УДК:004.49

# ВЫЯВЛЕНИЕ ВРЕДОНОСНОГО ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ НА ОСНОВЕ СТАТИЧЕСКОГО АНАЛИЗА

**Рахимбаев Нурмуханбет Динмухаммедович**
777rnd777@gmail.ru
Студент 4-го курса специальности «5В070400 - Вычислительная техника и ПО»
стажер ТОО «T&T Security»
**Абдиев Бауржан Сапаралиевич**
abdibaur@gmail.com

Докторант 1-го курса специальности «6D070400 - Вычислительная техника и ПО»
ЕНУ им. Л.Н.Гумилева, Нур-Султан, Казахстан
Научный руководитель – Д.Ж. Сатыбалдина

Статистика инцидентов и многочисленные исследования показывают, что вредоносные программы представляют собой одну из наиболее значительных угроз компьютерной безопасности [1, 2]. Вредоносное программное обеспечение (ПО) – это программы, предназначенные для нанесения вреда компьютерам, сетям и пользователям путем кражи информации, повреждения файлов, нарушения функционирования сети, доступности ресурсов или просто совершения вредных действий, раздражающих пользователей. Вредоносные программы всегда были угрозой для цифрового мира, но с быстрым ростом использования Интернета воздействие вредоносных программ становится серьезным. Поскольку почти все сферы жизни используют Интернет для повышения качества обслуживания, возрастает необходимость обнаруживать и деактивировать вредоносные программы как можно раньше, чтобы можно было избежать негативных результатов, создаваемых ими. Хотя исследователи разрабатывают новейшие технологии для своевременного обнаружения вредоносных программ, но и создатели вредоносных программ постоянно улучшают методы сокрытия или трансформации вредоносного кода. Эти идеи связаны как с шифрованием кода, так и созданием олигоморфных, полиморфных и метаморфических вирусов [3].

Традиционные методы обнаружения вредоносного ПО на основе сигнатур хороши при обнаружении известных вредоносных программ, но они не могут обнаружить неизвестные вредоносные программы и полиморфные вредоносные программы, потому что они могут изменять свои сигнатуры. Эвристические методы могут обнаруживать как новые, известные, так и неизвестные вредоносные программы, но они имеют высокий уровень ложных срабатываний, что приводит к необходимости разработки более точных методов детектирования. В связи с ограниченностью существующих методов обнаружения вредоносных программ, в условиях постоянно развивающихся методов скрытия вредоносного функционала, методы машинного обучения и интеллектуального анализа данных объединяются с существующими методами статического и динамического анализа данных [4, 5].

Чтобы обнаружить вредоносные программы, сначала необходимо проанализировать, как вредоносное ПО выполняет свою функцию, и какова конечная цель разработки вредоносных программ. Понимание такого рода облегчает разработчикам детекторов как эффективно обнаруживать вредоносных функции, так реализаовать защитные функциональности. Методы анализа вредоносного ПО делятся на три категории в зависимости от времени и техники проведения анализа: статические, динамические и гибридные техники [2].

Когда программное обеспечение или фрагмент кода анализируется без выполнения, этот вид анализа называется статическим анализом или анализом кода. Статическая информация извлекается из кода, чтобы определить, содержит ли программа вредоносный код или нет. В этом методе вредоносное или потенциально вредоносное ПО реверс инжинирингу с использованием различных инструментов (отладчик, дизассемблер, декомпилятор и анализаторы исходного кода). Методы, которые используются при выполнении статического анализа, включают проверку формата файла, извлечение строк, цифровую дактилоскопию, антивирусное сканирование и дизассемблирование.

Когда функциональные возможности ПО анализируются и наблюдаются при исполнении исходного кода, это называется динамическим или поведенческим анализом [6]. Это может быть сделано путем отслеживания вызовов функций, потоков управления, а также путем анализа инструкций и параметров функций. Вредоносный код запускается в виртуальной среде для наблюдения за его поведением и разработки действий против этого негативного поведения. Инструменты, которые используются для динамического анализа:

песочница, симулятор, эмуляторы и приложения для мониторинга, например, Process Explorer. Этот тип анализа занимает больше времени, так как нужно спроектировать среду для выполнения и тестирования вредоносного ПО.

Гибридная техника сочетает в себе методы статического и динамического анализа, поэтому может использовать преимущества обоих подходов. Сначала ПО отслеживается анализом кода путем проверки сигнатуры вредоносной программы, а затем запускается в виртуальной среде для наблюдения за его реальным поведением.

В настоящей работе представлены результаты проектирования и программной реализации статическиго анализа исходного кода с целью выявления вредоносной функциональности. В качестве языка разработки выбран компилируемый язык программирования Go (Golang) Google.

Вредоносное ПО может быть создано на любом языке программированияиз них, но не на каждой ЭВМ оно запустится. Такие языки как Python, Go, Java требуют своей собственной установки, так что их не встретишь на каждом компьютере. В качестве потенциально опасных объектов были выбраны:

- Batch: последовательность команд из командной строки;
- JavaScript: используется повсеместно в web-разработке;
- Powershell: исполнение команд на удаленных и локальных системах;
- Visual Basic Script: используется вместе с HTML Application и серверный программный код;

Они могут запуститься без каких-либо приготовлений.

При работе со скриптом, в первую очередь, определяется язык, на котором он был написан. Данный этап был реализован при помощи специальных сочетаний команд, присущих только одному из этих языков. Например:

- Batch: echo + off;
- JavaScript: try + catch;
- Powershell: if + $;
- Visual Basic Script: Try + End;

В случае неточного ответа код проверяется на наличие тех или иных команд из каждого языка. Например:

- Batch: sc;
- JavaScript: alert;
- Powershell: New-ItemProperty;
- Visual Basic Script: Dim;

Для чистоты поиска из кода удаляются все строки, комментарии и переменные. Расширение файла, который отправится на проверку, не имеет значения, так как анализ идет по тому, что в нём содержится.

Вторым этапом идёт создание баз данных для переменных и функций. Функции рассматриваются, как отдельный код программы на уже определенном языке.

Третьим этапом идет поиск потенциальных угроз. Осуществляется проверкой кода на наличие тех или иных команд. Если к этому времени не был выявлен однозначный ответ на то, на каком языке программирования был написан скрипт, то выносится финальный вердикт.

Заключительным этапом идёт проверка на наличие реальных угроз в зависимости от языка и результата поиска потенциальных угроз. Например, если идёт работа с JavaScript или Visual Basic Script, то при работе с реестром в обоих языках будет использована библиотека WScript.Shell. Без неё обращение к реестру недопустимо, соответственно этот функционал безопасен. В противном случае программа выдаст всё обращение к реестру. Если угроза не показалась в потенциальном списке, то в реальном она не может возникнуть.

Ниже представлены примеры работы разработанной программы статического анализа кода. На рисунке 1 представлен фрагмент анализируемого кода из файла download.vbs, на рисунке 2 – результат статического анализа данного файла.

```
Dim xHttp: Set xHttp = CreateObject("Microsoft.XMLHTTP")
Dim bStrm: Set bStrm = CreateObject("ADODB.Stream")
xHttp.Open "GET", "http://example.com/someimage.png", False
xHttp.Send

bStrm.Type = 1 '//binary
bStrm.Open
bStrm.Write xHttp.responseBody
a = "c:\temp\"
bStrm.SaveToFile a & "someimage.png", 2
End With
```

Рисунок 1. Фрагмент анализируемого кода download.vbs

Как видно из рисунка 2, в начальной строке указывается название входного файла, на следующей строке выводится название язык программирования, на котором был написан анализируемый файл. После этого выводятся потенциальные угрозы, реально реализуемые угрозы (данный файл при исполнении начинает скачивание файла из Интернета и передает файлы через Интернет).

```
download.vbs
Script language: Visual Basic

Possible dangers:
May open a file
May write to a file
May create a file
May download files from the Internet
May upload files to the Internet
May work with HEX

Real dangers:
Tries to create a file:
        Path: c:\temp\someimage.png
Tries to download files from the Internet:
        URL: http:\example.com\someimage.png
        File: c:\temp\someimage.png
Tries to upload files to the Internet:
        File: createobject
        URL: http:\example.com\someimage.png
```

Рисунок 2. Результат статического анализа кода.

На рисунке 3 представлен фрагмент анализируемого кода из файла hex_autorun.vbs, на рисунке 4 – результат статического анализа данного файла.


Рисунок 3. Фрагмент анализируемого кода из файла hex_autorun.vbs

В результате проведенного статического анализа кода установлено, что скрипт из файла hex_autorun.vbs при исполнении может создать новый файл и определить его в реестре на автозапуск.

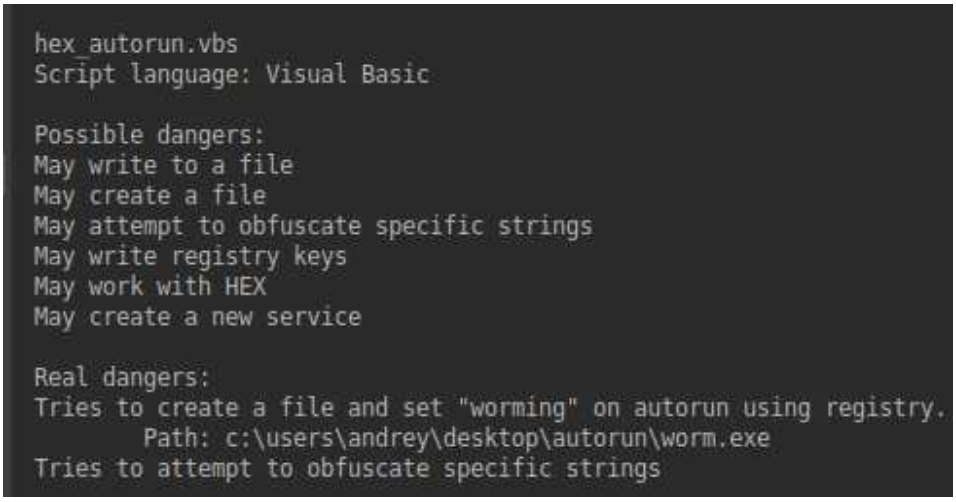


Рисунок 4. Результат статического анализа кода для примера 2

Таким образом, можно заключить, что разработанное ПО и сформированный вектор атрибутов наличия вредоносности в исходных кодах на разных языках программирования, позволяют выполнить статический анализ кода с высокой степенью обнаружения потенциальных угроз безопасности. Предложенный подход был протестирован на реальных семействах вредоносных программ. Результаты тестирования показали, что используемый подход обеспечивает согласованные результаты по сравнению с аналогичными разработками.

Научно-исследовательская работа выполнена в рамках дипломного проектирования по теме «Идентификация типа скрипта по синтаксису и выявление индикаторов вредоносности на платформе tLab». Данное направление исследований было инициировано компанией-работодателем – ТОО «**T&T security**» (генеральный директор, PhD, А. Тохтабаев).

**Список использованных источников**

1. https://www.av-test.org/en/statistics/malware/

2. Rabia Tahir, A Study on Malware and Malware Detection Techniques //I.J. Education and Management Engineering. – 2018.- № 2. – Pp. 20-30. DOI: 10.5815/ijeme.2018.02.03

3. Kim D. et al. (2017) DynODet: Detecting Dynamic Obfuscation in Malware. In: Polychronakis M., Meier M. (eds) Detection of Intrusions and Malware, and Vulnerability Assessment. DIMVA 2017. - Lecture Notes in Computer Science, Springer, Cham. - vol.10327. - pp 97-118.

4. Tokhtabayev A., Kopeikin A., Tashatov N., Satybaldina D. Malware Analysis and Detection via Activity Trees in User-Dependent Environment. In: Rak J., Bay J., Kotenko I., Popyack L., Skormin V., Szczypiorski K. (eds) Computer Network Security. MMM-ACNS 2017. Lecture Notes in Computer Science, Springer, Cham. - 2017.- vol. 10446. – pp. 211-222. DOI:10.1007/978-3-319-65127-9_17

5. Kopeikin A., Tokhtabayev A., Tashatov N., Satybaldina D. tLab: A System Enabling Malware Clustering Based on Suspicious Activity Trees. In: Rak J., Bay J., Kotenko I., Popyack L., Skormin V., Szczypiorski K. (eds) Computer Network Security. MMM-ACNS 2017. Lecture Notes in Computer Science, Springer, Cham. - 2017.- vol. 10446. – pp. 195-210. DOI:10.1007/978-3-319-65127-9_16

6. Tokhtabayev, A.G., Skormin, V.A., Dolgikh, A.M. Expressive, efficient and obfuscation resilient behavior based IDS. In: Gritzalis, D., Preneel, B., Theoharidou, M. (eds.) ESORICS 2010. Lecture Notes in Computer Science, Springer, Heidelberg. - 2010.- vol. 6345, pp. 698–715. DOI: 10.1007/978-3-642-15497-3_42