PE File Features in Detection of Packed Executables

Dhruwajita Devi and Sukumar Nandi

***Abstract*— Portable executable or PE file features play a key role in detection of packed executables. Packing performs a lot of changes to the internal structure of PE files in such a way that it makes it very difficult for any Reverse Engineering Technique, Anti-Virus (AV) scanner or similar kind of programs to figure out whether the executable is malware or benign. Therefore, it is very important to figure out whether a given executable is packed or non-packed before detecting it as malicious or benign. Once a binary is detected as packed, it can be unpacked and can be given to AV or similar kind of programs. In this paper we have included a brief description of Portable Executable file format as we need to know the internal structure of PE before figuring out Packed Portable Executables. We have considered the packed executable by UPX packer only, and hence mentioned the functioning of UPX packer very briefly. Our approach basically works in two phases. In the first phase, it extracts various features of portable executables and in the second phase it analyses the extracted features and comes up with best set of features, which can be used to identify whether a given binary is packed or not by UPX Packer. Experimental results are shown to the end of this paper. We figure out the key feature set with proper justifications to show differences between packed and non-packed executable by UPX packer.**

***Index Terms*—Malware, non-packed, packed, portable executable.**

I. Introduction

Pcking technique makes it easier for the writer of the malicious softwares to hide their malicious code from Anti-Virus or similar kind of a program. This is one of the most popular obfuscation techniques among all obfuscation techniques available, as in [1]. It is easier to collect packer softwares since several open source and commercial executables packers are available in the market. In a very simple way, we can define packing as an executable inside another executable. A Packer is basically a software which produces a number of data blocks that form the compressed and/or encrypted version of the original executable, as in [2]. A packer always inserts one unpacker stub inside the resultant executable itself to unpack the packed original executable at the time of runtime, as in [1]. The packing techniques vary from packer to packer. It is because different types of writer have different types of motivation for writing source code of his/her own packer.

Some of the packers uses more sophisticated technique to eavade detection. Multilayer-packing, Anti-unpacking are some of these techniques, as in [3]. Examples of such packers are Enigma, as in [3], Themida, as in [4] etc.

Manuscript received March 8, 2012; revised May 10, 2012.

Dhruwajita Devi, Sukumar Nandi, Indian Institute of Technology Guwahati, Assam India (e-mail: {dhruwajita.devi, sukumar} @ iitg.ernet.in).

Here, in this paper section II briefly describes the Portable Executable file format. Section III explains functioning of UPX packer in a nutshell. Section IV describes our approach followed by section V that includes the experimental part. Section VI finally concludes the paper which is followed by various references.

II. Portable Executable File Structure

Before going to experimental section, here we give a very brief description about the portable executable file structure. We know that the PE layout itself is a huge structure. But briefly it is given as follows.

This section starts out with familiar MS dos header followed by PE header. The PE Header itself contains three sections namely File header, Optional header and Section header, as in [5], [6], [7]. Code and Data sections are for holding the code of the program and initialized data. Import is for importing functions needed by programs at the time of run time. Some of the most common resources are Icons, Version information, GUI resources etc.

At a minimum, a PE file will have two sections, one for code and the other for data. The predefined and the most commonly present sections for an application of Windows NT are: Executable Code Section, named .text, Data Sections, named .data, .rdata, or .bss, Resources Section, named .rsrc, Export Data Section, named .edata, Import Data Section, named .idata, Debug Information Section, named .debug, as in [8]. Moreover, two more sections which are common in most of the times are .reloc for Relocation information and .tls section, which stands thread local storage. Windows supports this special storage class in which a data object is not a stack variable, but is still local to each individual thread that runs the code. Therefore, each thread can maintain a different value for a variable declared by using TLS.

All the above mentioned sections are called standard sections. If there are sections rather than the standard sections, are called nonstandard sections, as in [9]. Each and every section has its own header structure. The data items of the structures are the key feature for analyzing any executable as packed or non-packed.

III. Functioning of UPX Packer in a Nutshell

UPX is the most popular among all the packer softwares. Compression and/or Encryption techniques of all the packers vary based on the code of their writers. UPX packer packs all the sections present in the input binary into a single section called packed data. It also includes unpacker code along with the packed data forming one nonstandard section in the resulting binary called UPX1. UPX0 is one more section in the resulting output binary. It is empty and reserves an

address range. The address range is needed by the packed data when it gets unpacked by the unpacker code at the time of runtime. If the input binary possessed a .rsrc section, the resulting output binary will also have one .rsrc or resource section and if the input binary did not have a .rsrc section, the output binary also would not have it, as in [3].

IV. Our Approach

We can devide our approach into two phases basically. First phase is feature extraction and second is the analysis phase. Based on our extraction mechanism and observation, we come up with the best set of features with which we can definitely differentiate the executables Packed and non-Packed by UPX Packer. The pictorial representation of our approach is as follows :

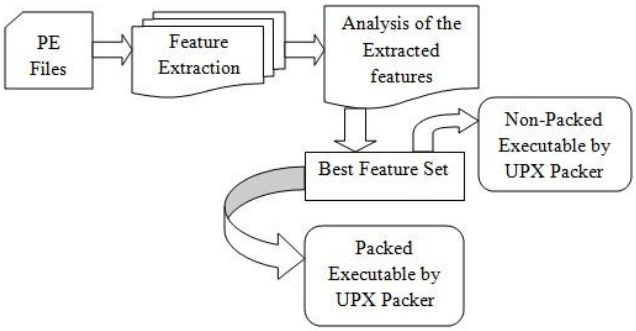

Fig. 1. Pictorial representation of our approach

We can visualize our approach by looking at the figure given above. It is basically comprised of two steps Feature Extraction and Analysis as mentioned.

V. Experiments

Initially we did manual extraction of features to have an idea about the features. We examine each of the portable executables we had after and before packing by UPX packer. For this purpose, we dump the files using Dumpbin Gui, as in [10], which is freely available. We collected UPX packer, as in [11] to pack the executable we collected.

We develop a C language program to extract features from the portable executable files. We collected 4095 executables files. Among them 2992 were malicious programs downloaded from http://offensivecomputing.net/. 1103 were benign executables collected from a newly installed windows machine and some other common software applications. We extract a lot many features from the executables before and after packed by UPX packer.

Windows is mostly written in C and C++. Therefore it is easier to extract the features of the portable executable files. We have extracted most of the features through our program. 16 features from DOS Header are extracted. PE header comprised of three parts, namely File header, optional header and Section header. We extracted 6, 29 features from file header and optional header respectively. Again, 10 features from each section in the section header. We also calculate the entropy of each and every file, after and before packing the same.

After a lot many observations and analysis, we come up with a feature set of four main features which can be used to figure out packed executables by UPX packer. These features are given in table.

TABLE I: List of Features

1	2	3	4
Entropy (ENTP)	Size Of Uninitialized Data (SOUID)	Size Of Headers (SOH)	Size of Raw Data (SORD)

The graphical representation of the difference between the executables packed and non-packed by UPX packer are also shown along with the justification of the features.

A. Entropy (ENTP)

Entropy can be considered as one of the major feature in classification of packed and non-packed executables. It is a measure of the inherent randomness in a probability distribution. Packing method conceals malicious executables' string, data and code. These methods transform some or all of the original bytes into a series of random-looking data bytes. That is why entropy of a packed executable is always higher than a non-packed executable.

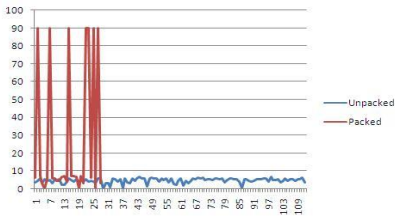

Fig. 2. Entropy

B. Size of Uninitialized Data (SOUID):

Compressed sections usually have the UNINITIALIZED DATA flag enabled. It is because of the null size on disk. The loader takes the compressed sections and unpacks them to their original memory locations at the time of execution.

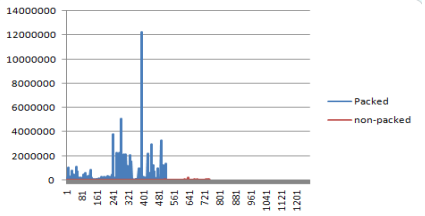

Fig. 3. Size of uninitialized data.

C. Size of Headers (SOH):

UPX packer wraps the whole exe into the packed data along with the unpacker code in UPX1. We know that the size of the header contains the size of the PE Header and the section table. That is why the size of header of the resultant PE after packed by UPX is generally greater than or sometimes equal to the size of exe not packed by UPX.

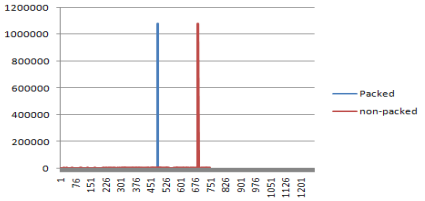

Fig. 4. Size of headers.

D. Size of Raw Data (SORD):

UPX packer changes the RAWSIZE of each packed section to 0. The size in memory remains unchanged, because the program still has to execute normally and be unpacked at its original location. If the RAWSIZE is null, it means the section is non-existent on disk.

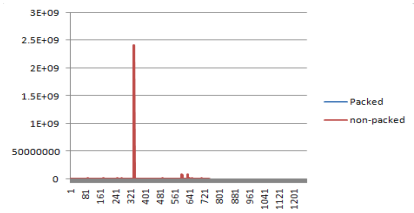

Fig. 5. Size of raw data (SORD).

Different packers have different key features. The features vary packer to packer as it depends on the implementation and the platform it is running on.

VI. Conclusion

In this paper, we present four features of portable executable which are key feature to differentiate executables packed and non-packed by UPX packer. It is always tedious to figure out malicious or benign executable once a PE is got packed. Therefore, our approach makes it easier to figure out whether an executables is packed or not by UPX just by extracting the feature set comprised of these four. Once an executable is detected as packed, we can unpack using universal unpacker for e.g. PolyUnpack. As soon as we unpack it, we can give it to antivirus or equivalent softwares to detect whether the file is malicious or benign. Hence, we can conclude that it is making life easier for traditional signature-based softwares to detect malicious executables.

Acknowledgment

The authors would like to thank Mr. Neminath Hubballi for his contribution during discussion related to this work.

References

[1] R. Lyda and J. Hamrock, "Using Entropy Analysis to Find Encrypted and Packed Malware," *IEEE Security and Privacy*, March/April 2007.
[2] M. Howard, "Revealing Packed malware," *IEEE Security and Privacy*, September/October 2008.
[3] F. Guo, P. Ferrie, and T. Chiueh, "A Study of the Packer Problem and Its Solutions," *RAID 2008, LNCS 5230*, pp. 98–115.
[4] L. Sun, S. Versteeg, S. Boztas, and T. Yann, "Pattern Recognition Techniques for the Classification of Malware Packers," *ACISP 2010, LNCS 6168*, pp. 370–390.
[5] M. Pietrek, *Peering Inside the PE: A Tour of the Win32 Portable Executable File Format*, 25th of Nov 2010.
[6] G. Erdelyi, Reverse Engineering III: PE Format.
[7] Loading a DLL from memory. [Online]. Available: http://www.joachim-bauch.de/tutorials/loading-a-dll-from-memory/
[8] Goppit, *Portable Executable File Format – A Reverse Engineer View*, 2006.
[9] R. Perdisci, A. Lanzi, and W. Lee, "Classification of Packed Executables for Accurate Computer Virus Detection," *Elsevier*, vol. 25 June 2008.
[10] DumpbinGUI. [Online]. Available: http://www.cheztabor.com/dumpbinGUI/
[11] Softpedia. [Online]. Available: http://www.softpedia.com/dyn-postdownload.php?p=90710andt=4andi=1