ЛЕКСИЧЕСКАЯ ОБФУСКАЦИЯ КАК СПОСОБ ВНЕДРЕНИЯ ВОДЯНЫХ ЗНАКОВ В ИСХОДНЫЕ КОДЫ ПРОГРАММ И ПРОЕКТНЫХ ОПИСАНИЙ

Видничук В. Н., Иванюк А. А.
Белорусский государственный университет информатики и радиоэлектроники
Минск, Республика Беларусь
E-mail: vidnichuk@bsuir.by, ivaniuk@bsuir.by

Предложен метод внедрения водяных знаков на основе лексической обфускации исходных кодов программ, его основные положения. Предложены символы использующиеся для обфускации и внедрения водяных знаков.

Введение

В современном мире, в связи с быстрым ростом производимых компаниями программ и проектных описаний растёт и проблема пиратства. Для решения данной проблемы предлагается рассмотреть метод внедрения водяного знака в метод лексической обфускации программного кода.

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

Основной целью лексической обфускации является переименование названия переменных и идентификаторов на похожие друг на друга. В большинстве языков программирования названия переменных можно представить в виде множества символов:

$$< ID >= \{[A..z] \cup [0..9] \cup [\_]\},$$

где, название переменной должно начинаться с буквы и может состоять из латинских символов и цифр. Также, некоторые языки программирования, поддерживают кипилические и специальные символы. В связи с этим для выполнения метода переименования переменных в разных языках программирования, используются разные наборы символов.

I. Метод переименования переменных

Это один из основных методов лексической обфускации кода. Данный метод заключается в том, что мы переименовываем все переменные и идентификаторы в похожие друг на друга. В начале проверяется количество переменных и выбирается длинна для переименованных. Далее с помощью равномерного распределения строится первоначальная переименованная переменная, которая состоит из символов ’0’ и ’1’. В результате получается название переменной вида: ’100101001011’. Используя сгенерированное начальное название переменной выполняется генерация остальных названий путём замены ’0’ и ’1’ на символы визуально похожие на них из набора символов замены. При использовании данного метода следует знать, что все идентификаторы должны быть похожи и отличаться друг от друга только на 1 символ, в связи с этим существует возможность внедрения водяных знаков в его, путём выбора мест замены символов в соответствии с водяным знаком.

II. Лексическая обфускация и внедрение водяных знаков в VHDL-описания

В языке проектирования VHDL множество символов, доступных для названия переменных представленно следующим множеством:

$$< ID >= \{[A..z] \cup [..] \cup [0..9] \cup [\grave{A}..\grave{z}]\},$$

где Á .. ź является множеством диакритических и специальных символов, которые на этапе синтеза VHDL-описания приводятся к соответствующему символу без диакритического знака. Например, диакритическая Õ приводится к обычному латинскому символу O, однако, имеют разные коды символов, поэтому при выполнении лексического анализа обфусцированного кода простейшими лексическими деобфускаторами могут возникать ошибки и непонятности. Отсюда следует возможность использования диакритических символов для формирования идентификаторов повышенной сложности. Можно рассмотреть следующий пример обфускации с использованием диакритических символов: идентификатор OlOOllOIlOlÖ при синтезе приведётся к идентификатору OlOOllOIlOlO , однако, идентификатор OlOOllOIlOlOŐ также при синтезе приведётся к идентификатору OlOOllOIlOlO и при поиске идентификатора, содержащего диакритический символ найдётся только он сам, отсюда следует, что в объявлении и использовании идентификаторов мы можем применять разные диакритические символы, что затруднит понимание кода, но не повлияет на результат синтеза. Следовательно, с помощью использования данных символов можно запутывать не только человека, но и программы для деобфускации кода. Например, идентификатор

OlO'OllOIlOlÖ, где 'O является кириллическим символом O и идентификатор OlOOllOIlOlÖ для человека выглядят идентичные, однако, при синтезе являются разными идентификаторами. В связи с этим было выделено следующее множество похожих символов, использующихся для лексической обфускации VHDL-описаний для замены единичных символов:

$$< 1mn >= \{1, l, i, j\}.$$

А для замены нулевых символов используется множество:

$$< 0mn >= \{0, O, O', Q\}.$$

После формирования первичного обфусцированного идентификатора выполняется внедрение водяного знака. Это осуществляется при формировании остальных идентификаторов. Анализируются количество уникальных идентификаторов в описании, и формируется ключ водяного знака. Данный ключ определяет какие из символов первоначально сгенерированного идентификатора могут изменяться при формировании названий остальных идентификаторов. Далее, для усложнения понимания исходных описаний, случайные символы 'O', 'I', 'l' заменяются на диакритические из множеств:

<DO> = [Ó, Ő, Ò, Ô, , , Ö, , Õ, Ø,].

<D1> = [Í, ł, , Í, Ì, Î, , İ, Ï, , ,].

Что не позволяет легко деобфусцировать и разобрать зашифрованный код.

III. Лексическая обфускация и внедрение водяного знака в языке C/C++

В языке программирования C/C++ множество символов, доступных для названия переменных выглядит следующим образом:

$$< ID >= \{[A..z] \cup [0..9] \cup [\_]\}.$$

В связи с этим набор символов для замены ограничен и представлен следующим множеством:

$$< ID' >= \{O, 0, Q, 1, l, I\}.$$

Для внедрения водяного знака в код предлагается использовать следующий метод: формируется первоначальная переменная, состоящая из '0' и '1'. Далее случайные символы этой переменной меняются на подобные. Получается обфусцированная первоначальная переменная, которая используется для генерации остальных переменных путём замены 1 из символов на подобный по ключу водяного знака.

IV. Лексическая обфускация и внедрение водяного знака в языке Java

Множество символов, доступных для использования в названиях переменных на языке программирования Java:

$$< ID >= \{[A..z] \cup [..] \cup [0..9] \cup [\grave{A}..\grave{z}]\}.$$

В отличие от языка проектирования VHDL, диакритические символы не заменяются на латинские из стандартного алфавита, поэтому усложнить понимание путём внедрения диакритических символов не представляется возможным. Метод лексической обфускации выглядит следующим образом: формируется первоначальный идентификатор, состоящий из '0' и '1', далее формируются замены из множества замен похожих символов. Для '0' множество выглядит следующим образом:

$$< ID >= \{[\grave{O}..\bar{o}] \cup [\grave{Q}..\bar{q}]\}.$$

Для '1' следующее мнножество замены:

$$< ID >= \{[\grave{I}..\bar{i}] \cup [\grave{J}..\bar{j}] \cup [\grave{L}..\bar{l}]\}.$$

После формирования первичных названий переменных создаются остальные по ключу водяного знака путём замены 1 символа, находящегося на месте возможной замены на один из множеств замены.

V. Выводы

В данной статье был предложен метод лексической обфускации на основе диакритических символов, что усложняет понимание исходных кодов программ и проектных описаний, а также предложен метод внедрения водяных знаков в лексически обфусцированный код.

1. Collberg, C. A Taxonomy of Obfuscating Transformations / C. Collberg, C. Thomborson, D. Low – Auckland: Department of Computer Science, 1997. – 36 p.
2. Сергейчик В. В. Методы лексической обфускации VHDL-описаний / В. В. Сергейчик, А. А. Иванюк // Информационные технологии и системы 2013 (ИТС 2013) : материалы международной научной конференции, БГУИР, Минск, Беларусь, 23 октября 2013 г. = Information Technologies and Systems 2013 (ITS 2013) : Proceeding of The International Conference, BSUIR, Minsk, 24th October 2013 / редкол.: Л. Ю. Шилин [и др.]. – Минск : БГУИР, 2013. – С. 198-199.
3. Видничук, В. Н. Использование лексической обфускации VHDL-описаний для внедрения водяных знаков / В. Н. Видничук // Компьютерные системы и сети: 55-я юбилейная научная конференция аспирантов, магистрантов и студентов, Минск, 22-26 апреля 2019 г. / Белорусский государственный университет информатики и радиоэлектроники. – Минск, 2019. – С. 191 - 193.
4. Garg S., Gentry C., Halevi S., Raykova M., Sahai A., and Waters B. «Candidate indistinguishability obfuscation and functional encryption for all circuits.» FOCS 2013.