УДК 004.932.2+004.932.72'1

Д.С. Сипаков, Н.Е. Губенко
Донецкий национальный технический университет, г. Донецк
кафедра компьютерных систем мониторинга

СРАВНИТЕЛЬНЫЙ АНАЛИЗ МЕТОДОВ ОБФУСКАЦИИ ПРОГРАММНОГО КОДА

Аннотация

Сипаков Д.С., Губенко Н.Е. Сравнительный анализ обфускации программного кода. *Выполнен анализ методов обфускации программного кода языков высокого уровня. Определены преимущества и недостатки каждого метода, произведен сравнительный анализ методов.*

Ключевые слова: *обфускация, обфускатор, защита кода, запутывание.*

Постановка проблемы. Защите программного кода от декомпиляции в современном мире уделяется большое внимание, так как исходные тексты программ являются коммерческой тайной. Кроме того, знание внутреннего устройства систем может помочь злоумышленникам атаковать пользователей. Для защиты исходных текстов программ используется обфускация. Обфускация - приведение исходного текста или исполняемого кода программы к виду, сохраняющему ее функциональность, но затрудняющему анализ, понимание алгоритмов работы и модификацию при декомпиляции. При реализации обфускации необходимо выполнить следующие цели:

- затруднение декомпиляции (отладки) и изучения программ;
- предотвращение обратной разработки;
- предотвращение копирования или незаконного использования исходного кода программы;
- защита авторских прав на программный код.

В ДонНТУ уже выполнялись работы по обфускации ([1, 2]) из данных работ была собрана информация о некоторых методах и технологиях обфускации.

Цель статьи – провести анализ методов обфускации, выделить недостатки и преимущества каждого метода.

Постановка задачи исследования. Необходимо произвести сравнительный анализ методов и технологий обфускации. Выделить достоинства и недостатки. Определить требования к защите исходного программного кода от повторного использования злоумышленниками.

Решение задач и результаты исследований. Существует три основных направления обфускации: лексическая обфускация, обфускация данных, обфускация потока управления. Так как обфускация данных и потока управления не всегда разрешается на разных платформах, то были исследованы основные методы и технологии лексической обфускации.

1 Удаление пробелов и переименование. В процессе обфусцирования из кода программы удаляются все комментарии, отступы и пробелы, делая данный код трудночитаемым. Также, если есть возможность, изменяются имена переменных или классов, или методов. Таким образом, после выполнения данного метода обфусцирования, полученный код теряет структурированность, а разобраться в нем становится в разы тяжелее. Результаты представлены на рис.1.

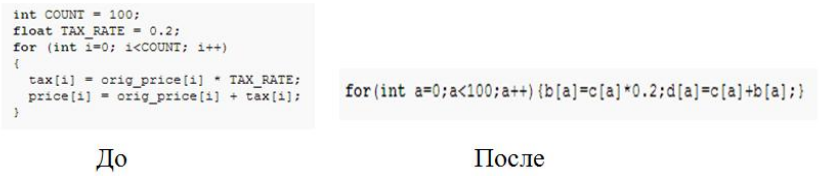

Рисунок 1 – Результаты удаления пробелов, отступов и переименования

Данный метод имеет следующие преимущества:

- потеря структурности;
- изменение типа переменных;
- запутывание логики.

Несмотря на преимущества, данная технология не лишена следующих недостатков:

- структурность легко восстанавливается после деобфускации;
- измененные имена переменных, возможно заменить на читаемые.

2 Вставка невалидных инструкций. Вставляются неописанные в стандарте языка так называемые опкоды (т.е. невалидные инструкции). Результат представлен на рис. 2.

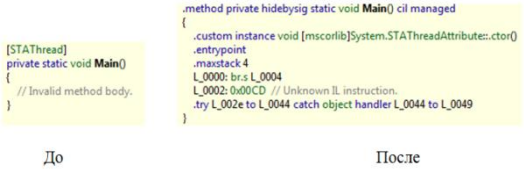

Рисунок 2 – Результат вставки невалидных инструкций

В данном методе определены следующие достоинства:

- полная потеря структурности и гибкости без возможности восстановления.

Однако в данном преобразовании имеются недостатки:

- после обфускации код становится зависимым от платформы;
- некоторые деобфускаторы способны привести код к начальному состоянию.

3 Искусственное «засорение» кода. Данный метод является простым в реализации, но обеспечивает надежную защиту от злоумышленника. Идея состоит в том, что исходный код программы «засоряется» не важной информацией, ненужными вызовами или методами. Помимо вставки длинных комментариев, вставляются комментарии, которые несут ложные пояснения дальнейшего кода. Кроме использования комментариев, при «засорении» используют вставки ненужного кода. Вызов классов, методов, функций, которые выполняются системой, но никак не влияют на работу и выполнения задачи приложения.

```
class Form
{
Form f = new Form();
f.GetShow();
}
```

Рисунок 3 – Исходный код

Данный код (см. рис. 3), запускает форму со стандартными настройками. На рис.4. и на рис.5 представлен результат двух разных способов «засорения» кода.

```
class Form /*swe324rw**/{/*sfdsfdvcx  **/From f = new Form();/*sfdsfr565**/
/*etretete**/f.GetShow();
/*21313hjk1 2**/}
```

Рисунок 4 – Вставка длинных комментариев

```
Form f = new Form();
private int i, j; // переменные
private char temp = ' ';
private String stringValue;
private String secondStringValue;
private char[] arrayStringValue;
private char[] arraySecondStringValue;
@Override
    public void actionPerformed(ActionEvent e) {
        // TODO Auto-generated method stub
        for(i=0; i<stringValue.length(); i++)// далее - алгоритм
{
for(j=0; j<secondStringValue.length(); j++)
{
if(arrayStringValue[i] == arraySecondStringValue[j])
{
arrayStringValue[i+2] = arrayStringValue[i+1];
arrayStringValue[i+1] = temp;
}
}
}
stringValue = String.valueOf(arrayStringValue); // конвертируем массив символов в строку
    }
    @Override
    public void actionMenu(ActionEvent e) {
        // TODO Auto-generated method stub

    }
f.GetShow();
}
```

Рисунок 5 – Вставка программного «мусора»

В ходе анализа данной технологии защиты программного кода были установлены следующие преимущества:

- потеря гибкости кода;
- потеря структурности кода;
- запутывание логики;
- увеличение объема;
- добавление неважного кода, который тяжело определить при деобфускации;
- добавление ложных комментариев усложняет деобфускацию.

Также были определены недостатки данной обфускации:

- легко деобфусцировать (если используется только вставка комментариев);
- вставка программного «мусора» может увеличить время выполнения программы.

В таблице 1 приведен сравнительный анализ методов обфускации программного кода, также представлены преимущества и недостатки каждого рассмотренного метода.

Таблица 1 – Сравнительный анализ методов обфускации программного кода

Результат обфускации	Метод обфускации		
	Удаление пробелов и переименование	Вставка невалидных инструкций	Искусственное «засорение» кода
Потеря структурности	+	+	+
Потеря гибкости кода	-	+	+
Возможность изменения имен переменных	+	-	+
Запутывание логики	+	-	+
Увеличение объема	-	+	+
Увеличение времени выполнения	-	-	+
Возможность легкой деобфускации	+	+	-
Зависимость от платформы	-	+	-

Исходя из таблицы 1, можно сделать вывод, что самым надежным методом обфускации является искусственное «засорение» кода. Этот метод гарантирует надежную устойчивость к деобфускации, но может повлиять на время выполнения программы. Самым ненадежным методом является удаление пробелов и переименование переменных, данный метод не гарантирует защиту от копирования и легко подвергается деобфускации. Вставка невалидных инструкций запутывает логику и структуру, но делает программу платформозависимой, а некоторые деобфускаторы могут преобразовать опкоды в начальный вид.

Выводы. Проведен анализ технологий и методов обфускации программного кода, с целью защиты от использования злоумышленниками. Установлены преимущества и недостатки методов, проведен сравнительный

анализ. Из приведенных результатов был сформулирован следующий вывод: для обеспечения надежной защиты исходного текста программы, необходимо комбинировать рассмотренные методы обфускации, используя только их преимущества такие как: запутывание логики, потеря структурности кода, потеря гибкости кода, изменение имен переменных, искусственное «засорение». Важно, чтобы результат обфускации не влиял на время выполнения программы и был устойчив к деобфускации.

Список литературы

1. Умяров Н.Х., Губенко Н.Е. Анализ и выбор методов защиты программного продукта от копирования с использованием обфускации - Материалы VI международной научно-технической конференции.–Д.:ДонНТУ , 2011.

2. Моргайлов Д.Д., Губенко Н.Е., Чернышева А.В. Система обфускации программного кода для языка PHP - Научные труды ДонНТУ. – Д.:ДонНТУ, 2012.

3. Преодоление обфускации / Интернет-ресурс. - Режим доступа: www/ URL: habrahabr.ru/post/7446- Загл. с экрана.

4. Чернов А. В. Анализ запутывающих преобразований программ / Интернет-ресурс. - Режим доступа:www/URL: citforum.ru/articles – Загл. с экрана.

5. Панов А. Реверсинг и защита программ от взлома – П.:БХВ-Петербург, 2006, 245стр.

6. Обфускация и защита программных продуктов / Интерет-ресурс. – Режим доступа: www/URL: citforum.ru/security/articles/obfus/- Загл. с экрана.

7. Варновский Н.П., Захаров В.А., Чернов А.В., Шокуров А.В. Об особенностях применения методов обфускации программ для информационной защиты – Труды института системного программирования РАН.Том 11, 2006.