УДК 003.26+004.431.4 DOI 10.17223/2226308X/10/30

АССЕМБЛЕР-ПРОГРАММА КАК СТЕГОКОНТЕЙНЕР ДЛЯ ТЕКСТА

Е. Ю. Грачёва, М. Я. Товштейн

Ассемблер-программа используется как контейнер для текстового сообщения, которое может закрываться криптографическим шифром. Символы внедряемого текста представляются командами ассемблера (например, ADC, ADD, MOV, SUB), использующими регистры AX, BX, CX, DX. Встроенные команды незаметны среди исходных команд контейнера. Заполненный контейнер выдерживает атаку компиляцией и реализацией.

Ключевые слова: *ассемблер, текстовая стеганография, стегоконтейнер.*

Использовать текст в качестве стегоконтейнера для текстового сообщения — чрезвычайно интересная задача, для её решения предложены различные методы. Например, можно менять высоту отдельных знаков, настройку шрифта или параметров, формирующих цвет знака, варьировать количество пробелов между фрагментами текста, изменять порядок следования маркеров конца строки CR/LF, изменения междустрочного расстояния неотображаемых символов строк и т. п. [1 – 4]. Достоинство этих методов в том, что они применимы к любому тексту. Мы хотим поделиться опытом применения текста специального вида в качестве контейнера — текста компьютерной программы на языке ассемблера [5].

Почему выбран именно язык ассемблера? Во-первых, этот язык сейчас знают очень мало программистов. Это позволяет надеяться, что не много охотников обратят внимание на такую программу и будут её анализировать без веских причин. Во-вторых, в команды этого языка можно преобразовать ASCII-коды символов, которые обычно используются в текстовых сообщениях. В-третьих, ассемблер-программы часто довольно объёмны, это не вызывает подозрений у программистов. В-четвёртых, команды, представляющие встроенное сообщение, очень похожи на команды пустого контейнера, и это помогает выдержать визуальную стегоатаку.

Текстовую закладку в ассемблер-программу можно предварительно зашифровать каким-либо криптографическим методом. Условимся в последующем изложении криптографическую сторону работы не затрагивать, а шифртекст именовать просто сообщением.

Каждый символ сообщения заменяется командой ассемблера. Для этого 8 разрядов ASCII-кода символа делятся на три части: первые две — по 2 бита и третья — 4 бита. Первые два бита заменяются обозначением одной из команд ассемблера так:

00 — ADC, 01 — ADD, 10 — MOV, 11 — SUB.

Вторые два бита заменяются именем регистров AX, BX, CX, DX:

00 — AX, 01 — BX, 10 — CX, 11 — DX.

Последние 4 бита — это 16-я константа.

Создаётся пара стегоключей K_1 и K_2, которые предварительно сообщаются адресату по закрытому каналу связи. Они также преобразуются в команды ассемблера. Команда-ключ K_1 вставляется в контейнер до сообщения, а команда-ключ K_2 — после него. Таблица показывает игрушечный текст «$X1 = a + b$;», ключи K_1 =«Ст» и K_2 =«ог» в виде двоичного кода и ассемблер-команд.

Как сообщение отфильтровывается из стегоконтейнера? Программа СтегоАссемблер принимает как входные данные ключи K_1 и K_2, превращая их в команды ассем-

Символы-закладки как команды ассемблера

Вставка	ASCII-коды	Команда	Ассемблер
С	209	11 01 0001	SUB BX, 1
т	242	11 11 0010	SUB DX, 2
X	88	01 01 1000	ADD BX, 8
1	49	00 11 0001	ADC DX, 1
=	61	00 11 1101	ADC DX, 13
a	65	01 00 0001	ADD AX, 1
+	43	00 10 1011	ADC CX, 11
d	100	01 10 0100	ADD CX, 4
;	59	00 11 1011	ADC DX, 11
о	238	11 10 1110	SUB CX, 14
г	227	11 10 0011	SUB CX, 3

блера. Затем в стегоконтейнере она находит «K_1»-команду и считывает следующие за ней команды, придавая им исходный вид, пока не встретит «K_2»-команду.

Интересно сопоставить описанный метод текстовой стеганографии с тем, который представлен в системе STEGANO-1S [4]. Она работает с текстовыми и графическими файлами и предназначена для скрытой передачи информации, внедрения водяных знаков и отпечатков пальцев. Авторы системы поясняют суть применяемого в ней метода: «При работе с текстовыми файлами система использует технологию форматирования текста количеством пробелов, отличным от единицы. Двоичным символам встраиваемого текста сопоставляется количество пробелов между словами файла-контейнера, например, один пробел соответствует биту 0, а два пробела — биту 1».

На один символ встраиваемого текста в предельном случае используется 7 бит, так что в контейнер с количеством одиночных и двойных пробелов, равным N, можно внедрить в среднем $N/7$ символов сообщения. STEGANO-1S работает с любыми текстами, а не только с программами на алгоритмическом языке. После внедрения закладки в контейнер он остаётся внешне неизменённым и удобочитаемым. Кроме того, стегоконтейнер может быть превращён в исполняемый файл и выполнен без проблем [4].

В нашем способе (см. таблицу) один символ сообщения увеличивает текст контейнера в среднем на 10 символов. И хотя СтегоАссемблер разрешает встраивать в контейнер довольно большой текст, желательно всё же, чтобы размер этого текста был не больше 10 % размера программы-контейнера, иначе контейнер может быть подвержен стегоатаке при сопоставлении размеров.

Надо, однако, заметить, что СтегоАссемблер позволяет выполнять перевод сообщения в ассемблерный вид не только так, как было описано выше. Можно, например, использовать принцип «то, что лежит на виду, часто остаётся без внимания». Вот объявление одномерного массива:

mas1 db 88, 49, 61, 65, 43, 100, 59

Как догадаться, что этот массив представляет текст «$X1 = a + b$;»?

Такой приём встраивания удобен ещё и тем, что название массива играет роль стегоключа. Кроме того, возможно объявление двумерных массивов, в которых элементы располагаются по определённому правилу, согласованному с адресатом. А это уже элемент криптографической защиты. Разрешается объявление нескольких массивов, размещённых в разных местах контейнера.

СтегоАссемблер ориентирован только на ассемблерную программу в качестве контейнера, а не на любой текст, как STEGANO-1S. Однако так же, как и STEGANO-1S,

программа СтегоАссемблер позволяет: внедрять сообщение большого размера, хоть и в определённой пропорции к размеру исходной программы; снимать подозрения в наличии закладки в заполненном контейнере, превращая стегоконтейнер в исполняемый файл с последующей реализацией; выдерживать визуальные стегоатаки, учитывая, что язык ассемблера знает не столь широкий круг программистов.

ЛИТЕРАТУРА

1. *Cox I. J., Miller M. L., and Bloom J. A.* Digital Watermarking. London: Morgan Kaufmann, 2002. 542 p.
2. *Шутько Н. П.* Алгоритмы реализации методов текстовой стеганографии на основе модификации пространственно-геометрических и цветовых параметров текста // Труды БГТУ. Сер. VI: Физ.-мат. науки и информатика. 2016. Вып. 6. С. 160–165.
3. *Блинова Е. А.* Стеганографический метод на основе изменения междустрочного расстояния неотображаемых символов строк электронного текстового документа // Труды БГТУ. Сер. VI: Физ.-мат. науки и информатика. 2016. Вып. 6. С. 166–169.
4. *Ярмолик С. В., Ярмолик В. Н.* Стеганографическая система передачи информации «Stegano-1S» // Технические средства защиты информации: тезисы докл. II Белорусско-российской науч.-технич. конф. (Минск–Нарочь, 17–21 мая 2004 г.) С. 54–62. `http://www.doklady.bsuir.by/m/12_104571_1_56218.pdf`
5. *Абашев А. А., Жуков И. Ю. и др.* Ассемблер в задачах защиты информации. М.: КУДИЦ-ОБРАЗ, 2004. 544 с.

УДК 004.056 DOI 10.17223/2226308X/10/31

О ВЕРИФИКАЦИИ СОБСТВЕННОРУЧНОЙ ПОДПИСИ

А. В. Епишкина, А. В. Береснева, С. С. Бабкин, А. С. Курнев, В. Ю. Лермонтов

Проанализированы способы онлайн-верификации собственноручной подписи на основе KNN-алгоритма, Range Classifier алгоритма, алгоритма на основе скрытой модели Маркова и простейшей перцептронной нейронной сети. Особенности применения данных алгоритмов исследованы в ходе реализации и тестирования с целью дальнейшей модификации и разработки наиболее эффективного по всем параметрам метода верификации.

Ключевые слова: *верификация собственноручной подписи, скрытые модели Маркова, нейронные сети.*

1. Подходы к верификации собственноручной подписи

На данный момент разработано несколько различных подходов к задаче верификации собственноручной подписи. Автономная система проверки подписей, представленная в [1], построена на основе нескольких статистических методов, в частности используются скрытые модели Маркова (СММ) в построении эталонной модели для каждого локального объекта.

Другая система, предложенная в [2], основана на машинном обучении. Для применения машинного обучения при верификации подписи необходима обучающая выборка. В процессе исследования изучалась возможность применения таких алгоритмов, как KNN, метод опорных векторов и метод логистической регрессии.

В [3] описана методика верификации подписи, которая основана на использовании нейронной сети. Для каждого объекта устанавливается специальный двухступенчатый