Secure File Systems for the Development of a Data Leak Protection (DLP) Tool Against Internal Threats

Isabel Herrera Montano, Isabel de la Torre Díez
Department of Signal Theory and Communications, and Telematics Engineering
University of Valladolid
Valladolid, Spain
isaherrera1403@gmail.com , isator@tel.uva.es

Jose Javier García Aranda, Juan Ramos Diaz, Sergio Molina Cardín, Juan José Guerrero López
Department of Innovation, Nokia, Maria Tubau Street, 9, 28050, Madrid, Spain
jose_javier.garcia_aranda@nokia.com,
juan.ramos_diaz.ext@nokia.com,
sergio.molina_cardin.ext@nokia.com,
juan.guerrero_lopez.ext@nokia.com

Abstract — **Data leakage by employees is a matter of concern for companies and organizations today. Previous studies have shown that existing Data Leakage Protection (DLP) systems on the market, the more secure they are, the more intrusive and tedious they are to work with. This paper proposes and assesses the implementation of four technologies that enable the development of secure file systems for insider threat-focused, low-intrusive and user-transparent DLP tools. Two of these technologies are configurable features of the Windows operating system (Minifilters and Server Message Block), the other two are virtual file systems (VFS) Dokan and WinFsp, which mirror the real file system (RFS) allowing it to incorporate security techniques. In the assessment of the technologies, it was found that the implementation of VFS was very efficient and simple. WinFsp and Dokan presented a performance of 51% and 20% respectively, with respect to the performance of the operations in the RFS. This result may seem relatively low, but it should be taken into account that the calculation includes read and write encryption and decryption operations as appropriate for each prototype. Server Message Block (SMB) presented a low performance (3%) so it is not considered viable for a solution like this, while Minifilters present the best performance but require high programming knowledge for its evolution. The prototype presented in this paper and its strategy provides an acceptable level of comfort for the user, and a high level of security.**

Keywords – Data Leak Protection; DLP; Dokan; Minifilters; Server Message Block; WinFsp.

I. Introduction

Currently, concern about data leakage from entities has become more acute, as data movement is increasingly necessary to facilitate telework imposed by the pandemic [1]. Data leakage is the disclosure of information from organizations or companies to other entities or unauthorized persons [2]. A Data Leak Protection System (DLPS) is primarily designed to monitor the flow of data in an organization and apply predefined measures on terminal devices or on the network within the organization. The measures range from logging activities, sending warnings to end users and administrators to quarantining data or completely blocking them [3], [4].

In the survey conducted by the authors of [5] with 150 employees from different companies and sectors, it was found that the existing DLP tools are considered very intrusive and hinder the work of employees. This affects the productivity of the workers and therefore that of the company. In the current circumstances, a less intrusive system is needed that allows the worker to telework without difficulties but at the same time protects the information. Therefore, the main objective of this paper is to propose secure File Systems (FS) for the development of DLP tools.

In order to meet the proposed objective, a prototype of a DLP tool focused on the insider threat, but transparent to the user, was designed. Internal threat is the one posed by employees or any person considered trustworthy in an organization to the organization's information, being the main source of data leakage nowadays [1], [6].

The designed prototype was implemented with four different technologies to assess and compare its results. Two of them are Windows OS tools, and the other two are Virtual File Systems (VFS). A VFS is an abstraction layer on top of a real file system (RFS), i.e. an intermediate layer between the system calls and the RFS driver. The purpose of a VFS is to allow client applications to access different types of RFS in a uniform way [7]. They also provide the possibility to perform operations before and after reading, writing, etc. In exchange for this intermediate "translation" between the applications and the RFS, part of the original performance of the file system is lost, this being a parameter to be taken into account in the assessment of the proposed technologies.

In the literature, studies were found that present DLP tools where security measures are applied in the FS, such as the case of [8], where the high level security of content protection and the DLP scheme (CPSec DLP) scheme based on mandatory encryption at the kernel level are proposed, and the SM2 algorithm is adopted for session key management. The system is based on kernel-level middleware that enables "write encryption, open decryption" operations. In [9], a DLP tool is proposed, where unwanted file system operations, such as CopyFile or ReadFile, are blocked by implementing a mini-filter driver. The study [10] focuses on the implementation of a DLP solution that protects any data that are about to leave an endpoint using a Windows Minifilter driver framework. The application provides a plain text view of files even if they are stored in encrypted form on disk. The authors of [11] develop a free DLPS for Microsoft Windows. It is based on blocking file copying through USB ports using Windows minifilter technique and machine learning. In the study [12] the Dokan VFS is proposed for the design and implementation of an encrypted cloud storage system.

The contributions of this paper with respect to existing similar studies in the literature are the following: 1) It proposes and assesses four tools for secure FS implementation. 2) It presents novel tools in the literature (WinFsp and Server Message Block) for the implementation of secure FS, and the comparison of their performance.

2022 17th Iberian Conference on Information Systems and Technologies (CISTI)
22 – 25 June 2022, Madrid, Spain
ISBN: 978-989-33-3436-2

Authorized licensed use limited to: Hochschule Rosenheim (FH). Downloaded on November 01,2023 at 10:55:04 UTC from IEEE Xplore. Restrictions apply.

Section II of this paper presents the architecture and functionality of the prototype DLP tool with each of the technologies studied. Section III shows the results and discussion of its assessment, and Section IV presents the conclusions and future lines of research necessary for the DLP tool to be able to compete with other solutions in the market.

II. ARCHITECTURE, REQUIREMENTS AND FUNCTIONALITY

Although the developed tools are prototypes (and not complete tools), the pursued requirements for a complete tool should be:

a) Non-intrusive and lightweight

b) Cypher key not stored but formed by concatenation of outputs of certain pieces of code, named challenges, that perform computations (not validations) based on execution context such as location, network, computer fingerprint, etc.

c) Keep format and length of the ciphertext.

d) Prevent the extraction of uncyphered files from computer unless a trace is left behind for the organization.

The prototype is designed for Windows 10 and it was developed in C and C++ programming languages on the Visual Studio platform. The development was carried out on a virtual machine to protect the OS from the real machine.

This prototype is focused on the main security holes detected in the internal threat, the output of confidential information via pen drives, and the Internet [2]. The tool consists of creating a new volume that will be the FS configured with the security measures, where the employee will work. The FS created will be a mirror of the FS being monitored, it is based on the encryption of documents in the read or write input and output (I/O) operations, conditioning the operation according to the application making the request (by capturing the Process Identifier (PID) parameter) and the location to be read from or written to. Table I shows the operation used for the implementation. In this table, the letter "M" corresponds to the part of the hard disk that is monitored, and "K" is a monitored external disk (pendrive). The operational table could be configurable to meet encryption/decryption needs.

TABLE I. OPERATION OF THE DLP TOOL PROTOTYPE

Application	**Examples (configurable in lists)**	**Disk**	**Operations**
Browsers and email	outlook.exe chrome.exe firefox.exe	M:	READ: Encrypt
Any application (including browsers and email)	word.exe paint.exe	VFS of pendrive (K:)	READ: Decrypt WRITE: Encrypt

In the above table it is possible to upload unencrypted files, by uncyphering them previously with a specific tool which makes an encryption with an inverse cypher, leaving traces for the organization. Therefore, the ciphering scheme requirements for this DLP tool involve the existence of inverse of the cypher and that the cyphered data size equals the original data size.

This protection is compatible with encryption tools such as Microsoft BitLocker, preventing from user to extract the hard disk or boot the computer from an external drive. The operation of the DLP tool preventing malicious users from extracting uncyphered files from the computer.

For this approach to be valid it is required that the computer is handed over to the user with the C disk (or any other disk to be monitored) hidden from the file explorer, so that the user cannot access C when opening the computer. This is done by the company before handing over the PC to the employee.

One way to do this is to hide the C drive in the File Explorer using GPEDIT.msc by modifying the settings in "the path": Local Computer Policy > User Configuration > Administrative Tools > Windows Components > File Explorer (setting the "Prevent access to drives from My Computer" key to Enabled and selecting "Restrict C drive only"). You can also set the "Hide these specified drives in My Computer" to Enabled and select "Restrict C drive only" to prevent them to be seen in "My Computer".

Once the configuration described above is done, the user will have the desktop in M, to avoid creating files and folders in C. The same with "my pictures", "my videos" and "my stuff". Links (shortcuts) from the desktop to C executables still work even if they are on a desktop in M, because C still exists, it is just hidden. If we were to remove the letter C from Diskpart, those executables would no longer be accessible via shortcuts, so only access to C from the file explorer should be removed. Also, any existing desktop links pointing to folders in C should be modified to point to M before giving the PC to the user.

For the pendrive scenario, the possibility of deactivating Windows automatic mounting is being studied to mount portable devices (external disks, pendrive, etc.) in subfolders of "C:\Portables\" (each subfolder could be named as the GUID associated with the disk). Then monitor each subfolder (i.e. "C\Portables\{...}\") on a free letter (i.e. "K:").

A. Minifilters

Although not exactly the final design, to perform a secure FS based on Minifilters, the filesystem operations are intercepted. [9][11] encrypting and decrypting the data intercepted or obtained when necessary so that no data leaving the system is compromised in the clear. Specifically, the goal is, just after the read operation a decryption is performed before returning the data to the requesting application; just before the write operation an encryption is performed on the data being sent to the filesystem. In this way, transparent file encryption can be achieved.

Figure 1 shows the operation of the prototype developed with the minifilters technology for the pendrive scenario. Disk C represents the local disk and disk T the pendrive. In any case, this box should be understood as the set of lower level driver stacks (i.e. filesystem, volume, partition, and storage). Arrows represent requests and responses to read and write operations. On disk C, reads and writes are simple and do not require encryption or decryption (files in the clear). On disk T, reads are followed by decryption, and writes are preceded by encryption (encrypted files).

Authorized licensed use limited to: Hochschule Rosenheim (FH). Downloaded on November 01,2023 at 10:55:04 UTC from IEEE Xplore. Restrictions apply.

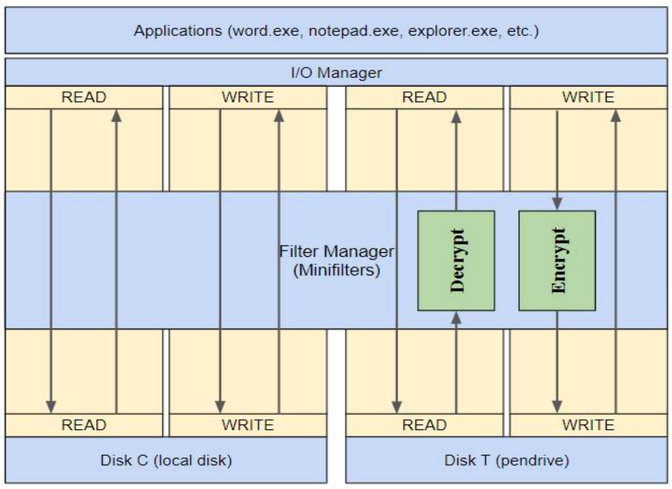

Figure 1. Diagram of the functioning of the prototype with Minifilter technology

The functioning is simple. Firstly, the user installs the minifilter (this is only done once, and it is installed forever). Then you start the minifilter (this would be done automatically at system startup, so no external functionality is required). From that moment on, the disks that have been configured as "local" will work in clear, and the rest will be taken as the pendrive in Figure 1, that is, the files will be encrypted and worked on by transparently encrypting and decrypting them at read/write time.

Since read/write operations are intercepted, both scenarios (pendrive and Internet) are feasible by applying cyphering policies based on requester process (identified in each call) or destination (pendrive or HDD).

B. Dokan

Dokan is a library for the development of a Windows user-mode file system, which is used to develop VFS in a simple way [12][15][16]. The Dokan Library [16] contains a user-mode DLL (dokan1.dll) and a kernel-mode file system driver (dokan1.sys). Once the file system driver is installed, a VFS can be created that functions like the Windows RFS.

File operation requests from user programs (e.g., CreateFile, ReadFile, WriteFile, etc.) will be sent to the Windows I/O subsystem (running in kernel mode), which will then forward the requests to the Dokan file system driver (dokan1. sys), as shown in Figure 2. By using functions provided by the Dokan user mode library (dokan1.dll), file system applications can register callback functions to the file system driver. The file system driver will invoke these callback routines to respond to the requests it received. The results of the callback routines will be sent back to the user program. For example, when Windows Explorer requests to open a directory, the CreateFile with Direction option request is sent to the Dokan file system driver and the driver invokes the CreateFile callback provided by the file system application. The results of this routine are sent back to Windows Explorer as a response to the CreateFile request. Thus, the Dokan file system driver acts as a proxy between user programs and file system applications. The advantage of this approach is that it allows programmers to develop user-mode file systems that are safe and easy to debug.

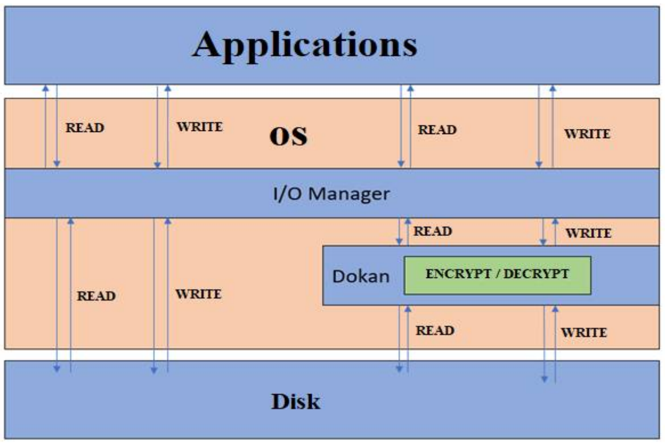

Figure 2. Diagram of the functioning of the prototype with Dokan technology

As shown in Figure 2 the OS contains a module called I/O Manager, which is accessed by some applications, such as synchronizers (onedrive.exe, googlesync.exe, dropbox.exe, etc.). In these cases, the applications make the call directly to the memory, without going through the file system. This represents a limitation of our prototype in the face of synchronizing applications. This topic will be developed in the section dedicated to the evaluation of the prototypes, where the limitations of each of them are presented.

Like with minifilters, read/write operations are intercepted and both scenarios (pendrive and Internet) are feasible by applying cyphering policies based on requester process (identified in each call) or destination (pendrive or HDD).

C. Server Message Block (SMB)

It is a Windows protocol similar to NTFS for network file sharing. The idea of building a secure FS based on SMB is based on the ability to intercept read and write messages on a file server [19].

For a pendrive scenario, if the mounting of the pendrive is done through an SMB server that resides on the same PC, all read and write calls can be intercepted, and thus decrypted on read and encrypted on write. Conditional decryption is also possible so that if the file is not encrypted, it is not decrypted on read.

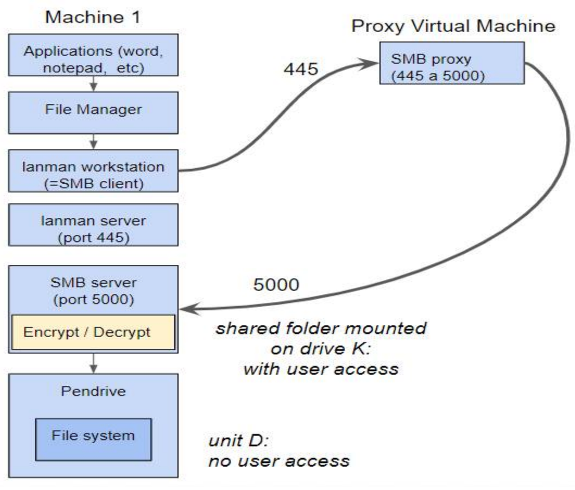

Figure 3. Components of the SMB prototype for a pendrive scenario

Authorized licensed use limited to: Hochschule Rosenheim (FH). Downloaded on November 01,2023 at 10:55:04 UTC from IEEE Xplore. Restrictions apply.

Figure 3 identifies the components of the prototype that solves the flash drive threat. As can be seen, there is a need for a second machine due to the presence of the Windows service "lanman server" on port 445 and the impossibility of identifying in Windows a shared folder through a port other than 445. This makes it necessary to map the requests through an intermediate proxy service, which for the prototype we have created in a virtual machine, although in a definitive service it would be minimized through a docker or similar. Another possibility could be to deactivate Microsoft lanman server and use 445 on local machine without a proxy.

The Internet scenario (Google drive) consists of encrypting (or not decrypting if the hard drive is encrypted) those files opened for reading by Chrome in an open Google drive tab. The scenario is intended to be extensible to other processes, such as mail or synchronizers.

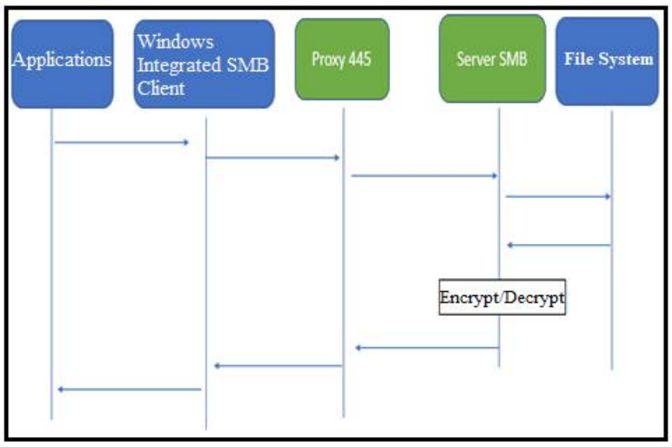

Figure 4. Diagram of the functioning of the prototype with SMB technology

In the pendrive scenario, the commissioning steps are: 1) Connect the pendrive, 2) Once the pendrive is connected, it should be mounted as a disk drive, but the user (non-administrator) should not have access to the drive. This process of change of permissions would be automatic, 3) Next, also automatically, the new disk must be shared as a folder from the SMB process of the administrator server, 4) Finally, automatically this "remote" folder (since we will identify it as a proxy machine resource) is mounted as disk drive K to which the user does have access. As shown in the diagram in Figure 4, from this moment on applications could use disk K and encryption is performed transparently.

In the Google Drive scenario this prototype has the difficulty that in SMB the originating PID of the request does not arrive as a parameter, so they need alternative solutions such as the following:

1) *Check open files by processes:*

In this solution, two tests are performed, one based on the functionality provided by the Python psutil library and the other using the Windows Handle tool [20], obtaining similar results. The test carried out with the Handle is shown in [21], the experiment consisted of opening a file (a video) from chrome.exe and then trying to find out which process has the mp4 file open. This would be the task that the SMB server would do when receiving a read operation, and in a positive case it would encrypt the reading. The idea is to see which user processes there are and those with the name "chrome.exe", exploring their open files looking for the file whose reading has been requested.

2) *Solution based on Sandbox:*

This solution involves encrypting everything that is written to the directory used as a sandbox, which must be mounted as a disk and shared in the same way as is done with the pendrive. Writes would always be encrypted. The sandbox solution would be viable for browsers, mail, and synchronization programs such as Google sync, but it would make it difficult to use the computer, since before uploading a file to a website, it would have to be moved to a "special" directory, which would be the sandbox directory. Figure 5 shows the functioning of the SMB prototype for the Internet scenario with sandbox technology.

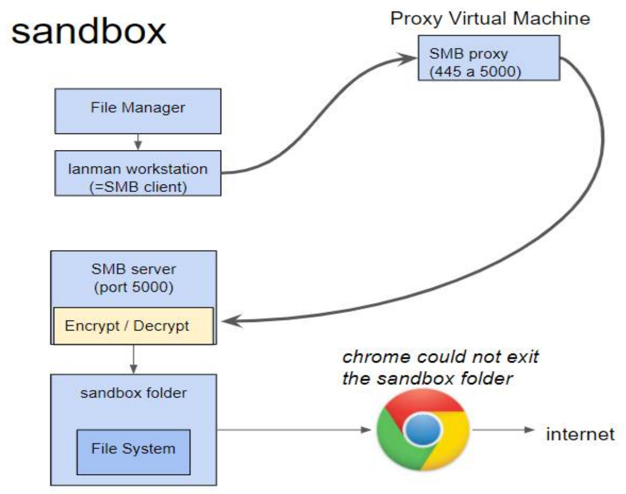

Figure 5. SMB prototype functioning for the scenario Internet using Sandbox

D. *WinFsp*

Generally, any data or storage can be organized and presented as a file system through WinFsp, with the advantage that any Windows application can access the data through the standard Windows file APIs. WinFsp is released as a Microsoft Software Installation (MSI) installer that includes a signed driver and all the files needed to run and develop user-mode file systems on Windows. The installer supports native Windows, FUSE, .NET, and Cygwin file systems [24].

With WinFsp it is possible to create VFS on RFS drives such as NTFS, or standalone drives with FSs created with this technology. In this prototype two tests are performed with the "passthrough" VFS provided as an example in the WinFsp documentation [24].

Basically, the functioning of the VFS created with WinFsp consists in passing through the file system operations requested by the kernel to an underlying file system (NTFS or FAT32 in our case). All operations are intercepted and launched against the RFSs, conditionally encrypting/decrypting based on the logic defined in the application lists. Figure 6 shows the desired functioning scheme of the VFS prototype with WinFsp, with M being the VFS monitoring the local disk C, and P the VFS monitoring the pendrive D.

Authorized licensed use limited to: Hochschule Rosenheim (FH). Downloaded on November 01,2023 at 10:55:04 UTC from IEEE Xplore. Restrictions apply.

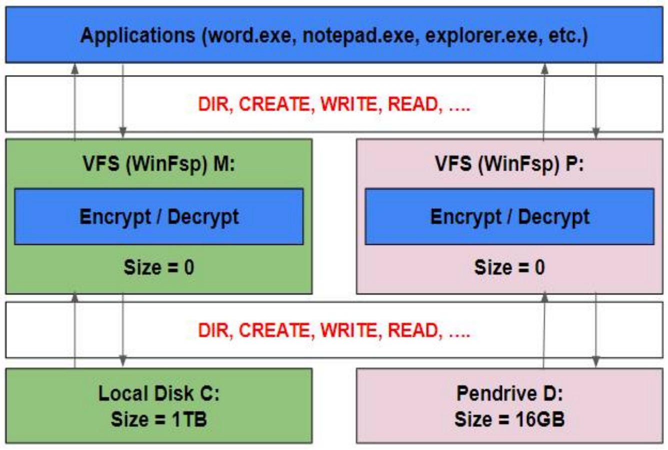

Figure 6. Diagram of the functioning of the prototype with WinFsp technology

The development of the file system application, in this case, passthrough (the application that is already created with the main functions that an FS must have), consists of functioning on the already defined functions, such as the read and write functions where you want to insert encryption operations.

With WinFsp it is not possible to capture the process that makes the request in READ and WRITE but in OPEN, and therefore, in this case the PID has been captured in the OPEN request. This allows to identify the process that makes the request to obtain its handle, path, type of application, etc., to compare the process that makes the READ and WRITE request with the table of operations and thus determine the operation that the prototype must perform in each case. For the tests performed, a table has been created and the process has been directly compared with "chrome" for the case of reading (Google Drive scenario) so that if this application is the one that makes the request, the file is automatically encrypted before reading it.

In the implementation of the prototype, it was detected that the VFS presents problems to map the RFS-FAT32, this problem is still being investigated since according to the research done so far WinFsp does not present this type of problems.

For the tests performed and presented in this document, the flash drive has been formatted as FS-NTFS.

III. Results and Discussion

The source code of the prototype can be obtained from the repositories of the "Secureworld" project: Minifilters [19], Dokan [20], SMB [21] and WinFsp [22]; a user manual showing how to configure it for testing is provided in these repositories. Prototype demonstration videos with the different technologies are also available from Minifilters [23], Dokan [24], SMB [25], and WinFsp [26].

A. Assessment

To assess the different technologies, performance, complexity of implementation and limitations were analyzed.

For performance, tests were performed where the time taken by the operating system to read and copy 1000 files of 1MB from a source directory to a target directory without and with the prototype for each implementation, encrypting and decrypting according to the operating table, was calculated.

To assess the complexity, the following aspects were taken into account 1) programming language, 2) interference with the operating system (need to program in a VM to avoid crashes and blue screens, etc.), 3) flexibility of the solution: the complexity of evolving it. In addition, attention was paid to the limitations of each of the implementations for their use different from the native form.

The following table summarizes the results of the assessment tests performed on each implemented prototype.

TABLE II. SUMMARY OF ASSESSMENT RESULTS

Parameter	**Minifilter**	**Dokan**	**SMB**	**WinFsp**
Test pendrive	Feasible	Feasible	Feasible but requires a proxy	Feasible for pendrive with FS NTFS as it presents incompatibilities with FAT32
Test Google Drive	Feasible	Feasible	Unfeasible due to low performance in the time required to obtain the identifier of the process making the request.	Feasible
Complexity	High	Medium	Low	Medium
Limitations	Does not allow intercepting calls from applications that use memory mapping	Does not allow intercepting calls from applications that use memory mapping	Refresh slow, create file error, does not have the originating PID, and therefore cannot resolve the Google Drive scenario.	1) Error in RFS Fat32 mapping. 2) Does not allow for intercepting calls from applications that use memory mapping

As shown in Table II, the Dokan and WinFsp prototypes are of medium complexity, since they work with operations previously designed through the API, which makes them complex at user level, using the C++ programming language. While the Minifilters prototype presents a high complexity, both in the development (due to the specific knowledge required) and in the testing phase (due to the diverse casuistry of the operations to be tested). However, the prototype developed with SMB presents a low complexity but at the same time is too slow, as shown in Figure 7. In addition, it has been detected that it presents several limitations, such as the malfunction in the creation of a new file from the file manager, where it presents an error; the refresh of existing files in the folder is not immediate; it does not capture information of the PID that requests the read/write operation, which makes it unfeasible to combat the Internet security hole, as this is necessary to configure the encryption operation by identifying the process that makes the request, for example a browser. For this prototype other possible solutions were investigated such as [27] and [28], which, due to the complexity involved for the

Authorized licensed use limited to: Hochschule Rosenheim (FH). Downloaded on November 01,2023 at 10:55:04 UTC from IEEE Xplore. Restrictions apply.

user, have not been tested, so we have taken the prototype scenario only up to this point of definition, without actually testing it.

In the prototype developed by Dokan, WinFsp and Minifilters the memory mapping performed by the synchronization folders forces it to encrypt these folders. These file system-based technologies cannot intercept the memory mapping calls, since the requests do not pass through the file system. With Minifilters, it was found that memory-mapped files present a problem; once a file has been memory-mapped, no read/write requests are made to the file system, but rather it deals directly with memory and therefore no requests are intercepted, and hence no operation table can be applied.

The WinFsp prototype so far has only presented the limitation of FS Fat32 mapping, in addition to the one mentioned above.

Figure 7 shows a comparison of approximate performance in terms of the execution time of read and write operations encrypting in each prototype with respect to the execution time of these same operations in the RFS. Taking as a reference the execution time of the RFS operations, which represents 100%. It can be seen that the prototypes with the Minifilter and WinFsp technologies are the ones that provide higher performance, while the prototype with SMB is too slow.

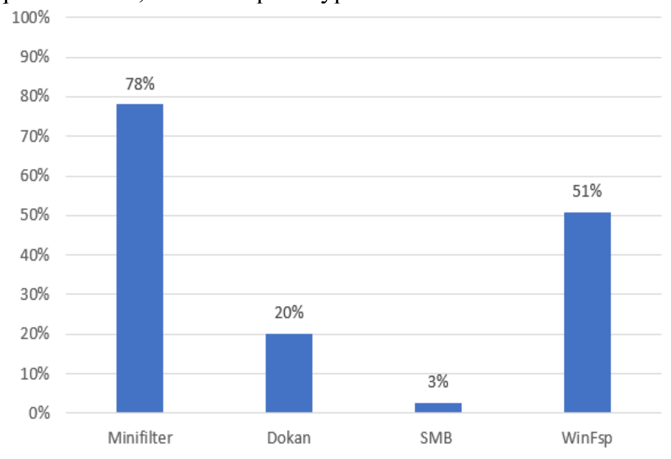

Figure 7. Percentage performance of VFS with respect to RFS

IV. Conclusion and Future Works

This paper, in relation to related works found in the literature, has the advantage of assessing four tools, which allows the reader to know the level of complexity, limitations and performance of each of them for the development of a DLPS transparent to the user, and thus be able to choose the one that best suits their needs and hardware characteristics. Our approach shows tools for the development of a DLPS and not the DLPS itself, so that the user can develop his own DLPS based on the study carried out. In the works found in the literature, systems are developed at the kernel level, which be able to very difficult for a user with little programming knowledge, however, in this paper we present two tools (WinFsp and Dokan) that have a user-level API that facilitates such work. Another technique used in the work found is the blocking of copy and read operations, as well as the USB ports of the computer, which can make the work of the employee very tedious, so this paper proposes transparent techniques for the user.

The development of the prototype with Dokan and WinFsp technologies proved to be very efficient and simple, since the developer interacts with user applications (mirror.exe and passthrough.exe, respectively), i.e., the most complicated part of request management is already done, allowing the developer to focus only on the development of encryption in the scenarios proposed and the different security techniques to be incorporated into the FS.

Unlike existing tools, the prototype presented does not sacrifice comfort at the expense of security, providing an acceptable level of comfort for the user, and a high level of security. The level of convenience is not full for the user, encrypted files downloaded from the Internet and personal files to be sent to a pendrive without encryption have to go through a decryption application. This application will be developed in future lines. It is convenient to opt for this strategy, since dispensing with this application does not provide full comfort if the user has to upload official documents to an Internet portal or save personal documents on a pendrive; this would not be possible.

As future lines of work, we propose to thoroughly develop each part of the system to guarantee the highest possible security for the information. We will suggest a suitable encryption for this system, which meets the requirements of being lightweight, dependent on a private key formed by challenges, and maintaining the format and length of the ciphertext. The main strength of the tool lies in the private key for encryption and decryption, which will be obtained from the calculation of the results of different challenges, forming the fragments of the key [29][30][31], which, in turn, will allow the user to be identified through biometric data, the location of the computer by nearby Wi-Fi signals or GPS, among other challenges that will be studied and proposed in future work.

Acknowledgment

This research has been partially supported by the Center for the Development of Industrial Technology (CDTI), Spanish Ministry of Science and Innovation under the project named "Technologies for secure digital relationships in a hyperconnected world (Secureworld)".

We thank Nokia Spain for the close collaboration to achieve successful results.

References

[1] "IBM," *IBM Report: Cost of a Data Breach Hits Record High During Pandemic*, 2021. https://newsroom.ibm.com/2021-07-28-IBM-Report-Cost-of-a-Data-Breach-Hits-Record-High-During-Pandemic.

[2] S. Alneyadi, E. Sithirasenan and V. Muthukkumarasamy, A survey on data leakage prevention systems, *Journal of Network and Computer Applications*, vol. 62, pp. 137–152, Feb. 2016, doi: 10.1016/j.jnca.2016.01.008.

[3] K. Gupta and A. Kush, "A Forecasting-Based DLP Approach for Data Security," 2021, pp. 1–8.

[4] B. Hauer, Data and Information Leakage Prevention within the scope of Information Security, *IEEE Access*, vol. 3, pp. 2554–2565, 2015, doi: 10.1109/ACCESS.2015.2506185.

[5] Y. Bertrand, K. Boudaoud, and M. Riveill, "What Do You Think About Your Company's Leaks? A Survey on End-Users Perception Toward Data Leakage Mechanisms," *Frontiers in Big Data*, vol. 8, Oct. 2020, doi: 10.3389/fdata.2020.568257.

[6] "IBM," *Security Informe 2020*, 2020. https://newsroom.ibm.com/cost-of-a-data-breach-2020.

[7] C. Hu, F. Chen, and H. Zheng, "Researches on the Security Protection and Inspection Method for Confidential Documents Based on Linux Operating System," in *Proceedings of the 3rd International Conference on Machine Learning and Soft Computing - ICMLSC 2019*, 2019, pp. 249–252, doi: 10.1145/3310986.3311029.

[8] Z. Ma, "CPSec DLP: Kernel-Level Content Protection Security System of Data Leakage Prevention," *Chinese Journal of Electronics*, vol. 26, no. 4, pp. 827–836, Jul. 2017, doi: 10.1049/cje.2017.05.002.

Authorized licensed use limited to: Hochschule Rosenheim (FH). Downloaded on November 01,2023 at 10:55:04 UTC from IEEE Xplore. Restrictions apply.

[9] A. Buda and A. Colesa, “File System Minifilter Based Data Leakage Prevention System,” in *2018 17th RoEduNet Conference: Networking in Education and Research (RoEduNet)*, Sep. 2018, pp. 1–6, doi: 10.1109/ROEDUNET.2018.8514147.

[10] D. Porízek*, “Transparent Encryption with Windows Minifilter Driver,” 2019.

[11] S. Thombre, “Freeware Solution for Preventing Data Leakage by Insider for Windows Framework,” *2020 International Conference on Computational Performance Evaluation, ComPE 2020*, pp. 44–47, 2020, doi: 10.1109/ComPE49325.2020.9200160.

[12] J. Yin, J. Yang, and Y. Chen, “The Design and Implementation of User Autonomous Encryption Cloud Storage System Based on Dokan,” in *Proceedings of the 2016 International Conference on Computer Science and Electronic Technology*, 2016, vol. 569, no. Cset, pp. 917–928, doi: 10.2991/cset-16.2016.18.

[13] H. Sparenberg, A. Schmitt, R. Scheler, and S. Foessel, “Virtual File System for Scalable Media Formats,” 2011.

[14] “Dokan.” https://github.com/dokan-dev/dokany/wiki/Build.

[15] N. A. Mohamed, A. Jantan, and O. I. Abiodun, “Protect governments, and organizations infrastructure against cyber terrorism (mitigation and stop of server message block (SMB) remote code execution attack),” *International Journal of Engineering Research and Technology*, vol. 11, no. 2, pp. 261–272, 2018.

[16] Windows, “Handle de Windows,” 2019. https://docs.microsoft.com/en-us/sysinternals/downloads/handle.

[17] Nokia, “Video Prueba Handle Windows para SMB,” *Video Handle*, 2020. https://drive.google.com/file/d/13Utq9IAs-mtjgE24AsZLOGpoCqz14uEk/view?usp=sharing.

[18] B. Zissimopoulos, “WinFsp,” *Documentacion WinFsp*. .

[19] Nokia, “SecureworldProject / prototype-MINIFILTER-NOKIA,” *Repositorio del prototipo de minifilters*, 2020. https://github.com/SecureworldProject/prototype-MINIFILTER-NOKIA.

[20] Nokia, “prototype-DOKAN-NOKIA,” *Repositorio del prototipo Dokan-Nokia*, 2020. https://github.com/SecureworldProject/prototype-DOKAN-NOKIA.

[21] Nokia, “Prototype SMB,” *Código fuente*, 2020. https://github.com/SecureworldProject/prototype-SMB-NOKIA.

[22] Uva-Nokia, “Prototipo WinFsp,” *Codigo fuente del prototipo WinFsp*, 2020. https://github.com/SecureworldProject/prototipo-WinFsp.

[23] Nokia, “Video Demostración Prototipo Minifilter,” *VFS Minifilter*, 2020. https://drive.google.com/file/d/1WQE8aUGdXDc5dmIH_jVPyh2VlSLj-tpD/view?usp=sharing.

[24] Nokia, “Video Demostración Prototipo Dokan,” *VFS Dokan*, 2020. https://drive.google.com/file/d/1FE1Tz-BrS2HE31mIOLCyfCOtC5d8FMB-/view?usp=sharing.

[25] Nokia, “Video Demostración Prototipo SMB,” *Demo SMB*, 2020. https://drive.google.com/file/d/1BMcJrWxi2TFxhVreKi1vnhPRTyrjbMh8/view?usp=sharing.

[26] Uva, “Video Demostración Prototipo WinFsp,” *Demo WinFsp*, 2020. https://drive.google.com/file/d/1suYTNjescP4uM21DxbZD8usxgDSrxW2z/view?usp=sharing.

[27] “Browser in the Box Effective protection against attacks from the Internet.” [Online]. Available: https://scdn.rohde-schwarz.com/ur/pws/dl_downloads/dl_common_library/dl_brochures_and_datasheets/pdf_1/service_support_30/Browser_Bro_en_3607-5760-32_v0100.pdf.

[28] “SANDBOXIE.” http://www.sandboxie.com/.

[29] A. Garcia, P. Holgado, J. J. Garcia, J. Roncero, V. Villagrá, and H. Jalain, “Sistema de cifrado basado en contexto aplicado a prevención de fuga de datos,” in *Proceedings XIII Jornadas de Ingenieria Telematica - JITEL2017*, Sep. 2017, pp. 93–100, doi: 10.4995/JITEL2017.2017.6576.

[30] P. Holgado, A. García, J. J. García, J. Roncero, V. A. Villagrá, and H. Jalain, “Context-based Encryption Applied to Data Leakage Prevention Solutions,” in *Proceedings of the 14th International Joint Conference on e-Business and Telecommunications*, 2017, vol. 4, no. Icete, pp. 566–571, doi: 10.5220/0006475205660571.

[31] J. J. A. Garcia Aranda, “EP 2 709 333 A1 EUROPEAN PATENT APPLICATION,” 2014.

Authorized licensed use limited to: Hochschule Rosenheim (FH). Downloaded on November 01,2023 at 10:55:04 UTC from IEEE Xplore. Restrictions apply.