Malware Obfuscation through Evolutionary Packers

Marco Gaudesi
Politecnico di Torino
marco.gaudesi@polito.it

Andrea Marcelli
Politecnico di Torino
andrea.marcelli@studenti.polito.it

Ernesto Sanchez
Politecnico di Torino
ernesto.sanchez@polito.it

Giovanni Squillero
Politecnico di Torino
giovanni.squillero@polito.it

Alberto Tonda
INRA UMR 782, MALICES
alberto.tonda@grignon.inra.fr

ABSTRACT

A *malicious botnet* is a collection of compromised hosts coordinated by an external entity. The malicious software, or *malware*, that infect the systems are its basic units and they are responsible for its global behavior. Anti Virus software and Intrusion Detection Systems detect botnets by analyzing network and files, looking for signature and known behavioral patterns. Thus, the malware hiding capability is a crucial aspect. This paper describes a new obfuscation mechanism based on evolutionary algorithms: an evolutionary core is embedded in the malware to generate a different, optimized hiding strategy for every single infection. Such always-changing, hard-to-detect malware can be used by security industries to stress the analysis methodologies and to test the ability to react to malware mutations. This research is the first step in a more ambitious research project, where a whole botnet, composed of different malware and Anti Virus software, is analyzed as a prey-predator ecosystem.

Categories and Subject Descriptors

K.6.5 [**Management of Computing and Information Systems**]: Security and Protection—*invasive software*; D.3.4 [**Programming Languages**]: Processors—*code generation, optimization*

Keywords

Malware, Evolutionary Packer, Computational Intelligence

1. INTRODUCTION

The analysis of the malware in the last 20 years shows a clear growth in the complexity of the hiding mechanisms. Nowadays, in order to evade signature-based detection, several malware authors exploit *packers* to change the binary file without affecting its semantics [2]. A packer is usually a standalone software that encodes and compresses an executable program, common encoding techniques include *Caesar chiper*, *XOR* and *Base64*. Initially, packers were used to mitigate reverse engineering and protect intellectual property (e.g. ASPack[1], Molebox[2] and Themida[3]), but eventually they have been introduced into the world of malicious software [4].

While the underlying idea is general and it may be applied to different scenarios and operating systems, including mobile OS, this abstract tackles packers for the *portable executable* (PE) used in Microsoft Windows operating systems since Windows NT 3.1. The PE format essentially defines the data structure containing the information necessary to manage executable code [1].

Usually, packers perform complex operations to modify a PE structure, including section reordering and header alterations. The *entry point* is set to an *unpacking stub* that, at run time, restores the original data in memory. In some cases the sensitive code is partially unpacked in a buffer, so that only a small portion of the protected code is exposed to the analysis at any given time. The stub routine is also responsible of resolving the import table of the original executable and of relocating memory address according to the *base relocation address*. Finally, it returns the control to the original entry point of the program [3].

Anti Virus (AV) software analyzes portions of code inside a PE both *statically* and *dynamically*: in the static analysis, file sections are checked against a database of *signatures* of known malicious software; in the dynamic analysis, the program operations are tracked while a heuristic mechanism tries to recognize behavioral patterns typical of malware.

The hiding mechanism proposed in this paper is based on evolutionary computation and embeds an evolutionary core directly in the malware. New candidate packers will be generated through genetic operators and checked internally to assess their efficacy. Such approach differentiates from the well-known *Polimorphic* and *Metamorphic* malware [5]: the set of possible obfuscation schemes are so vast that cannot be precomputed; moreover, the evolutionary core could be able to learn and to be trained.

2. EVOLUTIONARY PACKER

In the proposed approach, it is the malware responsibility to create a new hiding mechanism strong enough to ensure the survival of future generations. The core for creating the encoding routine is a Turing-complete evolutionary algorithm able to generate completely new algorithms.

[1]http://www.aspack.com/
[2]http://www.molebox.com/
[3]http://www.oreans.com/themida.php

GECCO '15 July 11-15, 2015, Madrid, Spain

ACM ISBN 978-1-4503-3488-4/15/07.

DOI: http://dx.doi.org/10.1145/2739482.2764940

In more details, both the encoding and the decoding functions are randomly-generated, variable-length sequence of x86 assembler instructions. Instructions are directly handled as binary opcodes, so there is no need of a compilation and linking phases. The generation process requires to find reversible assembly instructions (e.g., *INC*, *ROR*, *BSWAP*, *XCHG*) and small blocks of code that have a complementary one. Since even few bytes may represent a signature, it is also necessary to partially shuffle the instructions, although this has the drawback of potentially disrupting the encoding/decoding routines. It should be noted that, while the encoding and the decoding functions are created in parallel, only the latter is included in the generated malware.

In order to efficiently evaluate a candidate packer, the encoding and the decoding routines are applied subsequently to randomly generated sequence of bytes: if the final result is different from the original sequence, the candidate is simply discarded. Then, the packer is used to obfuscate the malware and the *Jaccard Similarity* is evaluated to assess candidate fitness values. Aiming to achieve *invisibility through diversity*, the process is iterated for a given number of generations, or until the Jaccard Index distribution is lower than an experimentally-defined threshold. Figure 2 shows an example of the Jaccard distribution of a malware variant that maximise the dissimilarity from the original code.

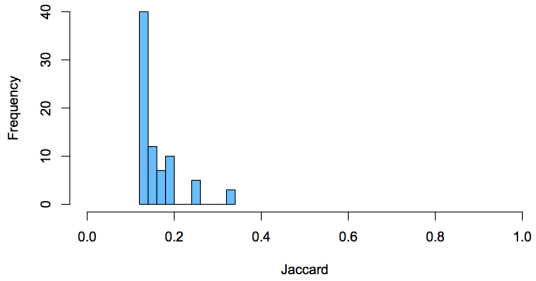

Figure 1: Histogram of the Jacquard Index distribution of a Malware Sample.

As the decoding routine is embedded in the PE, once the new malware is executed, it will restore each part of the program in memory ready for execution. Finally, the same code generation engine is used at run-time to mitigate heuristic-based research on malicious pattern.

3. EXPERIMENTAL EVALUATION

Preliminary experiments have been performed on a Windows based Operating System, with an Intel x86 architecture. The choice is dictated by the huge availability of AV software for the platform.

As a testbed, two well-known online malware scanner have been used: they make use of tens of different AV products, allowing a direct and simple comparison of the results. Moreover, in order to confirm the outcome precision, the five most effective AV software have been installed on different hosts.

The executable program to be tested executes a well-known TCP bind shellcode. The high initial detection rate and an executable behavior susceptible to heuristic evaluation, made this program well suited for the experiment. Table 1 summarizes the number of AV programs that detected the threat over the total. The item subjected to evaluation is an automatically generated malware variant in three different stages of the evolution of the packing mechanism. Other experiments show comparable distributions.

Table 1: Detection rate of the evolutionary variants of a packed malware.

	Uncoded	Evo 1	Evo 2	Evo 3
Virus Total	35/57	2/57	2/57	1/57
Metascan Online	25/44	4/44	3/44	1/44

More than half of the AV programs do detect the *uncoded* version of the executable as a threat. However, although *Evo 1* uses a quite simple encrypting technique, the detection rate is unbelievably low: most of the AV software does not employ a proper heuristic engine and packing an executable file makes the static signature-based search to fail, as it requires a specific signature for each packing routine. *Evo 2* implements a more sophisticated encoding mechanism with shuffled instructions; finally *Evo 3* represents a fairly complex version of packed malware that makes use of several computational intensive operations that aim to confuse heuristic engines. The result is quite scary: among all the tested scanning products, experiments denote the effectiveness of only one heuristic engine.

4. CONCLUSIONS

A malware program able to evolve its own packer, creating a brand new encoding routine in each infection, may still represent a challenge for the security community. Experimental results demonstrated that even though the static signature-based analysis is the most effective one in detecting threats, it could not be efficient with such an evolutionary mechanism. We strongly believe that the future of the malware research must focus on the identification of behavioral patterns.

While this research is far from being concluded, future work would also need to include anti-debug, anti-disassembly and other techniques that until now have only been used in few static methods [6].

5. REFERENCES

[1] Goppit. Portable executable file format - a reverse engineer view. *CodeBreakers Magazine*, Jan 2006.

[2] F. Guo, P. Ferrie, and T. Chiueh. A study of the packer problem and its solutions. pages 98–115, 2008.

[3] S. Michael and H. Andrew. *Practical Malware Analysis - The HandsOn Guide to Dissecting Malicious Software*. No Starch Press, 2012.

[4] K. A. Roundy and B. P. Miller. Binary-code obfuscations in prevalent packer tools. *ACM Computing Surveys (CSUR)*, 46(1):4, 2013.

[5] P. Ször and P. Ferrie. Hunting for metamorphic. In *Virus Bulletin Conference*, 2001.

[6] M. V. Yason. The art of unpacking. *BlackHat*, Feb 2007.