**Федеральное государственное бюджетное образовательное учреждение высшего образования   
«Саратовский государственный технический университет имени Ю. А. Гагарина»**

**Кафедра прикладных информационных технологий**

**Отчёт по практической работе «Инструментальные программные средства обфускации исходных текстов программ».**

Выполнил студент б1-ИФСТ-31,

Песчанов Арсений Андреевич

Проверил

Старший преподаватель

кафедры «ПИТ»

Пиминов Дмитрий Алексеевич

Саратов, 2023

# Инструментальные программные средства обфускации исходных текстов программ

**1 часть: Анализ существующих инструментальных средств обфускации исходного текста программ.**

**Obfuscator.io:**

* Obfuscator.io - это бесплатный онлайн-сервис обфускации кода, который поддерживает множество языков программирования, включая Java, JavaScript, Python, C/C++, PHP, Ruby и Go. Obfuscator.io использует комбинацию методов обфускации, включая переименование переменных, шифрование кода, добавление шума и запутывания вызовов функций (Рисунок 1).

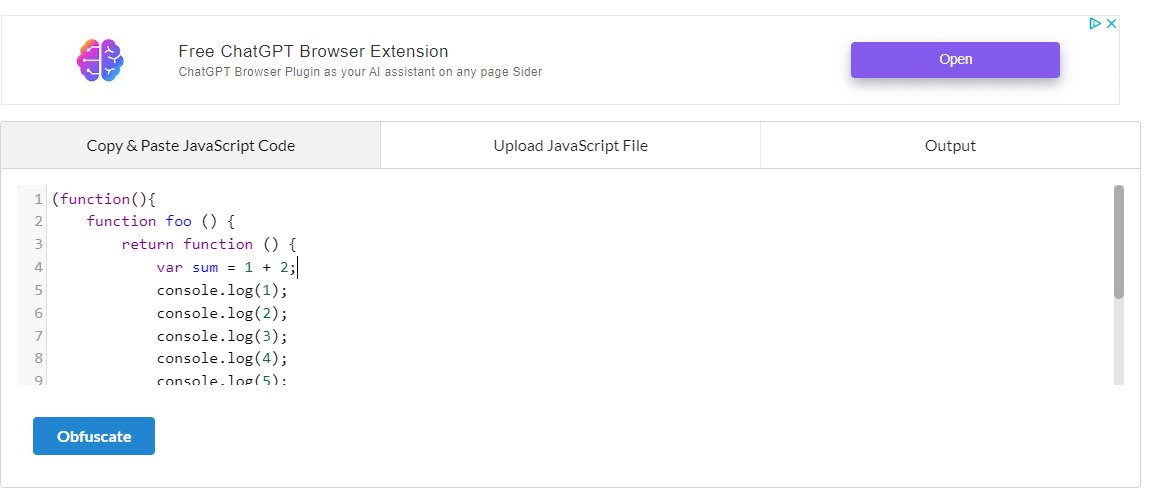


Рисунок 1. Код до обфускаци

* Ограничения использования: бесплатный план Obfuscator.io имеет следующие ограничения, максимальный размер файла - 10 МБ, количество файлов - 10 файлов в день, количество методов - 100 методов в день (Рисунок 2).

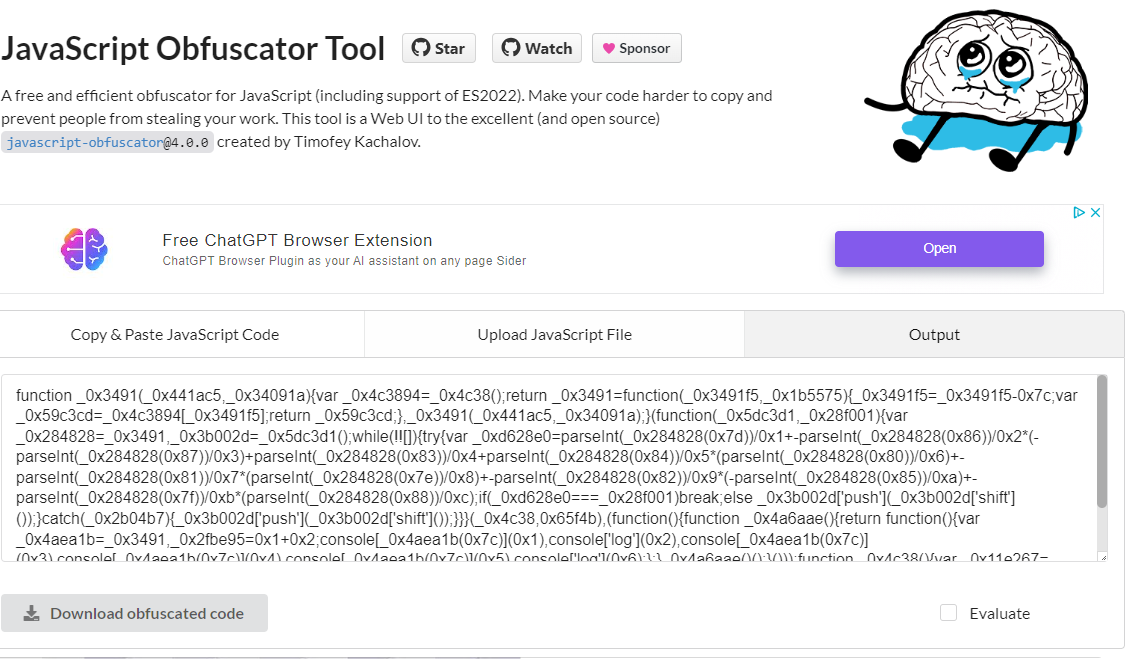


Рисунок 2. Код после обфускации

* Obfuscator.io использует комбинацию следующих методов обфускации: переименование переменных: переменные переименовываются в случайные имена, шифрование кода: код шифруется с использованием алгоритма AES - Advanced Encryption Standard — это алгоритм симметричного шифрования, то есть ключи шифрования и дешифрования совпадают. Поскольку AES является блочным шифром, данные делятся на 128-битные блоки перед их шифрованием 256-битным ключом, добавление шума: в код добавляются случайные символы, чтобы затруднить его понимание, запутывание вызовов функций: порядок вызовов функций изменяется, чтобы затруднить их анализ.
* Дополнительные возможности: Obfuscator.io предлагает следующие дополнительные возможности - поддержка нескольких языков программирования: Obfuscator.io поддерживает более 20 языков программирования, простота использования: Obfuscator.io имеет простой интерфейс, который позволяет легко обфусцировать код, надежность: Obfuscator.io использует надежные методы обфускации, которые делают код более сложным для понимания (Рисунок 3).

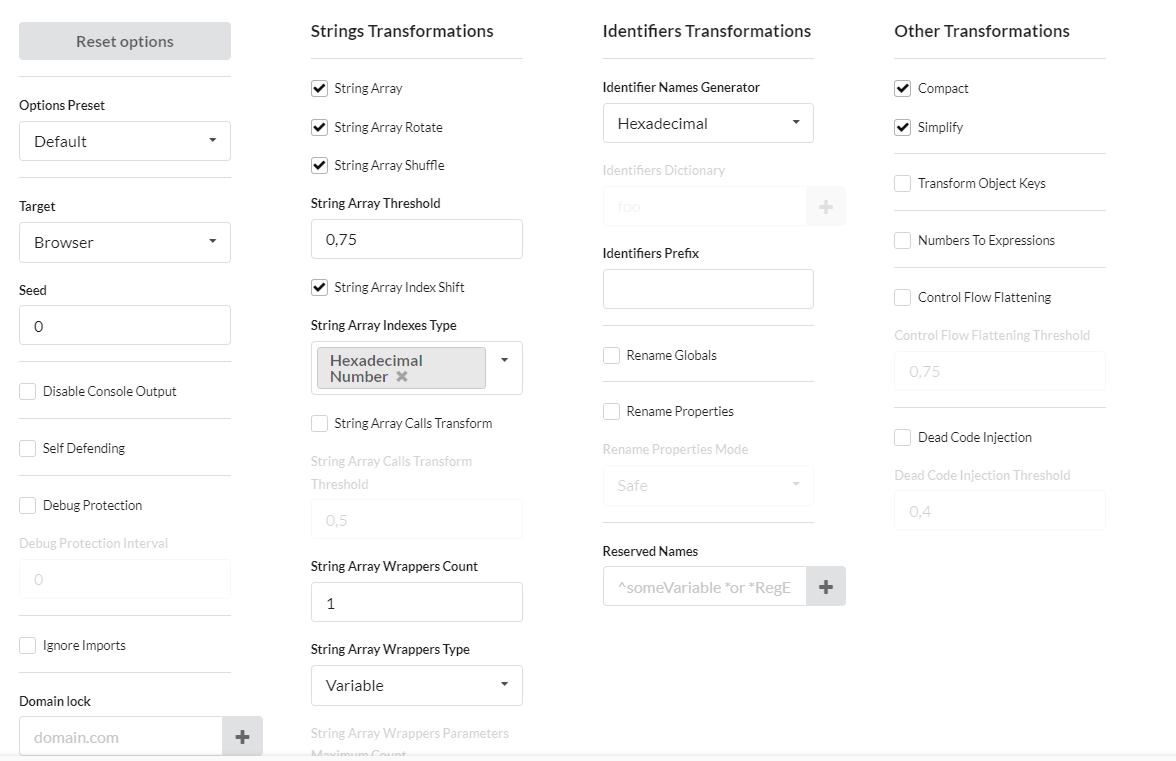


Рисунок 3. Настройки обускатора Obfuscator.io

**Java Obfuscator:**

* Java Obfuscator - это бесплатный инструмент командной строки для обфускации Java-кода. Java Obfuscator использует комбинацию методов обфускации, включая переименование переменных, шифрование кода, добавление шума и запутывания вызовов функций (Рисунок 4).

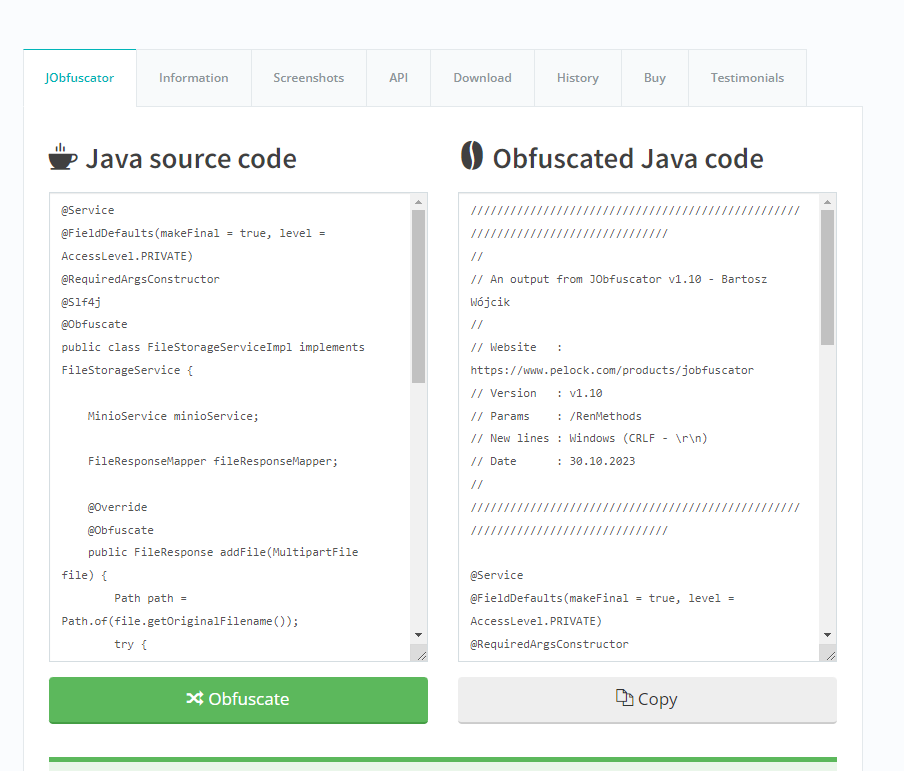


Рисунок 4. Интерфейс Java Obfuscator

* **Ограничения использования** - Java Obfuscator не имеет ограничений по размеру файла или количеству файлов. Однако он имеет ограничения по количеству методов, которые могут быть обфусцированы в одном сеансе. Ограничение составляет 1000 методов для бесплатной версии и 10000 методов для платной версии.
* **Условия использования** - Java Obfuscator доступен бесплатно для личного использования. Для коммерческого использования требуется лицензия.
* Алгоритмы обфускации, лежащие в основе инструментального средства Java Obfuscator использует комбинацию следующих методов обфускации: переименование переменных - переменные переименовываются в случайные имена, шифрование кода - код шифруется с использованием алгоритма AES, добавление шума - в код добавляются случайные символы, чтобы затруднить его понимание, запутывание вызовов функций - порядок вызовов функций изменяется, чтобы затруднить их анализ (Рисунок 5).



Рисунок 5. Настройки обускатора Java Obfuscator

**2 часть. Практическая реализация элементарного обфускационного инструментального средства.**

**Выбранный язык – Lua**:

* Lua - это скриптовый язык программирования. Интерпретатор языка является свободно распространяемым, с открытым исходным кодом на языке программирования Си (Рисунок 6).
* Динамическая типизация – в Lua типизация динамическая, то есть переменная может принимать значение любого типа. Это означает, что не нужно явно указывать тип переменной при её объявлении.
* Процесс интерпретации языка программирования - это выполнение исходного кода программы без его предварительной компиляции в машинный код. Вместо этого интерпретатор анализирует исходный код и выполняет его пошагово.
* Процесс интерпретации можно разделить на следующие этапы: синтаксический анализ - на этом этапе интерпретатор анализирует исходный код и строит его синтаксическое дерево. Синтаксическое дерево представляет собой иерархическую структуру, которая отражает структуру исходного кода. Семантический анализ - на этом этапе интерпретатор проверяет исходный код на наличие ошибок и выполняет семантический анализ. Семантический анализ определяет типы значений, переменных и выражений в исходном коде. Вычисление - на этом этапе интерпретатор выполняет вычисления, необходимые для выполнения программы. Это включает в себя выполнение операций, вычисление значений выражений и вызов функций.

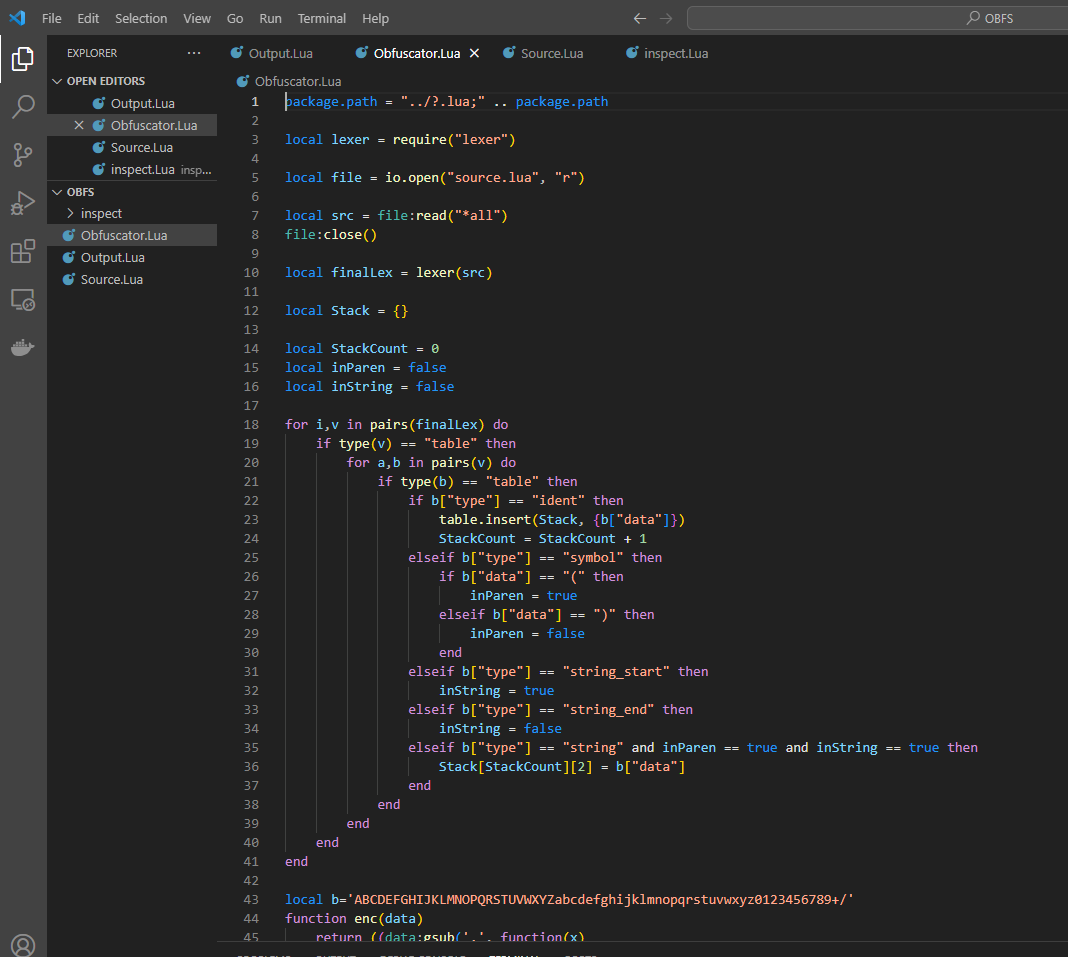


Рисунок 6. Язык Lua

**Обфускация**:

* Код выполняет обфускацию путем замены идентификаторов и строк в исходном коде на их бинарное представление:
* Бинарное представление идентификаторов и строк: идентификаторы и строки заменяются своим бинарным представлением с использованием функции enc и dec. Это делает их сложными для прочтения и понимания (Рисунок 7).

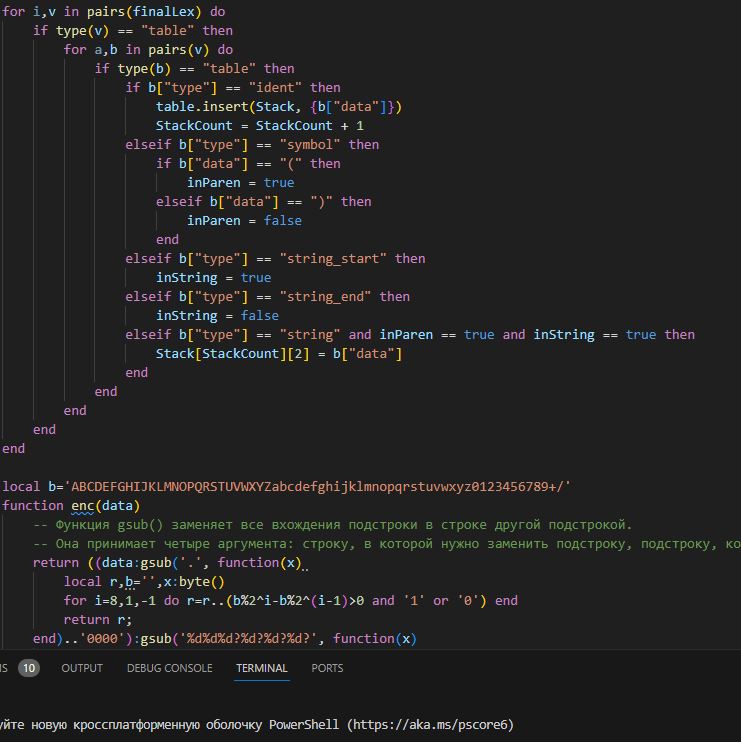


Рисунок 7. Бинарное представление идентификаторов и строк

* Множественные уровни вложенности: скрипт создает множество уровней вложенности, что делает код более трудным для понимания. В частности, создаются вложенные функции и использование таблиц.
* Избыточные переменные: использование избыточных переменных, таких как Stack, StackCount, inParen, inString, count, делает код более запутанным (Рисунок 8).

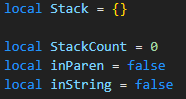


Рисунок 8. Избыточные переменные

* Генерация кода: генерация кода Lua, который приводит к изменению оригинального скрипта и созданию нового файла Output.lua. Это усложняет понимание логики программы.
* Использование setmetatable: используется setmetatable для создания метатаблиц Inspector\_mt (Рисунок 9).

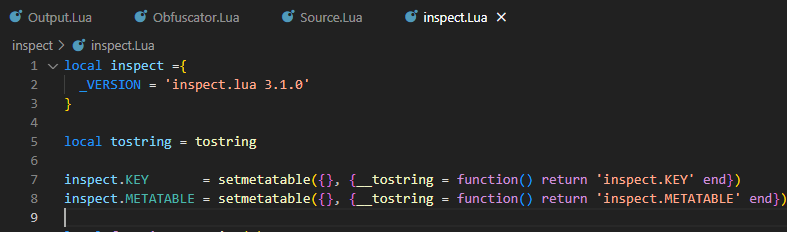


Рисунок 9. Setmetatable

* В коде есть использование строковых замен, например gsub для обработки специальных символов (Рисунок 10).

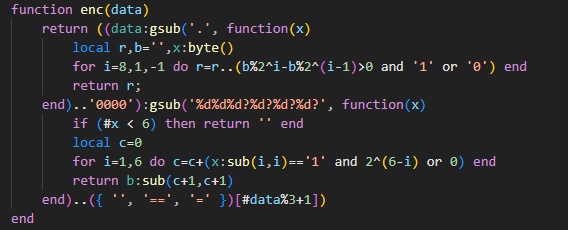


Рисунок 10. Setmetatable

* Код до и после обфускации (Рисунок 11).

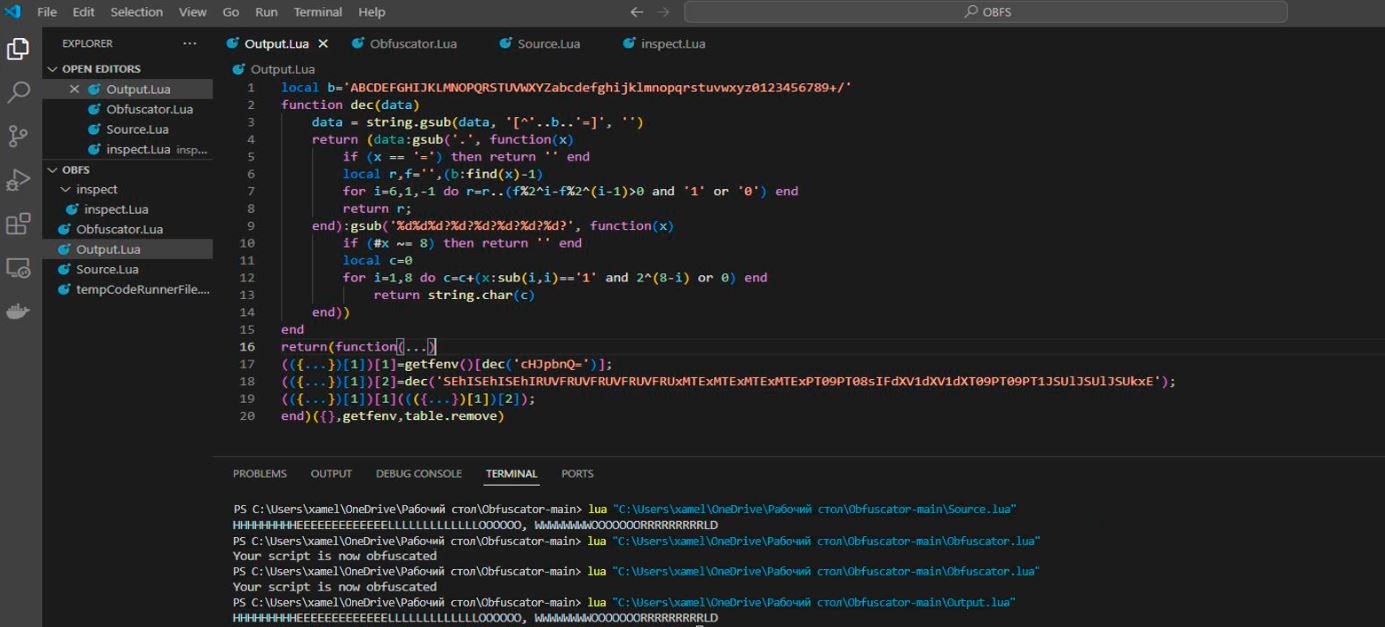


Рисунок 10. Выполнение обфускации

**Вывод:** в ходе выполнения практической работы по теме «Инструментальные программные средства обфускации исходных текстов программ» я познакомился с различными методами обфускации кода на примере языка Lua. Конкретно я использовал замену идентификаторов и строк в исходном коде на их бинарное представление, уровни вложенности - скрипт создает множество уровней вложенности, избыточные переменные такие как Stack, StackCount, inParen, inString,