POSTER: Dr. Watson Provides Data for Post-Breach Analysis

Wasim Ahmad Bhat
Department of Computer Sciences
University of Kashmir
wab.cs@uok.edu.in

S.M.K. Quadri
Department of Computer Sciences
University of Kashmir
quadrismk@kashmiruniversity.ac.in

ABSTRACT

Nowadays security systems have become highly sophisticated. However, breaches are inevitable. Nevertheless, post-breach analysis is performed to assess the severity of the breach and to trace the intruder's actions. This paper proposes *drWatson*, a layered file system that in case of an illegitimate file system access provides data for post-breach analysis to assess the severity of the breach and to trace the intruder's actions. *drWatson*, when mounted on top of any concrete file system, works by logging all the operations along with their date time stamps targeted to the below mounted file system.

Categories and Subject Descriptors

D.4 [**Operating Systems**]: File Systems Management ;; D.4.6 [**Operating Systems**]: Security and Protection—*Access controls, Information flow controls, Security kernels*

General Terms

Design, Experimentation, Human Factors, Security

Keywords

file system; security breach; post-breach analysis; Dr.Watson

1. INTRODUCTION

Computer security encompasses concepts and methods for protecting sensitive resources in computer systems. Computer security starts from the policies that regulate access to protected resources. In technology, the focus is on mechanisms for enforcing these policies. A common motivation to invest in information security is to safeguard the confidential as well as the personal information. Over time, information security systems have evolved significantly. Policies are brought into action at various levels across all domains. However, even due to existence and continuous development of strong & reliable security mechanisms for all levels, security issues still worry administrators. The fact is that counter policies and techniques also evolve side by side to overcome new security blockages. Thus, it is mandatory to invest in security systems but equally is important to anticipate a breach and be prepared to retaliate. Taking regular backups can be one of the preparations for a breach while as the general retaliation would be to assess the severity of the breach, to trace the intruder's actions and to possibly hunt down the culprit.

As part of the preparation for the retaliation, the system needs to keep track of every possible access, modification, etc. done to the system resources. The persistent data residing in a file system in the form of hierarchy of files and folders is more vulnerable to breach than some transient data that resides in memory as the prey is always available to predator. Unfortunately, file systems are inherently incapable to help security systems during postmortem. As an example, all file systems maintain only 3 date time stamps per file or folder: Last Modified, Last Accessed and Creation date time stamps (together called as MAC DTS). This means during postmortem, the file system can't answer following questions: 1) How many times was the file/folder accessed or modified? 2) When was it accessed/modified for the first time? 3) What part of the file was accessed/modified every time? 4) What was the pattern of browsing? The answer to these questions will definitely not locate the intruder but will certainly help in assessing the damage and speculating the intention of intrusion.

We propose a layered file system, namely *drWatson*, to enhance the capability of file systems to help in post-breach analysis. *drWatson* when mounted on top of any file system, logs all the operations along-with their date time stamps targeted to it. During postmortem this log provides enough information to assess the damage and to speculate the intentions of an intruder.

2. RELATED WORK

Barik et al. [2] argued that MAC DTS procedures of popular operating systems are inherently flawed and were created only for the sake of convenience and not necessarily keeping in mind the security and digital forensic aspects. They proposed a solution in context of ext2 file system by using specific data structures to preserve the Authentic date and time stamps (ADTS). Similarly, Das et al. [3] and Ansari et al. [1] proposed preserving ADTS in Linux kernel 2.6.x using Linux Loadable Kernel Module (LKM) that can be applied to all file systems. However, all these solutions ([2] [3] & [1]) aim at preserving ADTS but not documenting every change in MAC DTS and the cause of change. Moreover, they don't document the type of access and amount of access.

CCS'13, November 4–8, 2013, Berlin, Germany.

http://dx.doi.org/10.1145/2508859.2512522

In contrast, we propose documenting every relevant information related to any type of access to any file or folder that will help in post-breach analysis. Furthermore, we propose using file system layering that has been used for many purposes (like monitoring, data transformation, operation transformation, size changing, and fan-out [8]) instead of using any file system specific data structures or file system source instrumentation.

3. THE DESIGN

The design of *drWatson* is motivated by many goals which include:

1. The solution should be applicable to all file systems i.e. no source modification and no data structures specific to any particular file system should be used.
2. The solution should log every operation targeted to the below mounted file system along with the type of operation, pre- and post-operation DTS, offset & amount of data read/written, and effective UID.
3. The solution should hide the log from normal listing and should deny access to it.
4. The solution should be portable across all OSes.
5. The solution should not add significant performance overhead to the file system operations.

In order to mitigate our goals, we designed *drWatson* as a layered file system so that it is applicable to all file systems. The capability of below mounted file system is augmented by the layered file system as the operations can be both pre- and post-processed while not modifying the source. Therefore, during pre- and post-processing, *drWatson* documents all parameters of the operation in a log hidden from user applications. Also, the layered file system is able to trap every file system operation.

Moreover, in order to be portable across all OSes it should be layered as a user process and not as some kernel module. Also, in order to be light-weight a well developed and a highly optimized framework should be used. Therefore, *drWatson* was developed using FUSE framework [7]. Figure 1 shows the interaction between various OS components and *drWatson* using a FUSE framework.

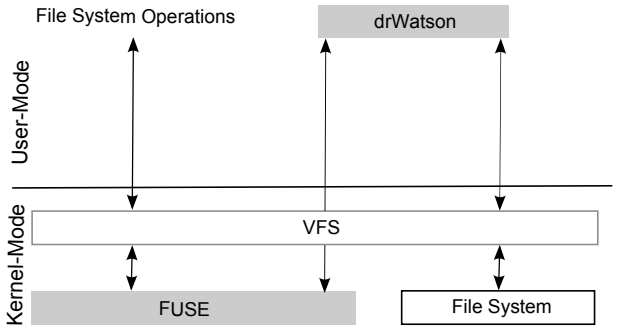

Figure 1: Interaction between OS components & *drWatson*

4. IMPLEMENTATION DETAILS

drWatson is implemented as a layered file system using FUSE framework. FUSE is an acronym for Filesystem in UserSpacE and is used to develop full-fledged file systems and to extend existing file systems [7]. As such, while maintaining the reliability of kernel, it allows access to facilities (like C library) which are lacking in kernel development; the one that is exploited by *drWatson* is usage of simple file I/O functions of C library to create and maintain the log.

The figure 2 shows that every file system operation can be

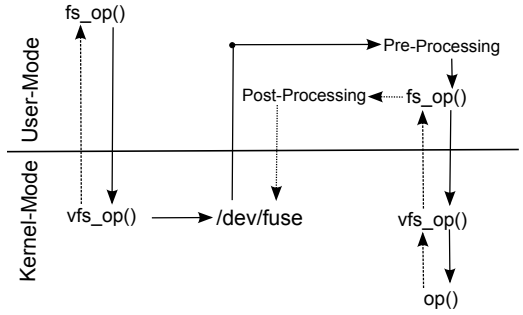

Figure 2: Pre- & post-processing performed by *drWatson*

pre- and post-processed in user space by *drWatson* to reflect the desired operation. Also, *drWatson* readdir() implements the logic to restrict listing of the log file.

The FUSE framework contains a null-pass virtual file system, fusexmp, which passes all the file system operations to below mounted file system without any modification. *drWatson* is implemented by overriding the various procedures of fusexmp. However, fusexmp doesn't support overlay mounting and as such any mount point passed to fusexmp during mounting, mounts the root directory (/) on that mount point. Having said that, a little modification to fusexmp gets it overlaid on the specified mount point, which can be a simple directory or a mount point of some other file system. It has two benefits; 1) it adds to transparency of *drWatson* as no path change is required by applications accessing that volume, and 2) it leaves no path to access native file system without the intervention of *drWatson*.

Furthermore, FUSE framework has been ported to almost all platforms including Windows [4].

drWatson in specific and FUSE file systems in general, incur performance overhead as kernel boundary is crossed to process the call. In addition to multiple context switching, multiple process switching and data copying during call processing also add overhead [6]. However, the benefits of development-ease, reliable environment and portable file system outweigh the drawbacks.

5. WORKING

The working of *drWatson* is simple. Whenever a file system syscall is invoked on any file system having *drWatson* layered over it, *drWatson* pre-processes the call by opening the log, followed by documenting the type of operation, name of target file/folder, effective UID, current MAC DTS of the target file/folder and in case of a read/write operation the offset at which read/write is supposed to take place and finally the amount of data read/written. This is followed by passing the operation to the actual file system. Finally, after

the call returns, *drWatson* post-processes the call to document the new MAC DTS of the target file/folder. Algorithm 1 summarizes the working of *drWatson*.

Algorithm 1 Algorithm for pre- and post-processing of syscalls

Input: $fsOperation$, $eUID$
1: **open**(log)
2: **append**(log) $\leftarrow fsOperation.Type$
3: **append**(log) $\leftarrow fsOperation.filename$
4: **append**(log) $\leftarrow fsOperation.eUID$
5: **append**(log) $\leftarrow fsOperation.filename.DTS$
6: **if** $fsOperation.Type$ = READ/WRITE **then**
7: **append**(log) $\leftarrow fsOperation.offset$
8: **append**(log) $\leftarrow fsOperation.count$
9: **end if**
10: **if** $fsOperation.filename \neq log$ **then**
11: **call**($fsOperation$)
12: **end if**
13: **append**(log) $\leftarrow fsOperation.filename.DTS$
14: **close**(log)

6. EXPERIMENT & RESULTS

We employed Filebench [5] to exercise ext2 file system with *drWatson* layered over it. Filebench is a file system and storage benchmark that allows to generate a large variety of workloads. The experiment was intended to check the amount of information documented by *drWatson* and to test the implementation for flaws. The results indicate that *drWatson* is successfully able to document every piece of information relevant to post-breach analysis.

7. CONCLUSION

In this paper we argued that file systems don't store enough information relevant to post-breach analysis as they maintain only 3 date time stamps viz. last Modified, last Accessed & Creation (MAC DTS). We found that current proposals are only able to preserve the authentic date time stamps (ADTS).

We proposed preserving MAC DTS (along with other relevant information) after every file system operation so that enough information is available during postmortem of the file system. However, in order to have the proposal applicable to all file systems and portable across all platforms, we proposed that the solution be developed as a layered file system in userspace.

Based on this idea, we developed a layered file system, namely *drWatson*, using FUSE framework that documents every information of every file system operation relevant to the post-breach analysis. On exercising the ext2 file system with Filebench benchmark, we found that *drWatson* is able to log enough information required during postmortem in case of a breach.

8. REFERENCES

[1] A. Ansari, A. Chattopadhayay, and S. Das. A kernel level vfs logger for building efficient file system intrusion detection system. In *Computer and Network Technology (ICCNT), 2010 Second International Conference on*, pages 273–279. IEEE, 2010.

[2] M. S. Barik, G. Gupta, S. Sinha, A. Mishra, and C. Mazumdar. An efficient technique for enhancing forensic capabilities of ext2 file system. *digital investigation*, 4:55–61, 2007.

[3] S. Das, A. Chattopadhayay, D. K. Kalyani, and M. Saha. File-system intrusion detection by preserving mac dts: a loadable kernel module based approach for linux kernel 2.6. x. *CSIIRW*, 9:1–6, 2009.

[4] E. Driscoll, J. Beavers, and H. Tokuda. Fuse-nt: Userspace file systems for windows nt. 2008.

[5] R. McDougall and J. Mauro. Filebench, 2005.

[6] A. Rajgarhia and A. Gehani. Performance and extension of user space file systems. In *Proceedings of the 2010 ACM Symposium on Applied Computing*, SAC '10, pages 206–213, New York, NY, USA, 2010. ACM.

[7] M. Szeredi. Filesystem in userspace, 2005.

[8] E. Zadok, R. Iyer, N. Joukov, G. Sivathanu, and C. P. Wright. On incremental file system development. *Transactions on Storage*, 2:161–196, May 2006.