ISBN 978–987–633–574–4

Monitoreo de Llamadas al Sistema como Método de Prevención de Malware

Fabián A. Gibellini, Sergio Quinteros, Germán N. Parisi, Milagros N. Zea Cárdenas, Leonardo Ciceri, Federico J. Bertola, Ileana M. Barrionuevo, Juliana Notreni, Analía L. Ruhl

Laboratorio de Sistemas / Dpto. de Ingeniería en Sistemas de Información/
Universidad Tecnológica Nacional / Facultad Regional Córdoba
Maestro M. Lopez esq. Cruz Roja Argentina S/N, Ciudad Universitaria (X5016ZAA) - Córdoba, Argentina
{fabiangibellini, ser.quinteros, germannparisi, milyzc, leonardorciceri, federicorbertola ilebarrionuevo, julinotreni, analialorenaruhl }@gmail.com

Resumen. De acuerdo con la categoría de un malware, se puede inferir que existen patrones de llamadas al sistema (syscalls) que permitirían descubrir qué tipo de malware se está ejecutando sobre un Sistema Operativo GNU/Linux y de esa manera reaccionar ante un ataque de estas características. Para esto es necesario monitorizar las llamadas al sistema en dicho sistema operativo. La herramienta que se presenta es un monitor de llamadas al sistema en tiempo real. Esta herramienta es parte de un proyecto homologado, cuyo objetivo es detectar malware basándose en patrones de llamadas al sistema en GNU/Linux.

Palabras Claves: Seguridad. Syscalls. Kernel. Linux. Detección.

1 Introducción

Existen diferentes tipos de malwares, además de los clásicos virus, podemos encontrar ransomwares, spywares, rootkits, troyanos y los más recientes, malwares residentes en memoria.

Según Raymond et al, el mayor desafío de crear un esquema completo de nombrado de malwares, se debe al número de muestras existentes de malware y a la frecuencia con la que nuevas muestras son descubiertas [1]. Si se considera la clasificación basada en comportamiento, propuesta por C. Elisan [2], se pueden distinguir a ransomwares, keyloggers, spywares, gusanos, troyanos, etc.

Cada uno de estos malwares intentan generar algún daño y para lograrlo es normal que utilicen al kernel para acceder a los recursos que necesitan. Un ransomware, por ejemplo, es una forma de software malicioso utilizado en ataques, en los que no se busca destruir irreversiblemente los datos, sino cifrar y cobrar por el servicio de recuperación de los datos cifrados [3] y para esto realiza operaciones de lectura y escritura sobre el disco, utilizando el kernel. Otro ejemplo es un keylogger, que es un software que se ubica entre el hardware y el sistema operativo e intercepta cada

pulsación de tecla y la almacena, para lo cual también ejecuta estas operaciones por medio del kernel.

Otro tipo de malware son los spywares, que se cargan de manera clandestina en una PC sin que su propietario se entere, y corre en segundo plano para ejecutar acciones a espaldas del propietario. Una de las formas en la que una máquina se infecta de spyware es por medio de troyanos. Existe una cantidad considerable de software gratuito que contiene spyware y el autor del software puede hacer dinero con este spyware [4].

Existe un tipo de malware, cuya variante se conoció en los últimos años, se trata de los malwares residentes en memoria o sin archivo (fileless). Son infecciones que no implican que los archivos maliciosos se descarguen o se escriban en el disco del sistema [5]. Generalmente están destinados a robar información y el atacante utiliza software existente, aplicaciones permitidas y protocolos autorizados en la víctima como portadores de actividades maliciosas. Este tipo de malware no puede ser identificado por los antivirus [6] [7].

Dentro de los casos reales de ataques de malware rememoramos algunos, como un estudio de Deloitte, el cual reporta que más de 360.000 computadoras, en más de 180 países, fueron afectadas por un ransomware conocido como WannaCry en mayo del 2017. Este ransomware generó costos económicos de 200 millones de dólares [8].

Como caso real de ataques sin archivos es la de un grupo de ciberdelincuentes, llamado Lurk, el cual fue uno de los primeros en emplear efectivamente técnicas de infección sin archivos en ataques a gran escala, técnicas que posiblemente se convirtieron en elementos básicos para otros malhechores. Se creía que Lurk había extraído más de $ 45 millones de dólares de organizaciones financieras, lo que finalmente afectó las operaciones, la reputación y los resultados de las víctimas [9].

Otro ejemplo de malware residente en memoria es Gold Dragon, que entre sus objetivos estaban los Juegos Olímpicos de Invierno en Corea del Sur. Dicho malware consistió de dos funciones primarias [10].

Una consultora de cibereconomía y ciberseguridad estima que los daños del cibercrimen tendrán un costo anual y global de seis millones de millones de dólares en 2021 [11]. Estos costos incluyen daños y destrucción de datos, dinero robado, pérdida de productividad, robo de propiedad intelectual, robo de datos personales y financieros, malversación, fraude, interrupción posterior al ataque en el curso normal de los negocios, investigación forense, restauración y eliminación de datos perjudicados y sistemas, y daño a la reputación [12].

Es debido al impacto asociado a estos malwares que conlleva al hecho de una búsqueda permanente para encontrar nuevas técnicas de prevención, o actualizar continuamente las ya existentes, de forma tal que minimice el impacto de estas amenazas. La más conocida los antivirus o antimalware, un antivirus compara los datos contra una base de datos de software malicioso (firma), si los datos mapean hacia alguna firma, entonces el antivirus muestra que el archivo está infectado [13].

Para los casos en que el malware supere las líneas de defensa planteadas y logre ejecutarse, es que se invierten costos y tiempos en investigar métodos reactivos de detección, de forma tal que una vez detectados puedan ser detenidos. Con relación a esto, predominan los trabajos dirigidos a la detección de un malware en particular, algunos de los mismos se mencionan a continuación.

Luckett et al, presenta una investigación para la detección de rootkits, donde examina una variedad de algoritmos de aprendizaje automático (como el de vecinos más cercanos, árboles de decisión, redes neuronales y máquinas de vectores de soporte) y propone un método de detección de comportamiento con un bajo consumo de energía de la CPU. Evaluando este método en los sistemas operativos Windows 10, Ubuntu Desktop y Ubuntu Server, junto con cuatro rootkits diferentes e identificando los algoritmos de mejor desempeño [14]. Muchos de los esfuerzos actuales para detectar los rootkits se basan en fuentes conocidas y son principalmente específicos de cada sistema operativo, por lo tanto, son ineficaces para detectar rootkits recién mutados, ocultos y desconocidos. Partiendo de esto, Ramani et al proponen un sistema para la detección de rootkits mediante la identificación de archivos ocultos. Este proceso de detección define un marco de monitoreo de procesos que mantiene continuamente una lista de archivos activos y puede detectar rootkits conocidos y desconocidos con una sobrecarga de rendimiento mínima.

También distinguimos un estudio que se centra en Cloud Computing, ésta es una instalación compartida y distribuida que son accedidas de forma remota por cualquier usuario final. Debido a esto, es que estos ambientes son vulnerables a varios ataques y requieren atención inmediata. Gupta y Kumar se concentran en ataques como rootkits, gusanos y troyanos en sistemas en un entorno Cloud Computing. Ellos describen críticamente y analizan técnicas para la detección de llamadas de sistema maliciosas y proponen una nueva técnica basada en la estructura de firmas de llamadas al sistema inmediatas, para determinar las ejecuciones de programas maliciosos en la nube [15].

Respecto a la detección de ransomware, malware que ha tenido un gran impacto en los últimos años, Popli y Girdhar realizaron un estudio al comportamiento de los ransomwares WannaCry y Petya, incluyendo análisis de procesos, análisis del sistema de archivos, análisis de persistencia y análisis de red, además de simulaciones en la herramienta Cuckoo con el objetivo de identificar técnicas que permitan distinguir cuándo un ransomware se convierte en un malware polimórfico y metamórfico [16].

Otros enfoques utilizan machine learning para identificar ransomwares y malwares en general, ya que implica un aprendizaje de los patrones en los datos para crear un modelo. Alrawashdeh y Purdy proponen un sistema una precisión limitada para la detección de ransomware dinámicamente utilizando machine learning [17]. Honeypots, es otra de las técnicas utilizadas, que a través de una configuración de archivos de señuelo permite “engañar” a un ransomware. Una vez que estos archivos son accedidos, el ransomware puede ser identificado. Incluso se ha aplicado también el análisis estadístico de llamadas a APIs entre una operación normal y un ransomware para generar modelos de detección de estos [18].

La siguiente y última técnica presentada se basa en identificar patrones en las llamadas al sistema (system calls) para así inferir si un proceso, en un sistema operativo GNU/Linux, puede ser considerado como un software potencialmente maligno. El proyecto de I+D (código SIUTNCO0007850) dentro el cual está inserto el presente trabajo propone ampliar esta técnica, en conjunto con identificación de patrones de datos para la detección de distintos malwares, como fileless, ransomware, entre otros de forma tal de lograr un monitoreo y detección de malwares en tiempo real y minimizar los daños económicos o de cualquier otra índole.

Una llamada al sistema o system call es un método o función que puede invocar un proceso para solicitar un cierto servicio al kernel o núcleo del sistema operativo [19]. Es válido aclarar que la necesidad de acceder a los servicios que brinda kernel a través de llamadas al sistema no es algo exclusivo de los malwares, sino que cualquier proceso lo realiza. La diferencia se encuentra en la manera de acceder, tanto hacia qué llamada, como la frecuencia y los parámetros con las que cada una es solicitada.

La detección de malwares en general y herramientas que permitan detectar cualquier tipo de software malicioso es un campo que sigue en expansión, dado a que el mapa de malwares tiende a seguir creciendo día a día. Actualmente se aplican diversas técnicas o combinaciones de una o más, como por ejemplo el análisis de comportamiento de procesos, el machine learning, las redes neuronales, el data mining y la clasificación de datos basada en comprensión [20]. De los cuales mencionamos algunos trabajos destacados, que junto con los descriptos anteriormente son considerados puntos de partida para el presente proyecto, y demostrando la importancia de generar nuevas herramientas que puedan detectar malware o inferir si un software (en ejecución) es malicioso antes que genere un impacto mayor del que ya ha generado.

Entre los métodos de detección de malware de análisis de comportamiento a través de llamadas al sistema, destacamos a Canzanese et al, que sostienen que un conjunto relativamente pequeño de tipos de llamadas al sistema provee una precisión comparable a la de modelos más complejos al momento de detectar procesos maliciosos. Además, afirma que su gran contribución son técnicas de extracción de características de procesos malicioso, con una tasa de falsos positivos muy baja [21]. En un artículo posterior, amplía este trabajo y describe un sistema que usa características extraídas de un seguimiento a las llamadas al sistema, lo que proporciona una lista de las llamadas que se ejecutan y el orden en el que lo hacen.

Combinando la minería de procesos y llamadas al sistema se encuentra Acampora et al, con un modelo para la detección de malware obtenido mediante un enfoque declarativo de minería de procesos a partir del análisis de algunos malwares en ejecución. La idea principal es que el conjunto de relaciones y patrones de ejecución recurrentes entre las llamadas al sistema de un malware en ejecución se pueden modelar para obtener una huella digital del mismo. Estas huellas se comparan y clasifican mediante un algoritmo de agrupación difusa para recuperar el mapa de relaciones de malware de todos los tipos de de malware considerados. La evaluación de este enfoque se realizó sobre un conjunto de datos de más de 4,000 software infectados en 39 tipos de malware [22].

Machine learning es aplicado en muchos trabajos, dentro de los cuales resaltamos los de Asmithad con Vinod y Saxe junto a Berlin. Los primeros proponen un sistema que utiliza machine learning para identificar procesos maliciosos en Linux. Extraen llamadas al sistema dinámicamente utilizando la herramienta Strace e identifican el mejor conjunto de características de procesos malignos y benignos para construir un modelo de clasificación de malware eficiente [23]. Saxe y Berlin incluyen, además de machine learning, redes neuronales en su sistema clasificador de malware de redes neuronales profundo que logra una tasa de detección utilizable del 95%, a una tasa de falsos positivos extremadamente baja y, según los autores, se adapta a volúmenes de ejemplos de capacitación en el mundo real sobre productos de hardware comunes [24].

Para finalizar mencionamos un trabajo que tiene en cuenta la complejidad de la información, y es expuesto por Alshahwan et al, en el que estudia la distancia de compresión normalizada (NCD) aplicada directamente a los binarios. La NCD es una medida teórica de la información y le permite obtener un 97.1% de precisión y una tasa de falsos positivos del 3% al momento de decidir si un programa sospechoso presenta mayor similitud a un malware o a un software benigno. Además, demostraron que esa precisión se puede optimizar combinando NCD con las tasas de compresibilidad de los ejecutables. Alshahwan remarca que el tiempo y el costo de cálculo de este método no es trivial [25].

Si bien hoy existen más tipos de malware que los descriptos, nos centramos en los que más impacto han generado en los últimos años. Lo mismo sucede con las técnicas de detección, para cada método o técnica (por ejemplo, machine learning) existen innumerables trabajos de los cuales solo hemos descripto los que consideramos más representativos y puntos de partida para el presente proyecto.

2 Desarrollo

Para determinar si un software es maligno es necesario monitorear sus procesos, por lo que para saber si en una computadora se está ejecutando un software malicioso es obligatorio monitorizar todos los procesos que se ejecutan en la misma. Para las computadoras que tienen un Sistema Operativo Linux/GNU es necesario monitorear las llamadas al sistema que realiza cada uno de estos procesos. Una llamada al sistema es una interfaz fundamental entre una aplicación y el kernel de Linux [26].

Para lograr este monitoreo se desarrolló un módulo del kernel [27] que intercepta ciertas llamadas al sistema y monitorea los procesos actuales a partir de dichas syscalls. De las 313 llamadas al sistema (syscalls) [19] que existen se seleccionó un conjunto, que permitirá identificar si un proceso es maligno. Las llamadas al sistema seleccionadas para esta versión del detector de malware están descriptas en la Tabla 1.

Table 1. Llamadas al sistema seleccionadas para esta primera versión del detector de malware.

ID	Nombre	Descripción
0	read	Lee bytes de un archivo referenciado por un file descriptor a un buffer.
1	write	Escribe bytes desde un buffer al archivo referenciado por el file descriptor
4	stat	Obtiene información sobre un archivo, como por ejemplo tiempo
2	openat	Abre un archivo
5	fstat	Obtiene información sobre un archivo

3	close	Cierra un file descriptor, por lo tanto, el archivo referenciado ya no puede ser accedido
101	ptrace	Permite observar y controlar la ejecución de un proceso

Estas llamadas al sistema fueron seleccionadas en base a la experiencia de los integrantes con ransomwares.

Para lograr este componente de monitoreo de llamadas al sistema fue necesario definir las siguientes interrogantes: ¿Cómo interceptar las llamadas al sistema? ¿La máxima cantidad de procesos se va a poder interceptar? ¿La máxima cantidad de llamadas al sistema por proceso se van a interceptar? ¿Se puede lograr que este monitoreo sirva para cualquier versión del kernel linux? Cuando se habla de proceso, se hace referencia al proceso asociado al programa que se está ejecutando en el sistema operativo GNU/Linux.

Lo primero que hace el módulo de monitoreo es inicializar la estructura de datos que va a usar para almacenar, a la cual se la puede conceptualizar como una matriz de 500x400 (Fig .1.).

La elección de 500 como límite de procesos a monitorizar es empírica, puede variar eventualmente. Por otro lado, el límite de 400 esta definido por la cantidad de llamadas al sistema definida en la Tabla de llamadas al sistema para sistemas x86 y x86_64 [19].

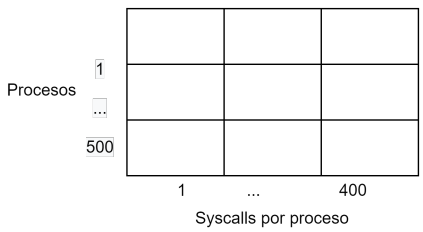

Fig. 1. Conceptualización estructura de datos para almacenar.

Una vez inicializada esta estructura, el siguiente paso es encontrar la tabla de llamadas al sistema, para poder "engancharse" e interceptar todas las llamadas que realizan las aplicaciones al kernel. Una vez que se encuentra esta tabla, es necesario quitar la protección contra escritura, osea el bit 16 del CR0 (Control Registry 0).

Los registros de control (CR0, CR1, CR2, CR3 y CR4) determinan el modo de operación del procesador y las características de ejecución de las tareas concurrentes.

El setear el bit 16 del CR0 permite a los procedimientos de nivel superior escribir en páginas de solo lectura, como por ejemplo en la página del Sistema Operativo donde se encuentra la tabla de llamadas al sistema [28]. Con esta protección contra escritura quitada queda "engachar" la función que cuenta las llamadas a cada syscall y por último volver a setear dicha protección contra escritura. A continuación, se muestra parcialmente el código que intercepta las syscalls originales y las redirige a una función

que acumula estas llamadas en base al PID y el ID de la syscall interceptada, para luego llamar a la función original de la syscall.

```
if (syscall_table != NULL) {
    write_cr0 (read_cr0 () & (~ 0x10000));
    original_open = (void *)syscall_table[__NR_open];
    syscall_table[__NR_open] = (long) &new_open;
    original_write = (void *)syscall_table[__NR_write];
    syscall_table[__NR_write] = (long) &new_write;
    original_read = (void *)syscall_table[__NR_read];
    syscall_table[__NR_read] = (long) &new_read;
    write_cr0 (read_cr0 () | 0x10000);
    printk(KERN_INFO "Syscall top iniciado\n");
 }
```

Una vez que este módulo es cargado al sistema, inicia con la lógica explicada recientemente, acumulando la cantidad de llamadas al sistema solicitadas por cada proceso que corre en el sistema operativo.

Cuando el módulo es descargado del kernel, se quita la protección contra escritura (bit 16 CR0 del procesador), se restablecen las funciones originales de cada syscall y se vuelve a establecer dicha protección contra escritura.

Una vez que este módulo es cargado al sistema, inicia con la lógica explicada recientemente, acumulando la cantidad de syscalls ejecutadas por cada proceso que corre en el sistema operativo.

Por último, se imprime un resumen de por cada PID “vivo” que el módulo encontró:

```
"PID: %1 - SYSCALL(%2) = %3"

Donde:

%1 Representa el Process ID del proceso.

%2 Representa el ID de la llamada al sistema.

%3 Representa la cantidad de veces que la llamada al
sistema con ID %2 fue llamada por el proceso con ID %1.
```

Las pruebas de esta herramienta de monitoreo por el momento han sido con ransomwares, logrando detectar un posible umbral de llamadas al sistema de escritura (write) de 1000, lo mismo para las llamadas al sistema de lectura (read) frente a programas “normales” o de uso diario.

3 Conclusiones

Este monitoreo de llamadas al sistema permitirá en base a ciertas reglas, por ejemplo, de cantidad de lecturas (read) o escrituras (write) o una combinación de ambas detener el proceso asociado a dichas llamadas al sistema.

Es importante mencionar que cuando se diseñó este módulo todavía no había salido la versión 5 del kernel Linux. Después de analizar esta nueva versión concluimos que, si bien este módulo de monitoreo no se ejecuta sobre la misma, el esfuerzo necesario para que esto ocurra no es mucho. Es decir, con algunas adaptaciones es posible llevar este monitoreo a la versión 5 del kernel Linux.

Como se mencionó previamente esta herramienta es un proyecto cuyo objetivo es detectar malware basado en patrones de llamadas al sistema en sistemas GNU/Linux cuyo resultado previsto es el desarrollo de un sistema de monitoreo y detección de malware para sistemas GNU/Linux, compuesto de dos herramientas.

La primera es una herramienta de monitoreo sobre las llamadas al sistema que cada proceso ejecuta y la cual generará diferentes vistas para que la información pueda ser comprensible. El actual trabajo forma parte de la primera herramienta y cuyos próximos pasos es generar una forma de visualización de estos datos. Esta forma de visualización debe ser en tiempo real y de lectura compresible, para posteriormente proceder con la segunda herramienta, la cual se integrará con la primera y permitirá detectar patrones posiblemente maliciosos en los procesos para informar al usuario de esta situación, pudiendo tomar una decisión.

Por otro lado, si bien las pruebas realizadas hasta ahora han sido con ransomwares también se ha identificado que esta técnica podría detectar minado de criptomonedas. Para esto sería necesario realizar pruebas para detectar llamadas al sistema usadas y posibles umbrales de cantidad de llamadas realizadas para inducir que se podría estar dando un minado de criptomonedas no autorizado.

Referencias

1. Canzanese R. (2015). Detection and Classification of Malicious Processes Using System Call Analysis. Recuperado el 28 de Mayo de 2019 https://pdfs.semanticscholar.org/8060/eae74c98a66cfcc736f4fca61d46f4dbc1d4.pdf.
2. Elisan C. (2015) Advanced Malware Analysis. McGraw-Hill. Capítulo 2. ISBN: 9780071819756.
3. K. Savage, P. Coogan, and H. Lau, (2018). The Evolution of Ransomware. Secur. Response, p. 57, 2015.
4. Tannenbaum A. (2009). Sistemas operativos modernos. Tercera edición. Pearson Educación. ISBN: 978-607-442-046-3.
5. Cruz M. (Junio 2017). Security 101: The Rise of Fileless Threats that Abuse PowerShell. Recuperado el 28 de Mayo del 2019 de https://www.trendmicro.com/vinfo/us/security/news/security-technology/security-101-the-rise-of-fileless-threats-that-abuse-powershel.

6. Viscuso M. (Febrero 2017). What Is a Non-Malware (or Fileless) Attack?. Recuperado el 28 de Mayo de 2019 de https://www.carbonblack.com/2017/02/10/non-malware-fileless-attack/.
7. Fileless Malware Attacks Are on the Rise, SentinelOne Finds, http://eds.b.ebscohost.com/eds/detail/detail?vid=2&sid=ee6fa74a-009d-4d30-9b94-fe6f826e0804%40sessionmgr103&bdata=Jmxhbmc9ZXMmc2l0ZT1lZHMtbGl2ZQ%3d%3d#AN=131651919&db=bsx.
8. Barros R., San-José P., Villanueva X. (2017) ¿Qué impacto ha tenido el ciberincidente de WannaCry en nuestra economía?. Deloitte. Recuperado de http://perspectivas.deloitte.com/hubfs/Campanas/WannaCry/Deloitte-ES-informe-WannaCry.pdf.
9. Yarochkin F., Kropotov V. (Febrero 2017). Lurk: Retracing the Group's Five-Year Campaign. Trend Micro. Recuperado el 28 de Mayo de 2019 https://blog.trendmicro.com/trendlabs-security-intelligence/lurk-retracing-five-year-campaign/?_ga=2.257191439.544304570.1558717075-418567566.1558717075.
10. Beek C., Dunton T., Grobman S., Karlton M., Minihane N., Palm C., Peterson E., Samani R., Schmugar C., Sims R. A., Sommer D., Sun B. (Junio 2018). McAfee Labs Threats Report. McAfee. Recuperado el 28 de Mayo de 2019 https://www.mcafee.com/enterprise/en-us/assets/reports/rp-quarterly-threats-jun-2018.pdf.
11. Morgan S. (Mayo 2017). 2018 Cybersecurity Market Report. Recuperado el 28 de Mayo de 2019 de https://cybersecurityventures.com/cybersecurity-market-report/.
12. Morgan S. (Diciembre 2018). Cybercrime Damages $6 Trillion By 2021. Recuperado el 28 de Mayo de 2019 de https://cybersecurityventures.com/cybercrime-damages-6-trillion-by-2021.
13. Choudhary, S., Saroha, R., & Beniwal, S. (2016). How Anti-virus Software Works?. International Journal of Advanced Research in Computer Science and Software Engineering, (April 2013), 5–7.
14. Luckett P., McDonald J., Glisson W., Benton R., Dawson B., Doyle A. (2018). Identifying stealth malware using CPU power consumption and learning algorithms". Journal of Computer Security 26(2018) 589-613. DOI 10.3233/JCS-171060.
15. Gupta, S. & Kumar, P. (2015). An Immediate System Call Sequence Based Approach for Detecting Malicious Program Executions in Cloud Environment. Wireless Pers Commun, 81: 405.
16. Popli N.K., Girdhar A. (2019) Behavioural Analysis of Recent Ransomwares and Prediction of Future Attacks by Polymorphic and Metamorphic Ransomware. In: Verma N., Ghosh A. (eds) Computational Intelligence: Theories, Applications and Future Directions - Volume II. Advances in Intelligent Systems and Computing, vol 799. Springer, Singapore.
17. Ransomware Detection Using Limited Precision Deep Learning Structure in FPGA.
18. Kok S.H., Abdullah A., Jhanjhi N. Z., Supramaniam M. (2019). Ransomware, Threat and Detection Techniques: A Review. IJCSNS International Journal of Computer Science and Network Security, VOL.19 No.2. Recuperado el 28 de Mayo de 2019 de http://paper.ijcsns.org/07_book/201902/20190217.pdf.
19. Searchable Linux Syscall Table for x86 and x86_64. https://filippo.io/linux-syscall-table/. Última visita 08/08/2021.
20. Roman Gonzalez A. (2012). Clasificación de Datos Basado en Compresión. Revista ECIPeru, pp.69-74. ffhal-00697873. Recuperado el 28 de Mayo de 2019 de https://hal.archives-ouvertes.fr/hal-00697873/document.
21. Canzanese, R., Mancoridis, S., & Kam, M. (2015). System Call-Based Detection of Malicious Processes. In Proceedings - 2015 IEEE International Conference on Software Quality, Reliability and Security, QRS 2015 (pp. 119-124). [7272922] Institute of Electrical and Electronics Engineers Inc.. https://doi.org/10.1109/QRS.2015.26.

22. Acampora, G., Bernardi, M. L., Cimitile, M., Tortora, G., Vitiello, A. (2018). A fuzzy clustering-based approach to study malware phylogeny. IEEE International Conference on Fuzzy Systems, , 2018-July doi:10.1109/FUZZ-IEEE.2018.8491625.
23.Asmithad K. A., Vinop P. (2014). A machine learning approach for linux malware detection. In: 2014 International Conference on Issues and Challenges in Intelligent Computing Techniques (ICICT), ISBN 978-1-4799-2900-9.
24. Saxe J., Berlin k. (2015). Deep Neural Network Based Malware Detection Using Two Dimensional Binary Program Features. Recuperado de https://ia802808.us.archive.org/14/items/arxiv-1508.03096/1508.03096.pdf.
25. Alshahwan N., Barr E.T., Clark D., Danezis G. (2015). Detecting Malware with Information Complexity. Recuperado de https://archive.org/details/arxiv-1502.07661/page/n5
26. Manual de Programación de Linux. Página Oficial: http://man7.org/linux/man-pages/man2/syscalls.2.html.
27.Modulos kernel Linux. https://www.kernel.org/doc/html/v4.16/admin-guide/README.html
28. Mayo 2020. The Intel 64 and IA-32 Architectures Software Developer's Manual. Intel. Link a https://software.intel.com/content/dam/develop/public/us/en/documents/325384-sdm-vol-3abcd.pdf. Última visita 08/08/2021.