engineering proceedings

MDPI

Proceeding Paper

Address Space Layout Randomization Comparative Analysis on Windows 10 and Ubuntu 18.04 LTS [†]

Raquel Vázquez Díaz [1], Martiño Rivera-Dourado [1,2,*], Rubén Pérez-Jove [1,2], Pilar Vila Avendaño [1,2,3] and José M. Vázquez-Naya [1,2]

[1] Grupo RNASA-IMEDIR, Departamento de Ciencias de la Computación y Tecnologías de la Información, Facultade de Informática, Universidade da Coruña, Elviña, 15071 A Coruña, Spain; raquel.vazquez1@udc.es (R.V.D.); ruben.perez.jove@udc.es (R.P.-J.); pilar.vila@forensic-security.com (P.V.A.); jose@udc.es (J.M.V.-N.)
[2] Centro de Investigación CITIC, Universidade da Coruña, Elviña, 15071 A Coruña, Spain
[3] Forensic & Security, 15190 A Coruña, Spain
[*] Correspondence: martino.rivera.dourado@udc.es
[†] Presented at the 4th XoveTIC Conference, A Coruña, Spain, 7–8 October 2021.

Abstract: Memory management is one of the main tasks of an Operating System, where the data of each process running in the system is kept. In this context, there exist several types of attacks that exploit memory-related vulnerabilities, forcing Operating Systems to feature memory protection techniques that make difficult to exploit them. One of these techniques is ASLR, whose function is to introduce randomness into the virtual address space of a process. The goal of this work was to measure, analyze and compare the behavior of ASLR on the 64-bit versions of Windows 10 and Ubuntu 18.04 LTS. The results have shown that the implementation of ASLR has improved significantly on these two Operating Systems compared to previous versions. However, there are aspects, such as partial correlations or a frequency distribution that is not always uniform, so it can still be improved.

Keywords: ASLR; memory; comparative analysis; Windows; Ubuntu

check for updates

Citation: Vázquez Díaz, R.; Rivera-Dourado, M.; Pérez-Jove, R.; Vila Avendaño, P.; Vázquez-Naya, J.M. Address Space Layout Randomization Comparative Analysis on Windows 10 and Ubuntu 18.04 LTS. *Eng. Proc.* **2021**, *7*, 26. https://doi.org/10.3390/engproc2021007026

Academic Editors: Joaquim de Moura, Marco A. González, Javier Pereira and Manuel G. Penedo

Published: 13 October 2021

1. Introduction

One of the main jobs performed by an Operating System is memory management. From a security perspective, there are plenty of attacks that leverage in the determinism of the memory space's layout to access and modify some parts of the memory. As an example of the most widespread ones, buffer overflow and string format must be mentioned.

In this context, ASLR (*Address Space Layout Randomization*) is a memory security mechanism which adds randomness into the virtual memory address space of a process. The aim of this technique is not to implement a patch to some specific problem, but making more difficult to exploit existing or future vulnerabilities.

The evolution and perfection of ASLR is notable during the last years. However, there exist an important amount of reports and studies that question the effectivity and solidity of this technique, even for the most recent Windows and Linux Operating Systems. In [1] Whitehouse performs a complete and detailed analysis of Windows Vista and, in its results, it reveals implementation errors that make the system not completely protected, situation that attackers can take advantage of to exploit memory vulnerabilities.

The main objective of this work was to base the methodology and transpararency on the Whitehouse study to obtain results of the real protection that offers ASLR, by comparing the behaviours and implementations on two recent distributions of Windows and Linux: Windows 10 and Ubuntu 18.04 LTS.

2. Materials and Methods

This section includes the methodology of the performed comparative analysis on the presented Operating Systems. Both the code and methodology are based on [1], but adapted to the modern versions of Windows and the specific features available on Linux.

2.1. Developed Code

The developed code [2], written in C for Linux and Windows, prints a log of the assigned memory addresses in each of the different areas of the process itself. With these logs, some bash scripts were crafted for the iterated execution of the tool, storing the memory addresses of each area in a CSV file. It is noteworthy that, for the compilation of the C programs, both Linux GCC and Windows WinGW compilers need concrete flags for ASLR to work.

In order to develop a comparative analysis for Windows and Linux, the code for both Operating Systems include almost identical verifications for memory addresses, from higher positions to lower ones: (1) stack, regarding variables inside a function; (2) heap, containing dynamic memory assignments; (3) BSS, where the uninitialized variables are kept; (4) data, regarding initialized static variables; (5) code.

In the case of Windows, the heap is verified by three means: creating a new heap, allocating memory in the heap and by using the `malloc()` function. Besides, the code for Windows also gets the PEB (*Process Environment Block*) address, which is a structure not existing in Linux. On the other hand, in Linux the code gets the heap address directly with the `malloc()` function. In the same way the Windows code gets the PEB address, the Linux code also gets addresses for some specific Linux memory parts like the VDSO (*Virtual Dynamic Shared Object*), as well as `brk()` and `mmap()` libraries.

2.2. Execution and Analysis

For the execution of the iterations, some batch and bash scripts were developed for Windows and Linux respectively, which are also available at [2]. These executions were performed in a PC with the following characteristics: AMD Athlon 64 ×2 4400+ with 4 GB de RAM. Regarding the specific releases of the Operating Systems, the tests were executed on Windows 10 10.0.17763 version and Ubuntu 18.04.2 LTS. In both of them, the scripts were run in two ways: continuously with 5 million iterations based on the work of Marco and Ripoll in [3]; and with system reboots each 100 iterations until reaching 500,000, considering a reasonable execution time and a significant number of repetitions, widely increased compared to the previous work of Whitehouse in [1].

Regarding the analysis, the logs resulting from the executions were pre-processed with the Pandas python library and then imported in Power BI Desktop, used as the main tool. The work is focused in the analysis of the level of entropy of the memory addresses, and the possible alterations compared with a desirable uniform distribution. For this purpose, we have used some techniques like pattern detection, correlations among areas, percentage of duplicated values and frequency distribution analysis.

3. Results and Conclusions

The first conclusion of this work is that the difference in the size of the user memory addressing space it is a determining factor in the behavioral differences between both Operating Systems. In Ubuntu, memory addresses present more entropy bits than in Windows, it presents a lower percentage of duplicated values and there exist fewer correlations among the different memory areas. In Windows, due to its memory addressing size, correlations seem not to be avoidable. Even though the addressing space in Ubuntu should be enough to avoid the correlations, they exist and in some cases are quite marked.

Regarding the frequency distribution in both systems, it tends to be uniform. It represents an improvement with respect to previous versions, as analyzed in Windows [1,4] and Linux [3,5]. However, in Windows the stack and PEB presents a bias towards lower memory addresses and in Linux, the percentage of repeated addresses in the VDSO is high.

Finally, the ratio of duplicated address values between the test without reboots and the test with reboots is higher in Ubuntu than in Windows. This factor can be understood as the impact of a reboot in the randomness of the addresses, and presents an unexpected behaviour. As shared libraries in Ubuntu are compiled as PIC, they take different virtual memory addresses in each process, in contrast with Windows where the values are equal between processes. Consequently, the impact of a reboot in Ubuntu should be lower on this matter in comparison with Windows. However, the results are higher in Ubuntu, maybe because of a specific characteristic in the implementation of ASLR in this Operating System.

In conclusion, ASLR has a better behaviour in Linux than in Windows, following the historical trend: it presents more entropy bits, less correlations and a better frequency distribution. However, considering its current memory addressing space, there are aspects that could be improved, like better correlation avoidance among the memory areas. In Windows, the correlation among memory areas is higher. In Windows 10, it would be convenient that the user memory addressing space was increased, for ASLR to be able to use more entropy bits and, in this way, the majority of the correlations would be avoided.

Author Contributions: Conceptualization, R.V.D., P.V.A. and J.M.V.-N.; methodology, R.V.D. and J.M.V.-N.; software, R.V.D.; validation, P.V.A. and J.M.V.-N.; investigation, R.V.D., P.V.A. and J.M.V.-N.; resources, P.V.A. and J.M.V.-N.; writing—original draft preparation, M.R.-D. and R.P.-J.; writing—review and editing, M.R.-D., R.P.-J. and J.M.V.-N.; supervision, P.V.A. and J.M.V.-N. All authors have read and agreed to the published version of the manuscript.

Funding: We wish to acknowledge the support received from the Centro de Investigación de Galicia "CITIC". CITIC, as Research Center accredited by Galician University System, is funded by "Consellería de Cultura, Educación e Universidade from Xunta de Galicia", supported in an 80% through ERDF, ERDF Operational Programme Galicia 2014–2020, and the remaining 20% by "Secretaría Xeral de Universidades" (Grant ED431G 2019/01). This work was also supported by the "Consellería de Cultura, Educación e Ordenación Universitaria" via the Consolidation and Structuring of Competitive Research Units—Competitive Reference Groups (ED431C 2018/49) and the COST Action 17124 DigForAsp, supported by COST (European Cooperation in Science and Technology, www.cost.eu, (accessed on 20 July 2021)).

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

PEB	Process Environment Block
VDSO	Virtual Dynamic Shared Object

References

1. Whitehouse, O. An Analysis of Address Space Layout Randomization on Windows Vista™. Symantec Advanced Threat Research, Tech. Rep. 2007. Available online: https://infocon.org/cons/Black%20Hat/Black%20Hat%20Europe/Black%20Hat%20Europe%202007/Presentations/bh-eu-07-whitehouse-WP-1.pdf (accessed on 20 July 2021).
2. rjetyk/aslr: Analysis and Comparative Study of Address Space Layout Randomization on Windows 10 and Ubuntu 18.04 LTS. Available online: https://github.com/rjetyk/aslr (accessed on 20 July 2021).
3. Ripoll-Ripoll, I.; Marco-Gisbert, H. Exploiting Linux and PaX ASLR's Weaknesses on 32- and 64-bit Systems. Black Hat 2016. Available online: https://paper.bobylive.com/Meeting_Papers/BlackHat/Asia-2016/asia-16-Marco-Gisbert-Exploiting-Linux-And-PaX-ASLRS-Weaknesses-On-32-And-64-Bit-Systems-wp.pdf (accessed on 21 July 2021).
4. Herrera Aristizabal, D.; Mora Rodríguez, D.; Yepes Guevara, R. Measuring ASLR Implementations on Modern Operating Systems. 2013 47th International Carnahan Conference on Security Technology (ICCST). Available online: https://ieeexplore.ieee.org/document/6922073 (accessed on 22 July 2021).
5. Ganz, J.; Peisert, S. ASLR: How Robust Is the Randomness? 2017 IEEE Cybersecurity Development. Available online: https://ieeexplore.ieee.org/abstract/document/8077804 (accessed on 22 July 2021).