Summary report of Avast Scholar program

Marek Milkovič

February 1, 2018

1 Introduction

This paper summarizes the work and progress in Avast Scholar program lead by Avast Software at Faculty of Information Technology, Brno University of Technology during the year 2017.

Since Avast Scholar program started, one paper with title *Extraction of Information from .NET Executables Files* [7] was published at International Masaryk Conference 2017 [3]. This summary report shortly describes what was the paper about and where is the further research heading.

2 Research

Research in Avast Scholar program and my PhD thesis is focused on extraction of information from binary files and creation of detection patterns our of the extracted features. After the initial research and the consultation with the supervisor we have diceded to start with the extraction of information from .NET executable files.

3 .NET

In the chapter, we briefly describe the paper *Extraction of Information from .NET Executables Files* that we published.

3.1 Motivation

The occurrence of malicious software written in .NET languages is rapidly increasing. Extracting the information out of .NET executable file is therefore necessary step in order to fight against this kind of malware [7]. We can do various things with the extracted information like clustering, creation of detection patterns and many more.

There are currently multiple sources that deals with how data types are encoded in .NET executable files such as official documentation of CIL (Common Intermediate Language) [2] or unofficial documentation in form of book *.NET IL Assembler* [5]. However, there are no sources which deal with type reconstruction.

From the analytical software point of view, there are many .NET disassembler which perform type reconstruction such as `dnlib` [1], `monodis` [8] and `ikdasm` [4]. Their problem

mostly lies in their unreliability when dealing with obfuscated and packed executable files because they tend to crash on such files. They are also written in C# and therefore require .NET runtime to be present on the system so they cannot be integrated into C/C++ toolset. Another problem is that we cannot be sure how these third party tools load executable files into memory. If they are not loaded in reflection-only mode then accidental execution of the potentially harmful code may happen.

In the paper, we present methods which address all these problems and solve them.

3.2 File format

Despite that .NET executable files are stored in PE (Portable Executable) [6] file format which is also used for the native applications, we need to approach it different way. Some of the properties which are prevalent for native application can be neglected in case of .NET executables such as import table. Basic layout of PE format is shown in Figure 1.

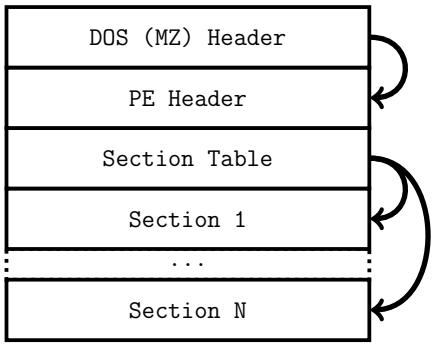

Figure 1: Structure of executable file in PE format [7].

PE header contains pointer to the structure which is called CLR (Common Language Runtime) data directory that further points to the so called *streams*. Every single *stream* contains different kinds of data based on its name. The most interesting stream for us is *metadata stream*, denoted by name `#~`, which contains various tables. These tables describe .NET data types such as classes, methods, attributes, properties and many others. Hierarchy of these structures is depicted in Figure 2.

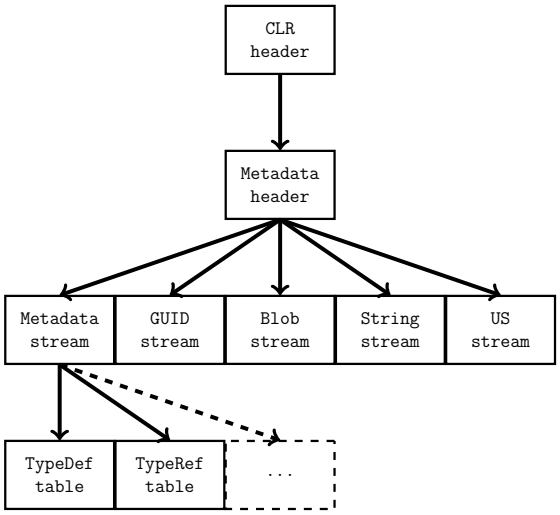

Figure 2: Hierarchy of structures for .NET executable file [7].

3.3 Extracted features

We propose methods to extract various kinds of types out of .NET executable files:

- Classes
- Methods

- Properties
- Fields
- TypeLib identifier
- Module Version identifier

Because of their complexity, methods to extract all of these features is not going to be mentioned in the summary report. You can look them up in the published paper.

3.4 Results

We implemented proposed methods in C++ as part of tool `fileinfo`, which is developed by Avast Software as part of open-source project RetDec [9].

We then compared out implementation with other .NET disassemblers. The interesting for us was robustness when dealing with corrupted, packed or obfuscated binaries. We have downloaded 100 000 appropriate samples from VirusTotal [10] and evaluated the results on this set. Results can be seen in Table 1.

Table 1: Success rate results.

	Errors	**Crashes**	**Timeouts**
`fileinfo`	0	0	1
`monodis`	905	47042	6
`ikdasm`	9342	149	192

We have also focused on the performance and therefore compared the running time of `fileinfo` with different implementations. Running time means how long were tests actually running on all 100,000 samples. Total time is sum of all durations of every single instance of tested program.

Table 2: Performance results.

	Running time	**Total time**
`fileinfo`	6m 45s	9h 45m 56s
`monodis`	1m 47s	1h 47m 12s
`ikdasm`	26m 36s	1d 13h 46m 26s

4 Conclusion

In our research of .NET executable files, we have proposed robust methods to extract different kinds of information. We have implemented them, compared with the existing solutions, achieving better results than them. The paper was accepted and published at International Masaryk Conference 2017. We are still continuing with the research, trying to find new features that can be extracted and could be used for malware detection. We are also researching methods to build detection patterns out of extracted features.

References

[1] *0xd4d/dnlib: dnlib is a library that can read, write and create .NET assemblies and modules.* `https://github.com/0xd4d/dnlib`. [online; 01-02-2018].

[2] ECMA International. *Standard ECMA-335 - Common Language Infrastructure (CLI)*. https://www.ecma-international.org/publications/files/ECMA-ST/ECMA-335.pdf. [online; 01-02-2018]. June 2012.

[3] *International Masaryk Conference for Ph.D. Students and Young Researchers*. http://www.masarykovakonference.cz/. [online; 01-02-2018].

[4] *jfrijters/ikdasm: Managed.Reflection based ildasm clone*. https://github.com/jfrijters/ikdasm. [online; 01-02-2018].

[5] S. Lidin. *.NET IL Assembler*. Apress, 2014.

[6] Microsoft. *PE Format*. https://msdn.microsoft.com/en-us/library/windows/desktop/ms680547(v=vs.85).aspx. [online; 01-02-2018].

[7] M. Milkovič. *Extraction of Information from .NET Executable Files*. Sborník příspěvků Mezinárodní Masarykovy konference pro doktorandy a mladé vědecké pracovníky 2017. Dec. 2017.

[8] *mono/mono: Mono open source ECMA CLI, C# and .NET implementation*. https://github.com/mono/mono. [online; 01-02-2018].

[9] *Retargetable Decompiler*. https://retdec.com/. [online; 01-02-2018].

[10] VirusTotal - Free Online Virus, Malware and URL Scanner. https://www.virustotal.com/en/. [online; 01-02-2018].