УДК 004.056.5

ЗАЩИТА ИСХОДНОГО КОДА ВЕБ-ПРИЛОЖЕНИЙ МЕТОДОМ ЛЕКСИЧЕСКОЙ ОБФУСКАЦИИ JAVASCRIPT-КОДА

М.А. БАРТОШИК[1], В.Н. ЯРМОЛИК[2]

Белорусский государственный университет информатики и радиоэлектроники
ул. П. Бровки, 6, г. Минск, 220013, Республика Беларусь
[1]maxbartoshik@gmail.com; [2]yarmolik@bsuir.by

В настоящее время широкое развитие получили технологии создания веб-приложений с использованием HTML5, CSS и JavaScript[1]. Обратное проектирование исходного кода позволяет анализировать и модифицировать механизмы защиты приложений с целью их несанкционированного использования, что указывает на необходимость разработки технических методов защиты веб-приложений[2]. Лексическая обфускация является одним из эффективных методов затруднения процесса обратного проектирования исходного кода.

Ключевые слова: обфускация, JavaScript, защита программного обеспечения, обратное проектирование.

В последние годы возникает большое количество веб-приложений. В качестве исходного кода данных приложений используется язык JavaScript. Так как программа на данном языке выполняется на стороне клиента, любой человек имеет доступ к исходному коду приложения, может разобраться в алгоритме его работы и обойти защитные механизмы.

Для затруднения процесса обратного проектирования следует применять лексическую обфускацию исходного кода. Данный процесс включает в себя изменение имен идентификаторов и строковых литералов[3]. Суть метода заключается в преобразовании логических имен в нечитаемые и сложные для восприятия человеком последовательности символов. Формально такое преобразование можно представить в виде:

$$\{\mathrm{V},\mathrm{S}\} \overset{\mathrm{T}_{\text{лекс.}}}{\rightarrow} \{\mathrm{V}',\mathrm{S}'\} \tag{1}$$

где V - множество идентификаторов программы, S - множество строковых констант, используемых в программе, V' - множество преобразованных идентификаторов программы, S' - множество преобразованных строковых констант, $\mathrm{T}_{\text{лекс.}}$ - множество лексических преобразований.

Множество лексических преобразований представлено следующим образом:

$$T_{\text{лекс.}} \in \{N, E, R, I\}, \tag{2}$$

где N – использование различных систем счисления, E – использование кодировок, R – использование регулярных выражений, I – использование индексаторов.

Суть метода с использованием различных систем счисления основывается на способе представления целых положительных чисел в позиционных системах счисления. Целое число без знака x в b-ричной системе счисления представляется в виде конечной линейной комбинации степеней числа b [4]:

Библиотека БГУИР

$$x = \sum_{k=0}^{n-1} a_k b^k, \quad (3)$$

где a_k это целые числа, удовлетворяющие неравенству $0 \le a_k \le b-1$.

При $b > 10$ в представлении чисел в качестве значений разрядов кроме цифр используются буквы английского алфавита, что позволяет кодировать строки при помощи числовых значений. Например, для получения строки «alert» можно воспользоваться представлением десятичного числа в 33-ричной системе счисления ($b = 33$):

```
a = (12630053).toString(33); // returns “alert”
```

Использование различных кодировок позволяет именовать идентификаторы случайным образом и представлять их различными символами (ASCII, UNICODE, HEX, OCT). Особый интерес представляют незадокументированные символы, поддерживаемые различными веб-браузерами.

Регулярные выражения сложны для восприятия человеком, их использование также увеличивает время, необходимое для изучения исходного кода приложения.

Использование индексаторов позволяет конструировать строки с использованием встроенных в JavaScript возможностей, например:

```
_ = $=+[],({}+[])[++$]; // returns ‘o’
```

Результатом выполнения данного кода является символ “o”, который можно использовать для дальнейшего конструирования строк. Данная техника позволяет создавать код, в котором не используется ни одной цифры или буквы английского языка.

Данный метод позволяет увеличить время, необходимое для обратного проектирования исходного кода приложения, однако он не позволяет навсегда скрыть алгоритм работы необфусцированного кода. Это накладывает ограничения на область использования метода, однако он отлично подходит для сокрытия параметров, которые теряют свою актуальность со временем. Пусть время актуальности скрытых параметров равно $T_{\text{акт.}}$, а время взлома данных параметров – $T_{\text{взл.}}$. Тогда условие, при котором использование метода будет эффективным, можно представить следующим образом:

$$T_{акт.} \le T_{\text{взл.}}. \quad (4)$$

Список литературы

1. Building Cross-Platform Apps with HTML5 [Электронный ресурс]. – Режим доступа: http://software.intel.com/en-us/articles/building-cross-platform-apps-with-html5. – Дата доступа: 20.12.2013.
2. *Chikofsky E.J., Cross J.H.* // IEEE Computer Society. January 1990. P. 13–17.
3. *Heiderich M.* Web application obfuscation. Syngress, 2007.
4. *Фомин С.В.* Системы счисления. М., 1987.

Библиотека БГУИР