ac86 = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ123456789@#$%^&\*()\*&^%';

\_0x18ac86 = \_0x18ac86['split']('');

var \_0x5f3eb4 = -0x23 + -0x70a + 0x737, \_0x2da960 = \_0x1e70c3['width'] / \_0x5f3eb4, \_0x36a7ee = [];

for (var \_0x4de37e = 0x930 + 0x946 + -0x1276; \_0x4de37e < \_0x2da960; \_0x4de37e++)

\_0x36a7ee[\_0x4de37e] = 0x67 \* 0x43 + -0x1096 + -0xa5e;

function \_0x43b105() {

\_0xb2df2['fillStyle'] = 'rgba(0,\x200,\x200,\x200.04)', \_0xb2df2['fillRect'](0x49 \* -0x58 + 0x611 \* -0x1 + 0x1f29, -0x89 \* -0x1 + -0x12b + 0x6 \* 0x1b, \_0x1e70c3['width'], \_0x1e70c3['height']), \_0xb2df2['fillStyle'] = '#0F0', \_0xb2df2['font'] = \_0x5f3eb4 + 'px\x20arial';

for (var \_0x55a7db = -0x8b \* -0x3b + 0x2f \* -0x63 + -0xddc; \_0x55a7db < \_0x36a7ee['length']; \_0x55a7db++) {

var \_0x4ee8f2 = \_0x18ac86[Math['floor'](Math['random']() \* \_0x18ac86['length'])];

\_0xb2df2['fillText'](\_0x4ee8f2, \_0x55a7db \* \_0x5f3eb4, \_0x36a7ee[\_0x55a7db] \* \_0x5f3eb4);

if (\_0x36a7ee[\_0x55a7db] \* \_0x5f3eb4 > \_0x1e70c3['height'] && Math['random']() > -0x6f \* -0x1d + -0x1 \* 0x1337 + -0x32 \* -0x22 + 0.975)

\_0x36a7ee[\_0x55a7db] = -0x1 \* 0x1137 + -0x680 \* 0x3 + 0xf1 \* 0x27;

\_0x36a7ee[\_0x55a7db]++;

}

}

setInterval(\_0x43b105, -0x14fd + 0x157 \* 0xe + 0x25e \* 0x1);

</script>

</body></html>