УДК 004.492.2:347.7

Алгоритм динамической обфускации информации с ограничением количества попыток расшифровки, исполнения и просмотра на web-клиенте

Е. О. Евстафьев
Пермский государственный национальный исследовательский университет
Россия, 614990, г. Пермь, ул. Букирева, 15
Evgeniy.Evstafev@tplusgroup.ru; +7-902-83-74-137

С. Ф. Тюрин
Пермский государственный национальный исследовательский университет
Россия, 614990, г. Пермь, ул. Букирева, 15
tyurinsergfeo@yandex.ru; +7-952-32-02-510

Рассмотрена нетривиальная проблема защиты авторских прав информации, размещенной и передаваемой с помощью телекоммуникационной сети Интернет. Предложен вариант реализации "запутывающего" алгоритма, который подразумевает динамическую смену функциональной составляющей, и доступен в каждый момент времени для расшифровки и исполнения web-клиентом только один раз.

Ключевые слова: *алгоритмы; обфускация; запутывающие программы; авторское право; web-технологии.*

DOI: 10.17072/1993-0550-2018-4-56-59

Введение

Со стремительным развитием информационных технологий, в частности связанных с телекоммуникационной сетью Интернет, все более актуальным становится вопрос защиты авторских прав, передаваемой и хранимой в открытом доступе информации. Основная проблема заключается в том, что информационные материалы, возвращающие web-сервера, однозначно декодируются конечными web-клиентами, а, следовательно, могут быть сохранены, несанкционированно продублированы и распространены.

Задачи запутывания программ и алгоритмов информации имеют три основных аспекта:

1. Теоретический, который включает в себя исследование и разработку алгоритмов преобразования графа потока управления или трансформации данных программы [1], а также теоретическую оценку сложности их анализа и раскрытия.

2. Прикладной аспект в свою очередь включает в себя разработку методов запутывания или распутывания (т.е. наиболее эффективных комбинаций алгоритмов), эмпирический сравнительный анализ различных методов, эмпирический анализ устойчивости методов, и т. д.

3. Этот аспект – психологический, в данный момент не поддается строгой формализации, но не может полностью игнорироваться. Обратная инженерия программ – это процесс, результатом которого является некоторое знание субъекта, изучающего программу, который является неотъемлемой частью процесса понимания [3]. На наш взгляд, методы запутывания должны максимально использовать свойства человеческой психики.

Не умаляя ценности теоретических исследований, нужно заметить, что теоретические выводы должны подтверждаться результатами практического применения предложенных методов. В данной работе исследуется прикладной аспект задачи запутывания.

В настоящее время широко распространены языки программирования, такие как JavaScript, который используется преимущественно для гипертекстовых документов. Существует большое число деобфускаторов для гипертекстовых документов как распространяемых свободно, так и коммерческих, что упрощает несанкционированное использование, обратную инженерию и модификацию информации. В качестве одного из способов борьбы с этим рассматриваются более сложные способы запутывания программ, шифрование информации, обфускация.

Учеными уже разработано около двух десятков различных обфускаторов. Большинство из них – простые минификаторы [4]. Простые запутыватели удаляют таблицы символов и отладочную информацию и заменяют исход-ные имена методов бессмысленными короткими именами (обычно по порядку a, b, c и т.д.). В результате размер документов уменьшается – до 50 %, а скорость выполнения значительно возрастает, поэтому такое запутывание может рассматриваться и как один из способов оптимизации. Более развитые запутыватели программ выполняют преобразования графа потока управления программы и ее структур данных. Методы, используемые в них, как правило, подобраны эмпирически и слабо обоснованы теоретически. Сравнительный анализ запутывателей, доступных через Интернет, проведен в работе [5].

Запутывающие преобразования можно разделить на несколько групп в зависимости от того, на трансформацию какой из компонент программы они нацелены.

- Преобразования форматирования, которые изменяют только внешний вид программы. К этой группе относятся преобразования, удаляющие комментарии, отступы в тексте программы или переименовывающие идентификаторы.
- Преобразования структур данных, изменяющие структуры данных, с которыми работает программа. К этой группе относятся, например, преобразование, изменяющее иерархию наследования классов в программе, или преобразование, объединяющее скалярные переменные одного типа в массив. В данной работе мы не будем рассматривать запутывающие преобразования этого типа.
- Преобразования потока управления программы, которые изменяют структуру ее графа потока управления, такие как развертка циклов, выделение фрагментов кода в процедуры, и другие. Данная статья посвящена анализу именно этого класса запутывающих преобразований.
- Превентивные преобразования, нацеленные против определенных методов декомпиляции программ или использующие ошибки в определенных инструментальных средствах декомпиляции.

1. Динамическая обфускация

Для частичного решения проблемы применяются так называемые "запутывающие" алгоритмы обфускации (от лат. obfuscare – затенять, затемнять; и англ. obfuscate – делать неочевидным, запутанным, сбивать с толку), описанные в [5, 6]. Простейшим примером может быть шифрование email-адресов и номеров телефонов, размещенных на web-страницах в исходном коде, но расшифровывающихся автоматически на web-клиентах [7]. Это усложняет задачи так называемых "сборщиков" персональных данных – автоматических программ, использующих их, например, в будущем для рассылки спама [8]. Однако следует понимать, что конечный пользователь с помощью своего web-клиента имеет доступ к уже расшифрованной информации, и алгоритм обфускации используется один (или несколько – не важно, можно сократить от одного до N – конечное число), а это означает, что расшифровать обфуцированный код является делом техники [9]. Для решения данной проблемы предлагается следующий алгоритм:

1. В конечном документе на стороне web-сервера, при рендеринге web-контента, производить полное шифрование исходящего трафика алгоритмом AES 256 [10]. Ключ в данном случае генерировать псевдослучайным образом и размещать в cookies, отдаваемых web-браузеру.
2. Генерировать псевдослучайные имена функций и классов, а также записывать их в заголовки cookies. Кроме этого, в произвольные части документа добавлять нефункциональный и не значимый шум.
3. С помощью JavaScript класса AES 256 [11] расшифровывать и исполнять контент, используя имена и ключи, записанные в файлах cookie.
4. При расшифровке автоматически стирать данные в cookie либо заменять псевдослучайной информацией (обязательно параллельно с расшифровкой, чтобы транзакция была согласованной).

Листинг 1. *Пример реализации динамической обфускации*

```
1. <?PHP
2.          REQUIRE ("AES.CLASS.PHP");
3.          $COOKIE_NAME =
MD5(MICROTIME()).MD5(MICROTIME()).MD5(MICROTIME());
4.          $COOKIE_VALUE =
MD5(MICROTIME()).MD5(MICROTIME()).MD5(MICROTIME());
5.          $FUNC_NAME1 = 'F'.MD5 (MI-
CROTIME()).MD5(MICROTIME()).MD5(MICROTIME());
6.          $FUNC_NAME2 = 'F'.MD5 (MICROTIME
()).MD5(MICROTIME()).MD5(MICROTIME());
7.          SETCOOKIE($COOKIE_NAME, $COOKIE_VALUE, TIME() + (86400), "/");
8.          $PR1 = RAND(1,550);
9.          FOR($X=1; $X<=$PR1; $X++) {
10.                   $PR.=' ';
11.         }
12.         $VAR [0] = AESCTR::ENCRYPT (' <BR/>EUGENE EVSTAFIEV
CHIGWEL@GMAIL.COM<BR/>'.$PR, $COOKIE_VALUE, 256);
13.?>
14. <SCRIPT TYPE="TEXT/JAVASCRIPT" SRC="AES.JS">
15. </SCRIPT>
16. <SCRIPT>
17.         VAR
_0X44ED=['LENGTH','POP','SHIFT','DECRYPT','WRITE'];(FUNCTION(_0X3BAE5B,_0X40
A6B2){VAR _0X3BBA42=FUNCTION(_0X1737E0){WHILE(--
_0X1737E0){_0X3BAE5B['PUSH'](_0X3BAE5B['SHIFT']());}};_0X3BBA42(++_0X40A6
B2);}(_0X44ED,0X1EF));VAR
_0X4132=FUNCTION(_0X25117B,_0X4AA7D3){_0X25117B=_0X25117B-0X0;VAR
_0X2FF431=_0X44ED[_0X25117B];RETURN _0X2FF431;};FUNCTION <? ECHO
$FUNC_NAME2; ?>(_0XE49201){VAR _0X1877CC=';\X20'+DOCUMENT['COOKIE'];VAR
_0X3D5629=_0X1877CC['SPLIT'](';\X20'+_0XE49201+'=');IF(_0X3D5629[_0X4132('
0X0')]==0X2)RETURN
_0X3D5629[_0X4132('0X1')]()['SPLIT'](';')[_0X4132('0X2')]();};VAR CIPHER='<?
ECHO $VAR[0];?>';VAR PASSWORD=<? ECHO $FUNC_NAME2; ?>('<?PHP ECHO
$COOKIE_NAME; ?>');VAR PLAIN-
TEXT=AES['CTR'][_0X4132('0X3')](CIPHER,PASSWORD,0X100);DOCUMENT[_0X4132
('0X4')](PLAINTEXT);
18.         VAR
_0X7D34=["\X63\X6F\X6F\X6B\X69\X65","\X3B","\X73\X70\X6C\X69\X74","\X6C\X6
5\X6E\X67\X74\X68","\X3D","\X20\X3D\X3B\X20\X65\X78\X70\X69\X72\X65\X73\X2
0\X3D\X20\X54\X68\X75\X2C\X20\X30\X31\X20\X4A\X61\X6E\X20\X31\X39\X37\X30\
X20\X30\X30\X3A\X30\X30\X3A\X30\X30\X20\X55\X54\X43"];FUNCTION <? ECHO
$FUNC_NAME1; ?>(){VAR _0XB88CX2=DOCUMENT[_0X7D34[0]];VAR
_0XB88CX3=_0XB88CX2[_0X7D34[2]](_0X7D34[1]);FOR(VAR
_0XB88CX4=0;_0XB88CX4< _0XB88CX3[_0X7D34[3]];_0XB88CX4++){VAR
_0XB88CX5=_0XB88CX3[_0XB88CX4][_0X7D34[2]](_0X7D34[4]);DOCUMENT[_0X7D
34[0]]= _0XB88CX5[0]+ _0X7D34[5]}}<? ECHO $FUNC_NAME1; ?>()
19.</SCRIPT>
```

В таком варианте мы получим следующий результат (пример реализации представлен на рис. 1, 2).

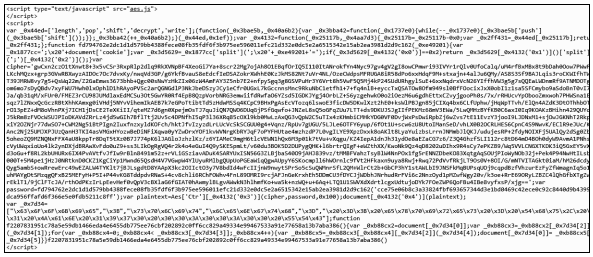

Рис. 1. *Пример обфуцированного документа*

Web-клиент, получивший исходный код гипертекстового документа в "нечитаемом" и зашифрованном виде, в процессе расшифровки избавляется от ключей дешифровки.

Рис. 2. *Расшифрованный результат (без черного обрамления)*

Это означает, что второй раз данный код расшифровать уже невозможно.

Выводы

Сегодня одновременно с развитием интернет-технологий, увеличением уровня доступности информации, скорости распространения и пр. неразрывно связаны вопросы, связанные с безопасностью этой информации, защитой, в частности, защитой авторских прав информации, передаваемой в телекоммуникационной сети Интернет.

В данной статье предложен вариант реализации "запутывающего" алгоритма, который подразумевает динамическую смену функциональной составляющей, и доступный в каждый момент времени для расшифровки и исполнения web-клиентом только один раз.

Поскольку теория запутывания программ и информации сейчас находится в стадии активного формирования, кроме того, разрабатываются все новые и новые эмпирические методы запутывания, по словам А.В. Чернова [1], методы запутывания должны учитывать свойства человеческой психики и пытаться ставить в тупик человека, который управляет системой анализа программ. Предложенный алгоритм выдает результат при каждом запросе шифрования одного и того же информационного содержимого различного по длине, наполнению и ключа, автоматически удаляемым при расшифровке, необходимым для понимания исходного документа. Это означает, что повторно расшифровать документ просто нечем. Кроме этого следует отметить простоту реализации – используется всего лишь два класса (клиентский и серверный).

При незначительных модификациях данный способ обфускации можно применять для практически любых информационных материалов, передающихся через телекоммуникационную сеть Интернет. Однако стоит отметить, что в будущем возможно усовершенствование механизма обфускации, например, следующими способами:

1. Динамическое изменение управляющего потока программы-обфускатора. Это означает, что можно в перспективе менять не только структуры используемых переменных и данных, но и саму программу, используя для выполнения различные функции и методы внутри обфускатора, желательно псевдослучайно выбранными из максимально возможного набора.
2. Применение алгоритма Лемпеля–Зива–Велча. В идеальном случае – применение данного алгоритма для преобразований

исходного кода во всех существующих на сегодня кодировках. Также стоит отметить, что его можно применять как до обфускации информации, так и после, а также и "до" и "после" одновременно. Все варианты можно чередовать псевдослучайным образом.

3. Передача ключей от сервера клиенту другими способами или передача ключей от сервера клиенту разными способами в псевдослучайном порядке.

Разумеется, предлагаемые решения и потенциальные улучшения не идеальны, потому что в любом случае клиент имеет возможность расшифровать (хоть и единожды) информационные материалы. Кроме того, клиенту должна быть предоставлена возможность физического просмотра информации, в противном случае процесс не имеет смысла. Тем не менее, в представленной реализации, человек или система, которые столкнутся с задачей деобфускации постоянно изменяющихся данных; ситуации, когда достаточно неоднозначно определить наличие и способ передачи ключей; с учетом постоянно изменяющихся кодировок (с помощью алгоритма Лемпеля–Зива–Велча), шума и пр. – в такой ситуации деобфускатор столкнется с динамическим списком трудностей для решения которых придется реализовывать более сложное решение, нежели чем предлагаемый обфускатор.

Список литературы

1. *Чернов А.* Анализ запутывающих преобразований программ: тр. Ин-та системного программирования РАН. М., 2003.
2. *Mayrhauser A. von, Vans A.M.* Program Understanding: Models and Experiments. In M. Yovits, M. Zelkowitz (eds.) Advances in Computers, Vol. 40, 1995. San Diego: Academic Press. P. 1–38.
3. *Чигиринский Е.* Microsoft Ajax Minifier (Автоматическая оптимизация JavaScript and CSS для веб сайтов высокой производительности). Конференция "HighLoad++". М., 2010.
4. *Lai H.* A comparative survey of Java obfuscators available on the Internet. February, 2001.
5. *Чернов А.В.* Интегрированная среда для исследования "обфускации" программ: докл. на конф., посвященной 90-летию со дня рождения А.А. Ляпунова. Новосибирск. 8–11 октября 2001 г.
6. *Barak B., Goldreich O., Impagliazzo R., Rudich S., Saha A., Va-dhan S., Yang K.*. On the (Im) possibility of Obfuscating Programs. LNCS. 2001; 2139. P. 1–18.
7. *Chow S., GU Y., Johnson H., Zakharov V.* An approach to the obfuscation of control-flow of sequential computer programs. LNCS. 2001; 2200. P. 144–155.
8. *Collberg C., Thomborson C., Low D.* Breaking Abstractions and Unstructuring Data Structures. In IEEE International Conference on Computer Languages, ICCL'98, Chicago. IL. May, 1998.
9. *Collberg C., Thomborson C., Low D.* Manufacturing Cheap, Resilient, and Stealthy Opaque Constructs. In Principles of Programming Languages 1998, POPL'98, San Diego, CA. January, 1998.
10. *Mortensen S.* PHP Implementation of AES encryption in CTR mode (128-bit AES, 192-bit AES, or 256-bit AES). Git-repository hosting service. GitHub Inc. July, 2012.
11. *Moore R.* A pure JavaScript implementation of the AES block cipher algorithm and all common modes of operation (CBC, CFB, CTR, ECB and OFB). Git-repository hosting service. GitHub Inc. April, 2018.

The dynamic obfuscation's algorithm of information with number restriction of interpretation's attempts, execution by the web-client

E. O. Evstafiev[1], S. F. Tyurin[2]

Perm State University; 15, Bukireva st., Perm, 614990, Russia

[1]Evgeniy.Evstafev@tplusgroup.ru; +7-902-83-74-137

[2]tyurinsergfeo@yandex.ru; +7 952-320-02-510

In this article, the uncommon problem of the protection of information copyright placed and transmitted by means of the Internet is considered. The realization option of the obfuscation's algorithm which means dynamic functional component change, and available in each time point for interpretation and execution by the web-client only once is offered.

Keywords: *algorithm; obfuscation; the confusing programs; copyright; web-technologies.*