Methods of Isolation for Application Traces Using Virtual Machines and Shadow Copies

George Pecherle
Faculty of Electrical Engineering and Information Technology, University of Oradea Oradea, Romania

Cornelia Győrödi
Faculty of Electrical Engineering and Information Technology, University of Oradea Oradea, Romania

Robert Győrödi
Faculty of Electrical Engineering and Information Technology, University of Oradea Oradea, Romania

***Abstract*—To improve the user's experience, almost all applications save usage data: web browsers save history and cookies, chat programs save message archives and so on. However, this data can be confidential and may compromise the user's privacy. There are third party solutions to automatically detect and wipe these traces, but they have two problems: they need a constantly updated database of files to target, and they wipe the data after it has been written to the disk. Our proposed solution does not need a database and it automatically reverts the application to its initial (clean) state, leaving no traces behind. This is done by using a monitoring process developed by us and the Volume Shadow Copy Service that takes snapshots when the application runs and restores them at the end of the run.**

***Keywords*—security; privacy; application traces; data wiping; virtual machines; shadow copies; sandbox**

I. Introduction

Storage capacity of disks has increased during the recent years - sometimes exponentially - facilitating a large number of programs to work together and make sometimes very complex operations, and also facilitating the amount of data that these programs work with. Therefore, for the user of a modern computer system, it has become impossible to know or manually check the data and the software stored on a computer system, for reasons that relate to the huge volume that is stored and to the way programs hide and/or encrypt data during their normal operations. It is noted in this context, that there is a strong need to protect the data stored on a computer system against external agents that might compromise the security without the user's knowledge, to ensure the user's privacy and a proper functioning of the operating system.

This protection was achieved by designing modern operating systems and even computer systems to avoid vulnerabilities to external factors and facilitate the implementation of subsystems designed for maintaining the security of the data.

The study from [1] shows that any system has vulnerabilities, 14,900 files being detected as files that are part of programs designed to attack an Android operating system, based on the Linux kernel, that has not traditionally been the target of attacks until recently.

Another important issue that comes up is that programs can leave traces of their usage that contain private information about the user. This is why most Internet browsers have features that allow the user to start private sessions that don't save usage traces, such as data about the accessed sites, passwords and other data entered in web forms [2] or so-called cookies used for saving sessions on specific portals that require registration or other information to identify the user, features that improve the user's experience on the web. The need to solve this problem started to become real, especially because of websites that lead the user to expose more data to the Internet browser (such as online shopping sites, flight bookings, banking services, etc.), very often on devices that do not belong to the user and that can be accessed by other people, programs or sites that are not trustable.

II. Current Solutions To Problems of Data Security and Private Data Protection

There are a lot of software methods to ensure data protection (both in terms of system reliability and data security) and they are implemented in various combinations by the operating systems and by specialized software.

Along with the methods and mechanisms that ensure data security and fault tolerance of the system (built into the operating system), there are also methods of protection provided by third parties. These programs are either actively working to detect problems generated in the system (such as suites of programs that detect viruses and other malware, or software that verify the integrity of data or of different subsystems), or programs that are passive. The experience shows that none of these methods can fully respond to security and privacy needs of the user. What differentiates security solutions from the user's perspective is their ability to either prevent a problem, or try to solve it after it has been generated.

Of course, users prefer the first method (preventive security) and this paper is about this type of security protection. Two methods will be presented and then how we used them to design our privacy protection system, that isolates private data using virtual machines and shadow copies.

A. Virtual Machines

A good compromise between performance / data accessibility and data security can be made by using virtual machines that are essentially computer systems with a similar behavior as the original (physical) ones. They are actually abstract implementations of real systems [3], and because of this reason, they can be protected against external factors, by filtering the communication ways with the outside environment, and also by the fact that the state of these machines can be saved periodically so there is always the

possibility of rolling back to a pre-infection state. Virtual machines can benefit from the existence of more processor modes, the supervisor mode of the virtual machine being named the "hypervisor".

This method, of using virtual machines, although it is useful in some situations - for example when you want to obtain a controlled environment to implement solutions with specific purposes, of "closed box" type [4] - has only minor advantages from a security point of view, compared to an operating system on a real machine. This is because, in most scenarios, the virtual machine cannot be completely isolated from the outside environment in order to meet the usage guidelines. And the more the accessibility of virtual machines increases, the more the security level of virtual machines becomes closer to that of a real system.

B. Sandboxes

A good method to secure the data and the operating system, that is closer to using virtual machines, are the so-called "sandboxes". These are programs that allow the execution of other programs (called host programs) with a limited and controlled set of resources.

These sandboxes work in different ways, depending on the purpose they were designed for. A recent trend that is worth mentioning is that part of the modern and the most successful operating systems, low-level sandboxes (close to the operating system layer) have been implemented for a large amount of applications [5].

Using a sandbox for a secure system can be achieved very easy, by installing and using a virtual machine with an operating system installed, for example Oracle VM Virtual Box [21].

III. Comparison With Other Similar Methods

There are a lot of solutions (especially software programs) that can automatically detect and securely delete traces of application usage. There are usually two types of solutions:

- Solutions that detect application traces and react to this phenomenon AFTER the data has been written to disk (securely erase it), sometimes long after that (e.g. when the user launches the erase process), leading to serious privacy concerns.
- Solutions that detect application traces and react to this phenomenon BEFORE the data is written to the disk. This is the most efficient and secure method because there is a very low risk for sensitive data to fall into the wrong hands.

Our method described in this paper falls into the second category and its originality comes from the way modified data is intercepted and handled: saving snapshots of the original data using the Volume Shadow Copy Service then restore it at the end. Before restoring the original data, a secure erase of the modified files could be implemented (we could make a list of modified files, using the preparation callback routines of the minifilter driver).

We have also identified another method that is also from the second category above and it is called Sandboxie [19]. This solution saves application traces in a special sandbox, not on their original locations. This way, sensitive data can be erased all at once, from the same place.

A problem we have detected with our method is: what if the user wants to save data in a file and that should remain modified? Taking into consideration the idea from the Sandboxie solution described above, we could modify our solution to declare a "safe" area, where this data can be saved and that should be the user's responsibility to clean after he no longer needs that data.

A limitation of our solution, but also of many other available solutions is that it's only for Windows operating systems.

The disadvantages of the solutions from the first category (just detect where applications save data and securely erase it) are obvious: there will always be a delay between saving sensitive data and erasing it. During this time, the data can fall into the wrong hands. Also, the locations where applications save data change rapidly, so a constantly updated database of locations needs to be maintained. And this can lead to sensitive data not being caught and erased.

Also, another method that is worth mentioning and that was previously proposed by me, implements an algorithm that determines the sensitivity of files using a pre-defined set of rules made by the user. These rules are self-adaptable, in a way that they can improve themselves, taking into account patterns detected in other files securely erased by the user [20].

IV. Proposed Sandboxed Solution To Isolate Private Data

The research we have performed and that will be presented here is a sandbox whose purpose is to isolate only files that are accessed by the host programs, using features already implemented in the Windows operating system. Some of them are features related to the file system, driver development for the Windows operating system and the Volume ShadowCopy Service (VSS).

The main benefit introduced by this research is the change of the rules imposed by a traditional sandbox, through controlled accessibility to files stored on the disk. The effects of this solution and also its efficiency in solving the data privacy problems will be presented next in this paper.

A. Motivation

In this section, we will present a research that demonstrates the flexibility and the simple way in which the security and protection methods of the operating systems can be extended. This design and implements a sandbox that is limited to protecting and isolating changes made by applications in the file system.

As shown previously, the full virtualization and very restrictive sandboxes don't always meet the security and data protection requirements of the users.

Moreover, the increased complexity of a virtual machine or that of a sandbox that abstracts levels very close to the kernel generates security problems that are similar to those of the operating system itself. Also, modern operating systems offer advanced protection mechanisms and policies that have been thoroughly tested. For this reason, it is enough to use features that are present in the operating system.

B. Requirements

Our research had to meet the following requirements:

- To be fully compatible with all recent Windows operating systems
- To run with minimal hardware resources
- To integrate with the operating system (through context menus, for example)
- To integrate with the graphical user interface of the host operating system
- To allow the user to choose the application to start in protected mode
- To prevent altering system files by the application that has started in protected mode
- To use a minimum amount of memory and to add a minimum wait time when the application is launched
- To avoid writing on the disk as much as possible
- To use a minimum amount of source code to avoid errors
- To use a minimum amount of source code that runs in supervisor mode
- The application should work transparently to the user

C. Specifications

The research we have done is based on a system application, and for this reason, we have used the C++ programming language that offers system oriented development features and that is very well supported by some of the Windows API functions, by their functionality and the documentation that is available for them. In order to develop a driver that we will use to monitor and block I/O requests with the file system, we have identified the filter drivers of the file system as being the type of drivers that we should use. These drivers had to be implemented in the C programming language. Another technology we have identified and that we have used to implement the file manipulation system is the VSS (Volume Shadow Copy Service). This technology was introduced in Windows Server 2003, it works with the NTFS file system and facilitates the creation of snapshots (saving the state of the file system on an NTFS volume), without affecting the normal operation of the system, using a technique similar to Copy-On-Write at block level [6] [7]. This technique operates at block level (not file level) and makes differential copies instead of full copies of the original data. First, a copy of the original data is created. Then, whenever a change to the original volume occurs, before this is written to disk, the block that is about to be modified is written to an area that stores "differences" to the original data. Using the original data blocks and the differences blocks, a shadow copy can be built that represents the copy of the data at the time it was created. The main advantage of this method is the speed, because it only writes the differences [18].

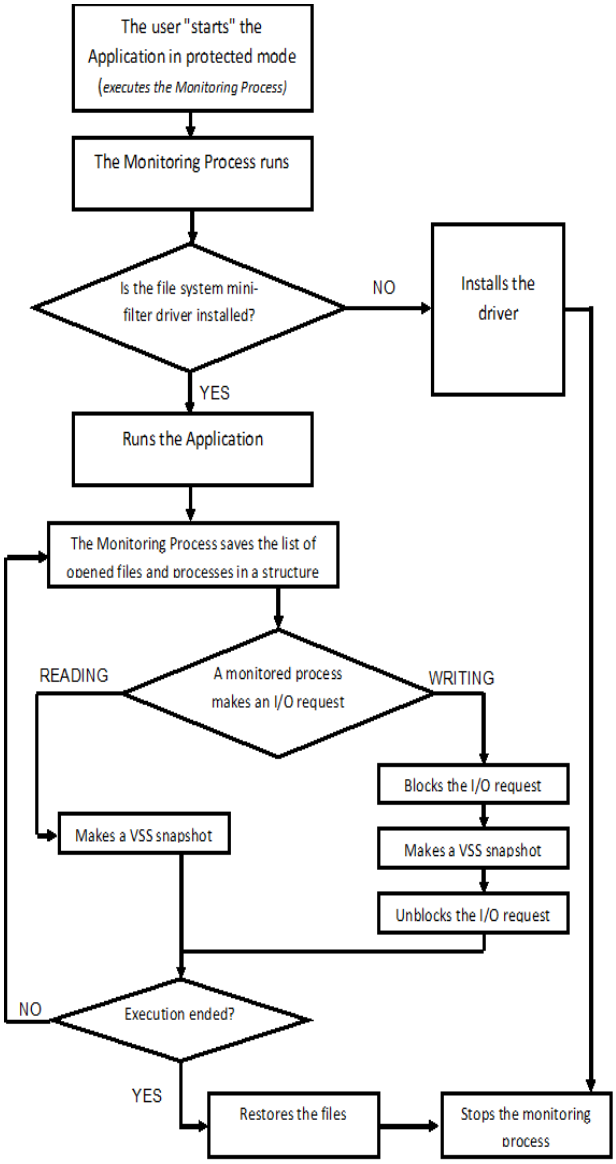

Fig. 1. The workflow chart of the monitoring process and its interaction with the protected (targeted) application

The minifilter driver can be implemented using a file system minifilter [8]. This makes it possible to write a small amount of source code for the supervisor mode, minimizing the risks of programming errors.

The concept of minifilter drivers is simple: a user makes a request for file I/O. Then the I/O manager sends the request to the file system. At that point, the filter manager intercepts the I/O request and calls the registered minifilters in their altitude orders. The altitude of a minifilter is a unique identifier that determines the order of attachment. The altitudes are allocated and managed by the operating system and it ensures that an instance of a minifilter driver is loaded at a location that is

appropriate to other instances of minifilter drivers. Also, minifilter drivers can register a preoperation callback routine, or a postoperation callback routine or both of them. Preoperation callback routines are called in descending order of altitudes (in case there are more minifilters drivers installed), then the I/O operation takes place, then the postoperation callback routines are called in ascending order of altitudes, from lowest to highest. [16]

To edit the C and C++ source code, we have used the integrated development environment of Microsoft Visual Studio 2012. In order to test and debug the program, we have used a virtual machine provided by Microsoft for Windows 7 - called the Windows XP Mode. The use of a virtual machine is just a method of protection of the system we have developed, on which we do not want to run untested code in supervisor mode.

Figure 1 describes the general workflow chart of the monitoring process and its interaction with the protected (targeted) application. One important thing to note is that the monitoring process will be launched when the user starts the protected (targeted) application.

If the file system minifilter driver is not present, the application will ask the user to install the driver before exiting. If the driver is present, the targeted application will be launched automatically. Then, a data structure will be created inside the monitoring process, to store a list of accessed files for each process of the targeted application. The monitoring process will continue to execute in parallel with the process (or the processes) of the targeted application. When an I/O request from one of the monitored processes is detected, the type of I/O request will be verified. For input operations, a preventive snapshot will be done (as optimization method). However, for output operations, the operation will be blocked until the Volume Shadow Copy Service performs the snapshot. For both situations (input and output operations), an internal verification will be done to detect when the execution of the targeted applications ends. If it's not ended, the monitoring process will continue to monitor the targeted application. If the targeted application ends, the original files will be restored using the Volume ShadowCopy Service snapshot and the execution of the monitoring process will end.

D. Implementation

The implementation consists of three separate components, each representing a separate project in Visual Studio:

- The monitoring process - that will monitor all I/O requests from the targeted application
- A service that will use SCM (Service Control Manager) [9] to start the monitoring process automatically and also to avoid User Account Control (UAC) messages, that elevate permissions for users. We will use the functions to install and uninstall the service.
- The file system minifilter driver

The documentation required for each of these components was obtained from:

- The documentation about writing Windows applications available from the MSDN (Microsoft Developer Network) website [10],
- The documentation about writing services in C++ using the Visual Studio 2012 development environment [11] and
- The documentation to write a file system minifilter driver, that is also available from MSDN [12]

The application has been integrated in the Windows graphical user interface using the Windows Registry, by extending a shell component (the main component of the Windows GUI), more exactly by changing a registry key from the Windows Registry as shown at [13]. The extension was done on the "exefile" subclass, and this means that a new context menu option will be present only for executable files and their shortcuts. We have called this context menu option, "Run in Protected Mode", as shown in Figure 2.

E. The Protected Mode Service

To start the monitored process in protected mode, we have used a service, that we called the Protected Mode Service. This is actually run in command line and can accept various command line parameters. The usage of the Protected Mode service is below:

```
pmservice [mode] [servicecommandstype] [command]
```

The [mode] parameters can accept 2 values: "service" or "process". If it's "process", normal process work is done. If it's "service", the next parameter is verified that it has one the following values:

- "config": the user can run some configuration commands, such as "query" (to retrieve and display the current service configuration, using the DoQuerySvc() function), "describe" (to update the service description to a default value, using the DoUpdateSvcDesc() function), "disable" (to disable the service, using the DoDisableSvc() function), "enable" (to enable the service, using the DoEnableSvc() function), "delete" (to delete the service, using the DoDeleteSvc() function).
- "control": the user can run some control commands, such as "start" (to start the service, if possible, using the DoStartSvc() function), "dacl" (to update the service DACL [14] to grant start, stop, delete, and read control access to the Guest account, using the DoUpdateSvcDacl() function), "stop" (to stop the service, using the DoStopSvc() function).
- "install": the service is installed in the Service Control Manager (SCM) database [15] by using the SvcInstall() function that we have implemented. Then a new entry ("Run in Protected Mode") is added in the context menu of executable files, by creating a special registry key under HKEY_CLASSES_ROOT, as below:

```
LPWSTR pm_command =
(LPWSTR)"C:\\dev\\ProtectedMode\\PMService.exe
process \"%1\"\0";

HKEY hkey;
DWORD  dwDisposition;
DWORD dwType, dwSize;

if (RegCreateKeyEx(HKEY_CLASSES_ROOT,
TEXT("exefile\\shell\\Run in Protected
Mode\\command"), 0, NULL, 0, 0, NULL, &hkey,
&dwDisposition) == ERROR_SUCCESS)
{

dwType = REG_SZ;
       dwSize = (wcslen(pm_command) + 1) *
sizeof(WCHAR);
       RegSetValueEx(hkey, NULL, 0, dwType,
(LPBYTE)&pm_command, dwSize);
       RegCloseKey(hkey);
}
```

And of course, we have implemented a class called ProcessMonitor, that has a constructor with the following arguments: ProcessMonitor(TCHAR *handlePath, TCHAR *monitoredProcPath) and that starts the monitored process as a child process, using the CreateProcess() function [17] and then makes the processing as described.

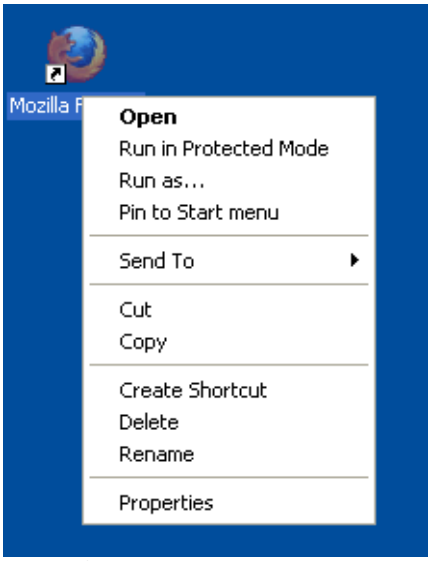

Fig. 2. Screenshot of the context menu with our new option "Run in Protected Mode" that will launch the monitoring process on this application

V. Conclusions

This paper implements a new way to protect the data manipulated by applications: isolating the data in a protected environment. This is a requirement because most applications have an uncontrollable and unpredictable way of saving their data and this can lead to privacy issues.

This is done using a system feature used for backup and system restore purposes, the Volume ShadowCopy Service, by doing a snapshot when the application makes the first I/O request and restoring it when the application ends.

The Windows operating system offers developers a set of programming interfaces that allow the extension of the system capabilities, also in supervisor mode, without the need to write long and complicated programs that are likely to have errors. The driver system, especially the filter drivers system, allows the extension of the functionality set for developers who need to obtain a different system behavior, by taking advantage of the hardware capabilities. On another note, the system offers security and protection measures that are meant to increase the system's reliability and the user's experience.

In the future, we would like to extend this research to mobile applications taking into account that data privacy on mobile devices is now a requirement of both home and corporate users.

References

[1] Y. Namestnikov, IT Threat Evolution: Q2 2012, Kaspersky Lab ZAO, http://www.securelist.com/en/analysis/204792231/IT_Threat_Evolution_Q1_2012.

[2] Private Browsing - Browse the web without saving information about the sites you visit, https://support.mozilla.org/en-US/kb/private-browsing-browse-web-without-saving-info.

[3] J. E. Smith and R. Nair, The architecture of virtual machines,Computer, vol. 38, nr. 5, pag. 32-38, 2005.

[4] T. Garfinkel, B. Pfaff, J. Chow, M. Rosenblum and D. Boneh, Terra: A virtual machine-based platform for trusted computing, ACM SIGOPS Operating Systems Review, vol. 37, nr. 5, pag. 193-206, 2003.

[5] App Sandbox Design Guide - http://goo.gl/tjG5B

[6] B. Milewsky, Virtual Machines: Memory - http://corensic.wordpress.com/2011/11/28/virtual-machines-memory/

[7] M. Howard, Address Space Layout Randomization in Windows Vista, - http://blogs.msdn.com/b/michael_howard/archive/2006/05/26/608315.aspx

[8] File System Minifilter Drivers - http://msdn.microsoft.com/library/windows/hardware/ff540402

[9] Microsoft TechNet - Service Control Manager - http://technet.microsoft.com/en-us/library/dd349449(v=ws.10).aspx

[10] Windows Development Reference - http://msdn.microsoft.com/en-us/library/windows/desktop/hh447209(v=vs.85).aspx

[11] Windows Service Template (C++) - http://msdn.microsoft.com/en-us/library/8dy6h580(v=vs.80).aspx

[12] File System Minifilter Drivers - http://msdn.microsoft.com/library/windows/hardware/ff540402

[13] Extending Shortcut Menus - http://msdn.microsoft.com/en-us/library/windows/desktop/cc144101(v=vs.85).aspx

[14] DACLs and ACEs (Windows) - http://msdn.microsoft.com/en-us/library/windows/desktop/aa446597(v=vs.85).aspx

[15] Service Control Manager (Windows) - http://msdn.microsoft.com/en-us/library/windows/desktop/ms685150(v=vs.85).aspx

[16] Filter Manager Concepts (Windows Drivers) - http://msdn.microsoft.com/en-US/library/windows/hardware/ff541610

[17] CreateProcess function (Windows) - http://msdn.microsoft.com/en-us/library/windows/desktop/ms682425(v=vs.85).aspx

[18] How Volume Shadow Copy Service Works - http://technet.microsoft.com/en-us/library/cc785914(v=ws.10).aspx

[19] Sanboxie - Sandbox software for application isolation and secure Web browsing - http://www.sandboxie.com/

[20] George Pecherle, Cornelia Gyorödi, Robert Gyorödi, Bogdan Andronic, Iosif Ignat "New Method of Detection and Wiping of Sensitive Information", ICCP 2011, IEEE 7th International Conference on Intelligent Computer Communication and Processing, 2011, Cluj-Napoca, Romania, 25-27 August, ISBN 978-1-4577-1478-8, CFP 1109D-PRT, pages 145-148

[21] Oracle VM VirtualBox - https://www.virtualbox.org/