INTERNATIONAL JOURNAL OF COMPUTERS COMMUNICATIONS & CONTROL
ISSN 1841-9836, 12(4), 461-474, August 2017.

Obfuscation-based Malware Update: A comparison of Manual and Automated Methods

C. Barría, D. Cordero, C. Cubillos, M. Palma, D. Cabrera

Cristian Barría, Claudio Cubillos*
Pontificia Universidad Católica de Valparaíso
Valparaíso, Chile
cristian.barria@udp.cl
*Corresponding author:claudio.cubillos@pucv.cl

David Cordero
Universidad Andrés Bello
Santiago, Chile
d.cordero.v@gmail.com

Miguel Palma
Universidad Tecnológica de Chile
Santiago, Chile
rmiguel.palma06@inacapmail.cl

Daniel Cabrera
Universidad de Valparaíso
Valparaíso, Chile
daniel.cabrera@uv.cl

Abstract: This research presents a proposal of malware classification and its update based on capacity and obfuscation. This article is an extension of [4][a], and describes the procedure for malware updating, that is, to take obsolete malware that is already detectable by antiviruses, update it through obfuscation techniques and thus making it undetectable again. As the updating of malware is generally performed manually, an automatic solution is presented together with a comparison from the standpoint of cost and processing time. The automated method proved to be more reliable, fast and less intensive in the use of resources, specially in terms of antivirus analysis and malware functionality checking times.
Keywords: Security, Malware, obfuscation techniques, cyberspace, antivirus.

[a]Reprinted (partial) and extended, with permission based on License Number 3954200696590 [2016] © IEEE, from "Computers Communications and Control (ICCCC), 2016 6th International Conference on".

1 Introduction

Malware is a malicious program that provides access to a system without user's authorization and executes undesirable actions. The terms malware and virus are usually used as the same concept, although they are, in fact, different [4] [19]. Initially, the use of malware was strictly associated with the investigation and protection of software developers for intellectual property (using cryptography), but throughout time its objective changed, and therefore, its importance. Currently, the aim of malware is mainly related to profit- making [8] [15].

The evolution of malware began in 1949 when Von Neumann established the idea of stored program and introduced the Automata Theory, which is the possibility of developing small replicates of a program that were capable of taking control of other programs with similar structures [13]. While the concept has many applications in science, it is quite easy to apply it to

viruses, considering that, at that time, the two concepts (i.e., virus and malware) were thought as equivalent.

In 1971, Bob Thomas created the first malicious code, named Creeper, that was capable of infecting IBM 360 machines on the ARPANET network by delivering an on-screen message that said, "I'm the Creeper, catch me if you can!" To find and eliminate the Creeper another code, called Reaper, was created. From this, it originated the current antivirus [3].

Primarily, malware affects the Critical Information Infrastructure (CII), which is of prime importance to both, the attackers and the defenders. It is important to highlight that a successful virus leaves the CII defenseless. This is why for both, the defenders and the offenders, there is a necessity of implementing a Security Information Management System (SIMI) with the objective of preserving the reliability, integrity and availability of the information in order to act upon these threats [18].

In the context of this research, obfuscation will be defined as the execution of transformations in a malware code (source or binary) that changes its appearance through the realization of a series of steps. This maintains the malware functionality and allows it to take control of the systems [6]. It is fundamental to continue carrying out research regarding obfuscation, in order to contribute to its analysis from a malware updating perspective. As the malware updating process is generally performed manually, an automatic solution is to be introduced so to compare both in terms of time and costs.

This research, an extended version of a previous work [4], provides a proposal of malware classification based on obfuscation. The novelty of the present work relies on; first, providing a summary of the obfuscation-based malware classification tackled in the previous work [4]; second, formalizing the malware update process and making an empirical demonstration on how a malware, that is detectable by antiviruses, could be reused by updating it and making it undetectable through obfuscation; third, comparing the manual and the automated methods for malware updating though an example.

In section 2, this paper presents a summary of malware classification, while in section 3 the use of obfuscation through a crypter for updating the detected malware is explained. In section 4, a chart of the malware updating and its classification is proposed. In section 5, the proposed malware updating process is formalized and explained through an example. In section 6, a software prototype of an automated version of the obfuscation process is presented. Section 7, provides an evaluation and comparison of the manual and automated solutions. Finally, in section 8, the conclusions of this research are presented and future work.

2 Related work

In this section, a brief overview of the different malware classification is provided. As it is known, there are different malware classifications stemming from different perspectives. For example, classifications regarding malware infection mechanisms, and others regarding means of malware spread. Although these are used in different investigations, there is no given classification from the perspective of obfuscation, as a common vision has not been agreed yet [17].

As it is shown in Figure 1, a structure that facilitates the malware classification (species) could be established according to the class, type and generation. This classification will enable to succinctly present a part of the essential information about the malware non-detectable capacities.

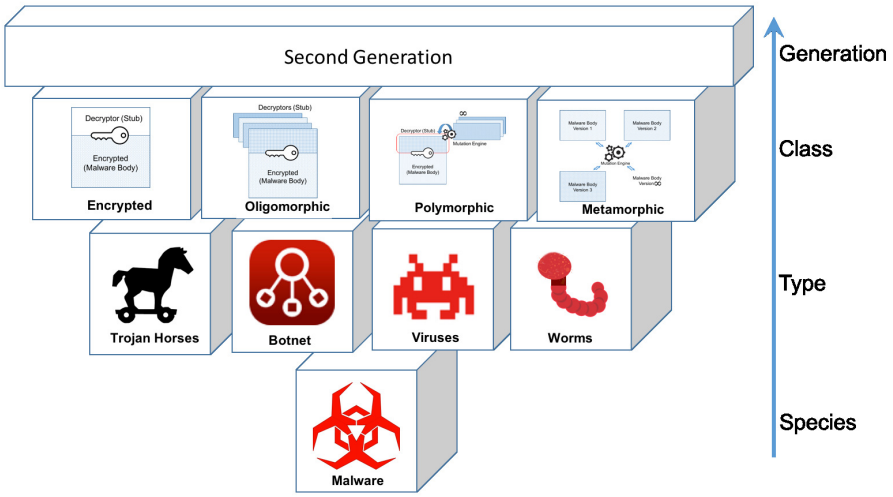

Figure 1: Malware classification

3 Malware updating based on obfuscation

According to the malware classification provided in previous investigations, first-generation malware were not considered part of the inverted pyramid shown in Figure 1, as they did not present any modifications in their code and they are commonly denominated No-Stealth [12] because of their obsolescence.

Although No-Stealth malware are detected by an antivirus, currently they are capable of being transformed into second-generation malware using crypters. Crypters are easy to obtain from specialized information security websites, and these could apply obfuscation techniques to any type of file. However, in this research, crypters will be applied to malware, yet not alter their functionality in any of their states [6] [9] [20].

3.1 Update

Malware updating requires obfuscation to give rise a second generation. This can be performed by using one of the following two methods: a) The manual method. In this, the operation is performed by a person and is applied to the Encryptor and Oligomorphic malware types. b) The automated method. This refers to a malware that can automatically undergo the process of obfuscation without requiring a human operator and is applied to Polymorphic and Metamorphic malware [14], as it is shown in Table 1.

Table 1: Obfuscation update

MALWARE	Manual	Automated
Encryption	X	
Oligomorphic	X	
Polymorphic		X
Metamorphic		X

3.2 Procedures

Among the procedures that are highlighted in various sources of information related to computer security, the following can be identified: AvFucker, DSplit, RIT, Hexing and XOR. These aim at generating a binary of the original code, producing a modification of a relevant hexadecimal value to evade the antivirus. Figure 2 allows us to perform an updating to the malware through the obfuscation process, which will depend on malware status, method, technique and procedure employed.

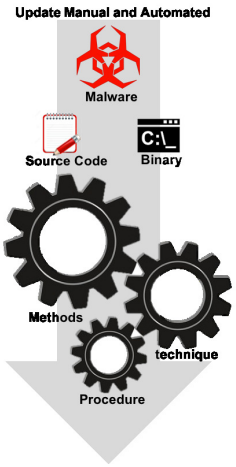

Figure 2: Malware obfuscation update.

In the case of an antivirus detection, a decryptor or stub is resorted to apply the obfuscation method, in order to re-apply the crypter to malware so to achieve its renovation and, thereby its ability of evasion. The report provided by the antivirus detection is directly given to the stub, if this gets detected it directly compromises the encrypted malware. This is put into practice through the following tests:

- Malware construction (No-Stealth)
  The construction of a malicious code has the objective of obtaining documents from a third party, including certain directories and various extensions from a computer. A part of the code is presented (for confidentiality reasons) in Figure 3.

```
user = getpass.getuser()
a=  r'C:/Users/'
b= user+'/'
c= r"Desktop"
d= r"Documents"
e= r"Downloads"
ficheros=os.listdir(a+b+c)
mydocumentos=os.listdir(a+b+d)
download=os.listdir(a+b+e)
included_extenstions = ['docx','doc','pdf']
file_names = [fn for fn in ficheros if any([fn.endswith(ext) for ext in included_extenstions])]
```

Figure 3: No-stealth malware code portion

- Antivirus tests
  The analysis of a malicious code is performed by a specific antivirus (NOD 32) signature, which has the ability to incorporate other antivirus engines. As it is shown in Figure 4, this code is detected and reported as an element of risk.

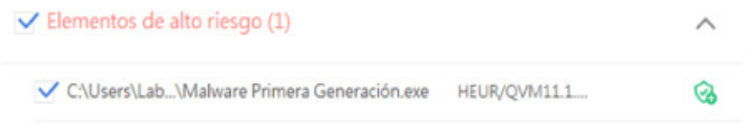

Figure 4: Antivirus test

- Obfuscation application
  After the malicious code has been detected, obfuscation techniques are applied in order to subject the malware to a crypter (encryption / decryption), as shown in Figure 5. Once the process has been completed, a detectable code may become undetectable.

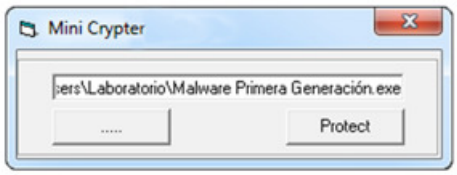

Figure 5: Crypter application

- Screening
  Once the malicious code has undergone obfuscation, the malware is subjected to the antivirus (NOD 32) again, under the same conditions described above (same house and built motors). As it is shown in Figure 6, the antivirus always generates a reporting message in case the malware is either detected or not. In case the malware goes undetected, the antivirus logs does not consider this file a threat, as shown in Figure 7, and therefore, the malware may fulfill its malicious intent.

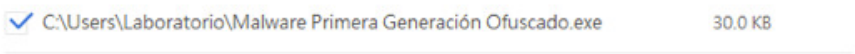

Figure 6: Antivirus test.

  As shown in Figure 8, it is evident that a first-generation malware could be subjected to obfuscation techniques. In order to do so, a crypter tool is applied to transform this first-generation malware into a second-generation, and thereby, rising its hierarchical criteria.

4 Obfuscation process and malware classification proposal

In accordance with the malware classification that was previously described, a first-generation malware is not considered because it has not undergone the obfuscation techniques yet. However, it is possible to prove that despite the obsolescence of this No-Stealth malware, this is still present in the cyberspace and need to be considered. Furthermore, this No-Sealth malware could be transformed into a threat only by applying an obfuscation procedure, resulting in this becoming undetectable by an antivirus. Taking into account the points previously exposed, a flow chart is proposed for the updating of malware, as shown in Figure 9.

Figure 7: Antivirus logs

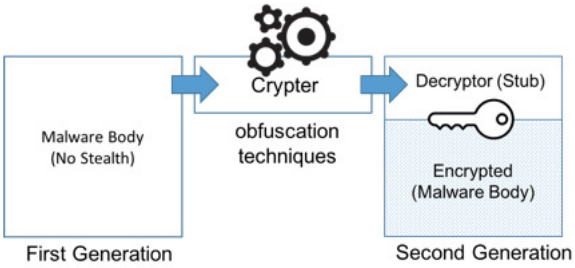

Figure 8: Malware transformation

Thus, in the first inverted pyramid previously described the No Stealth malware should be taken into consideration because this could be related to any other malware either belonging to a class (virus, worms, botnet, and Trojan horses), or belonging to the first generation. As it is shown in Figure 10, a revised inverted pyramid is proposed in order to include a complete structure that classifies malware.

5 Current malware updating process

The process of detecting malware through the recognition of their signatures, that these are later saved into the antivirus databases. However, through an updating process, these malware could be transformed and become undetectable without losing its functionality as it is described by Barría et al [4].

When considering the diagram in Figure 11 [11], it could be observed that the updating process of a malware starts in its binary state, and then a crypter is required in order to apply the method of encryption. After this, the crypter output must undergo the antivirus detection (signature-based analysis). In case the encrypted malware goes undetected, it means the updating process has been successful. In the specific case of this work, the obfuscation process employs the Dead-Code Insertion technique [1]. For all aforementioned, the modders defines this as the AvFucker [11].

This process involves a series of procedures in which the malware is encrypted by a Crypter

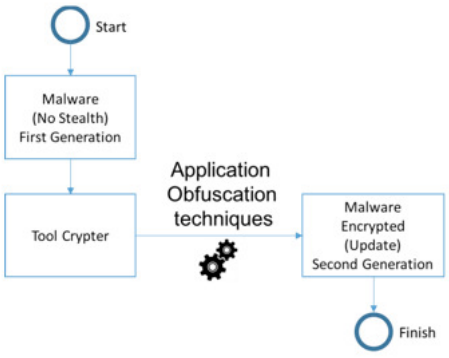

Figure 9: Proposed obfuscation procedure.

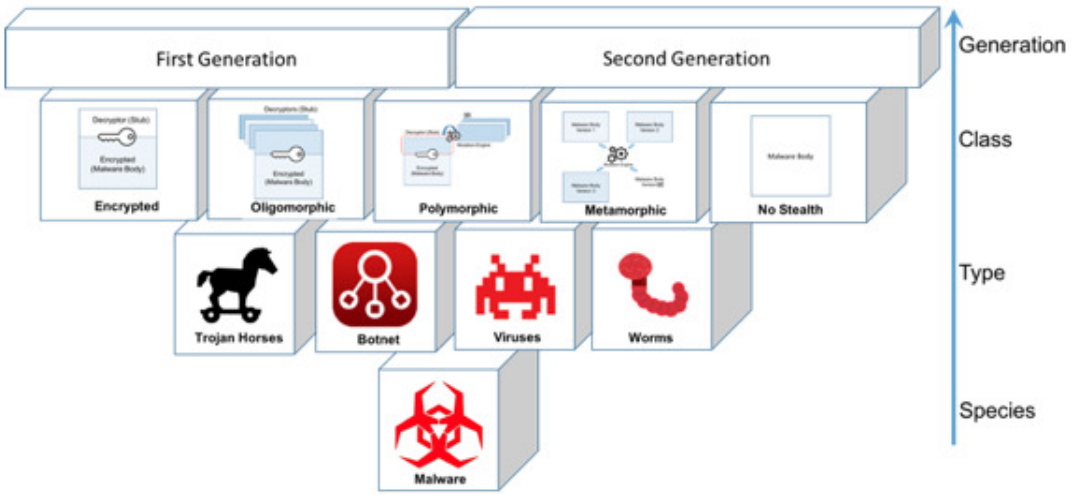

Figure 10: Proposed malware classification

and is then reviewed by one or more anti-malware systems. If it is detected, an insertion of a dead code, based on the process of obfuscation, followed by a re-encryption of the resulting code with a Crypter is performed.

Finally, the test is performed against the anti-malware or systems, and the cycle is repeated until a program is obtained with a signature that is no longer detected by these systems. A graphical explanation of this process is illustrated below in Figure 11 [10]. When the crypter output is contrasted through detection techniques based on signatures [2] and is detected as malware, the procedure is applied as a whole. This means that the stub, the key and the enciphered malware are included, although only the stub area is detected. This procedure consists of the malware copy creation (crypter output), whose quantity is given by the file size, according to the following equation:

$$\mathrm{Nc} = (\mathrm{St} + \mathrm{K} + \mathrm{Mc})$$

where:
Nc= Copy number.
St= Stub bytes size.
K = Key bytes size.
Mc= Enciphered malware bytes size.

Each one of these copies, is created for the Dead Code Insertion (00, 90), from byte to byte, from the initial to the final offset, correlatively [7], as observed in Figure 12.

Once all copies are made, they are analyzed by the antivirus. In this way, only files detected as non-malicious are obtained, and the rest are removed when matching with the antivirus registered signature. A group of files is then obtained that is able to avoid the antivirus, but its hexadecimal code has been modified, so it is necessary to verify that the functionality is not affected.

What was previously presented is not viable if a file of 1 Mb output is used as an example:

$Nc = (St + K + Mc) = 1.048.576\ copies$
$Nc^2 = Space\ for\ the\ copies\ on\ the\ hard\ disk$

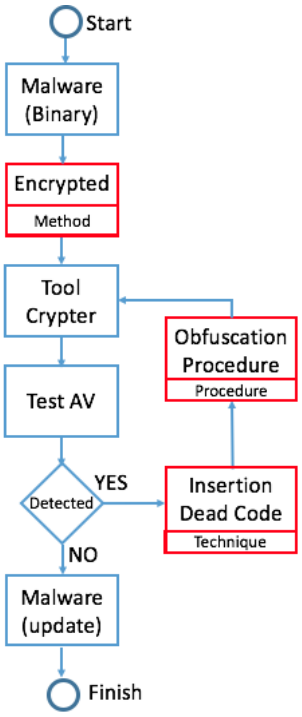

Figure 11: Diagram of the malware update process.

$$Nc^2 = (1.048.576)^2\ bytes\ =\ 1\ Terabyte$$

The previous analysis implies the following problems: a) capacity, since the number of files created could be higher than the number of information terabytes directly affecting the hardware capacity, which also affects equipment investment; b) test, as a function of the time in which the antivirus performs the analysis on each of the copies; c) functionality, a revision that must be made on an individual basis to the group of undetected files by the antivirus and then prove that the malware functionality is not affected.

This is how the procedure seeks to be optimized by increasing the Dead Code Insertion. If the 1 Mb output file is used, it is possible to create copies every 1000 bytes (offset with 0.1 percent of the total file size), and the proper analysis is made, considering only the portion of 1000 bytes that is undetected. Copies are then created every 100 bytes over it, and this procedure is repeated until reaching the portion of 1 byte [5], considering that the ranges are not necessarily consecutive. Thus, hardware use increases, and the number of created files decreases, so the antivirus testing time also decreases.

Nevertheless, the problem of the malware operation test persists because is needed a controlled environment that allows typical functions of this type of malicious code to be proved, such as persistence, connectivity and spreading. This takes significant time and a great amount of resources for the modder [11]. Thus, a tool called an "annotator" is used [16], which permits the functions described above to be verified by replacing input malware on the crypter. In addition, another characteristic is its tiny size, further allowing optimization of the creation of copies and reducing the antivirus analysis and detection time.

Currently, the iteration of this model is carried out with the support of various tools, including crypters, hex editors, and anti-malware engine systems, among others. It also usually uses a tool to generate a number n copies of the original file, modifying each copy in a different range of offsets and replacing those locations with a dead code. While the use of such tools represents

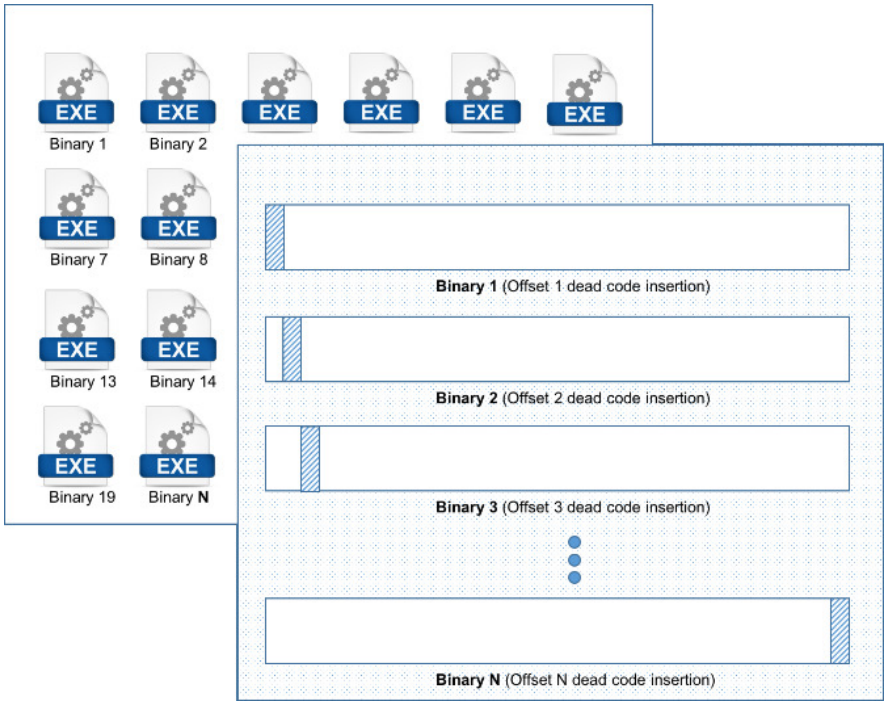

Figure 12: Malware copy generation

a significant reduction in the time required compared to the performance of the same process manually (e.g., editing each range of offsets using an editor), this part of the process adds more time to the total iteration, increasing the final cost of the process. Having identified this difficulty, the ability to automate this iteration emerges as a natural choice in regard to reducing time and monetary costs.

6 Software prototype validation

An algorithm was programmed in Python version 2, with the addition of PyWinAuto library to automate the process under study, exposed in Figure 13.

The automatic process and its algorithm provides the advantage of reducing the execution time of each task, while leaving the structure of the process itself intact, thus allowing an objective performance comparison between the original manual process and the automated one.

Avoiding human factors on the execution of each task reduces the risk of certain kind of errors, mostly related to distraction, fatigue, among others. This can be more critical in certain tasks, compared to others.

For example, selecting the malware sample and using the Crypter tool on it (the first task), can be considered a fairly simple and straightforward task. But replicating the file an undetermined amount of times, with different ranges, can eventually involve the repetition of an action hundreds of times, and each one requiring the exact selection of specific ranges, and offsets which is error prone. In the automated method, the algorithm itself will not suffer from fatigue after 5 hours of uninterrupted work, for example, while humans surely will, being a more reliable solution.

The general process involves five phases: malware encryption, offsets identification, obfuscation through dead code insertion, anti-malware analysis, and functionality analysis. The encryption phase, as it name sais, involves the encryption of the malware code through the use of

a crypter software. The offset identification regards the activity of splitting the file for making an exhaustive search looking for the point (offset) in which changing that bit provides a change in the overall malware signature recognized by anti-malware software, thus turning it in undetectable. The anti-malware analysis refers to antivirus check to be performed to the file sample with the modified offset in order to check if it is detected with the virus. Please consider that this step does not involve only a single antivirus software but a hole set of them, four or five different at least in order to ensure undetectability. The phase of functionality analysis, comes after the anti malware analysis for each of the files with their diverse offsets and pursues to check that the dead code insertion has not altered or braked the functionality to the malware within the file. The algorithm automatically goes through each iteration, changes the values for each range and advances until the end of the whole process. The only human interaction required is starting the script, since the automated process replaces every task originally executed by human resources.

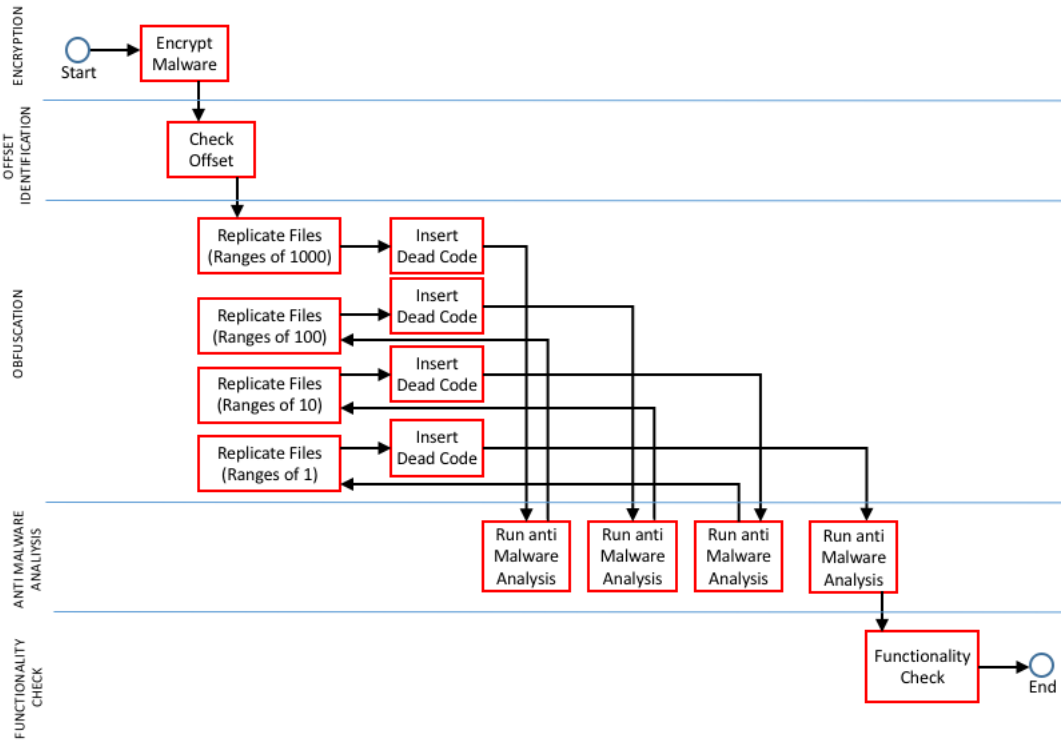

Figure 13: Malware update automated process

7 Experimentation

Once the prototype was prepared, we compared the time required to complete an instance of the process, from beginning to end, between the automated case and the original case then, it was used simulation software, to evaluate the time required in 50 instances.

To standardize the experimental conditions in both cases, the same tools and base file containing the signature of the malware were used, as detailed in the following list:

- Base File
- Crypter tool
- Hex editing tool and File Replication "Offset Locator"

- Anti-malware Engine "NOD32"

Both processes were executed on the same computer with the same operating system (Windows 7). Finally, the economic dimension of each process was evaluated, and each hour of work was assigned the value of CLP 1100.

7.1 Experimental evaluation

The manual process execution required 420 minutes (7 hours), while the prototype software process was completed in 5 minutes. In both cases, it was possible to obtain a series of files that successfully evaded the anti-malware detection engine.

Considering the value assigned to each hour of work, we observe the following, in Table 2.

Table 2: Time and cost comparative table evaluated experimentally.

	Manual Process	Automated Process
Time (hours)	7	0,083
Cost (CLP)	7.700	1.100

7.2 Simulated evaluation

Fifty process execution cycles in the simulation software generated the following results, in Table 3:

Table 3: Time and cost comparative table evaluated by simulation

	Manual Process	Automated Process
Time (hours)	350	4,2
Cost (CLP)	385.000	5.500

Due to the cost being calculated per hour, the automated process value was still considered as a full worked hour, regardless of having completed the entire set of tasks in much less time than that.

It is also important to note, that the human executed tasks may differ in terms of time needed, because of the possibility of mistakes during the execution of any action, thus causing an increase of the time needed and, in consequence, an increase in its cost. During this experimental evaluation, however, there were neither difficulties nor failures during the execution of the entire process, so the values provided are as close as possible to an ideal execution.

Although these values already provide clear evidence of the cost reductions obtained thanks to this automation, this difference may be even higher, because the human resources simulated on the manual process were considered with ideal conditions, such as 100 percent correct execution of tasks (no possibility of mistakes) and continuous, uninterrupted work hours during the whole process. As can be expected, actual human resources will not present such performance, meaning a higher cost for the manual process.

7.3 Cost and time comparison

Figure 14 and Figure 15 present the same data, but from a different perspective, which helps to easily perceive the cost reduction obtained with this automation.

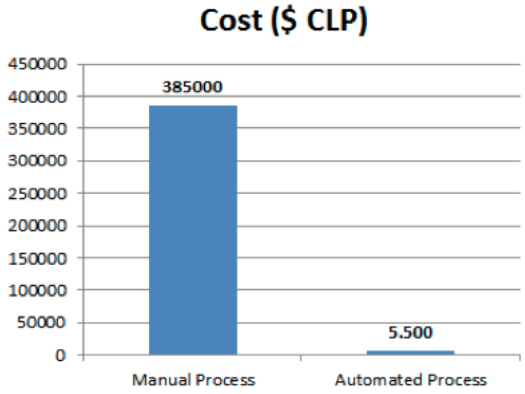

Figure 14: Cost comparison between manual process and automated process under simulated evaluation

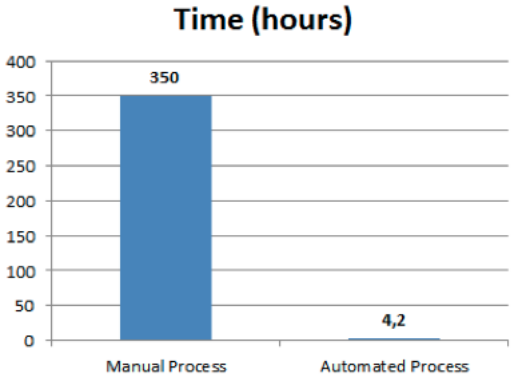

Figure 15: Time comparison between manual process and automated process under simulated evaluation

The numbers speak by themselves, as can be easily noticed that the same result can be achieved with way less time and money, thus exposing the advantages of automating the process using the algorithm. As mentioned before, ths structure of the whole process was left untouched, meaning that this comparison is not affected by neither adding, removing nor modifying tasks. Each task remains the same, but its execution is improved via automation.

8 Conclusions and future work

Previous researches have provided a malware classifications that has been generated from multiple perspectives; however, there is no classification generated on the basis of the obfuscation perspective. Consequently, the first contribution of the present work is to provide a classification that takes into account the obfuscation procedure. This revised classification allows us to hierarchically structure the malware based on their capacities. Furthermore, considering that these malware are still present in the cyberspace paves the way to create countermeasures to preserve the security of the information systems.

Because of the rapid evolution of second-generation malware, those that are considered to be part of the first-generation have been described as obsolete, and therefore unimportant. However, this research has proved that through the application of methods, techniques, procedures and tools of obfuscation it is possible that No-Sealth malware could be updated to the point of being

transformed into second-generation malware, and therefore, go undetectable by the antivirus systems. Similarly, the present work incorporates this updating procedure through a flow chart that considers this classification proposal of the revised inverted pyramid previously presented.

This work also explains that this updating procedure could be put into practice from a manual method to an automated method. It is evident that in the case of the automated method, the resources of time and costs are more efficiently employed. This represents a relevant challenge for the malware developers, who day-to-day have to deal with between those who protect and those who evade the system of data protection.

Future work may consider the improvement of this malware updating process in order to employ new techniques with the aim of searching matched patterns and making this process more efficient.

Acknowledgment

The present work has been partially supported by Pontifícia Universidad Católica de Valparaíso through scholarship INF-PUCV-2016.

Bibliography

[1] Balakrishnan A., Schulze C. (2005); Code obfuscation literature surveyt, *CS701 Construction of compilers*, vol. 19, 2005.

[2] Bazrafshan Z., Hashemi H., Fard S. M. H., Hamzeh A. (2013), Survey on heuristic malware detection techniquest, *Information and Knowledge Technology (IKT), 2013 5th Conference on*, 113-120, 2013. doi: 10.1109/IKT.2013.6620049

[3] Balakrishnan A., Schulze C. (2010), Code obfuscation literature survey, *CS701 Construction of Compilers*, URL http://pages.cs.wisc.edu/ arinib/writeup.pdf, 19, 1-10, 2005.

[4] Barria C., Cordero D., Cubillos C., Palma M. (2016), Proposed classification of malware, based on obfuscation, *2016 6th International Conference on Computers Communications and Control* (ICCCC),IEEE Xplore 2016, ISBN: 978-1-5090-1735-5, 37-44, 2016. http://dx.doi.org/10.1109/ICCCC.2016.7496735

[5] Barria C., Cordero D., Cubillos C., Osses R, Obfuscation procedure based in dead code insertion into cryptert, *2016 6th International Conference on Computers Communications and Control*, IEEE Xplore 2016, ISBN: 978-1-5090-1735-5, 23 - 29, 2016. http://dx.doi.org/10.1109/ICCCC.2016.7496733

[6] Egele M., Scholte T., Kirda E., Kruegel C(2008), A Survey on Automated Dynamic Malware-analysis Techniques and Tools, *ACM Comput.Surv.*, 44(2), 1-6, 2008. http://dx.doi.org/10.1145/2089125.2089126

[7] Khurram M., Syed Noor-ul-Hassan S., Zikria Y. B., Nassar I.(2010), Evading Virus Detection Using Code Obfuscation, *Future Generation Information Technology: Second International Conference, FGIT 2010*, 394-401, 2010. http://dx.doi.org/10.1007/978-3-642-17569-5-39

[8] Konstantinou E., Wolthusen S. (2008), *Metamorphic virus: Analysis and detection*, Technical report, Royal Holloway University of London, vol. 15, 2008.

[9] Kolter J., Maloof M. (2006), Learning to Detect and Classify Malicious Executables in the Wild,*Journal of Machine Learning Research*, 7(7), 2721-2744, 2006.

[10] Kumar A., Shrivastava V. (2013), BASIC: Brief Analytical Survey on Metamorphic Code, *International Journal of Advanced Research in Computer and Communication Engineering*, 2(9), 1-5, 2013.

[11] Kumar B., Prajapati A. (2013), Modelling and Simulation: Cyber War, *Procedia Technology*, 10, 987-997, 2013. http://dx.doi.org/10.1016/j.protcy.2013.12.447

[12] Livingston W. (2007), COTS: Commercial Off-The-Shell for Custom Off-The-Shelf, CrossTalk , www.stsc.hill.af.mil, 31-31, 2007

[13] Neumann J. (1996), *Theory of self-reproducing automata*, University of Ilinois Press, Edited and completed by A. Burks, 1996.

[14] [Online] ISO, 9241-11:1998, Ergonomic requirements for office work with visual display terminals (VDTs) - Part 11: Guidance on usability, March 1998.

[15] [Online] Available: www.trendmicro.com/cloud-content/us/pdfs/security-intelligence/reports/rpt- cashing-in-on-digital-information.pdf, TrendMicro, *Roundup, 2013 Annual Security*, 2013.

[16] [Online]. Available: www.securitybydefault.com/2013/09/crypters-localizando-firmas- delos.html. A. Pasamar, *CRYPTERS: Localizando firmas de los antivirus*, September 2013. [Last Access: September 28 2016].

[17] Rad B., Masrom M., Ibrahim S. (2012), Camouflage in malware: from encryption to metamorphism, *International Journal of Computer Science and Network Security*, 12, 74-83, 2012.

[18] Vinod P., Jaipur R., Laxmi R., Gaur M. (2009), Survey on malware detection methods, *Proceedings of the 3rd Hackers? Workshop on Computer and Internet Security*, 74-79, 2009.

[19] You I., Yim K. (2010), Malware obfuscation techniques: A brief survey, *Proceedings of the 2010 International Conference on Broadband, Wireless Computing, Communication and Applications*, 297-300, 2010. http://dx.doi.org/10.1109/BWCCA.2010.85

[20] Zhang Q., Reeves D. (2007), Metaaware: Identifying metamorphic malware, *Computer Security Applications Conference, ACSAC 2007*, 411-420, 2007.