From TCT to the Adaptability of Computer Forensic Tools

Haohao Zhai Wenchang Shi Bin Liang Liang Wan

School of Information，Renmin University of China，Beijing 100872，China

Key Lab of Data Engineering and Knowledge Engineering of Ministry of Education（Renmin University of China），Beijing 100872，China

Abstract In practical computer forensic investigation, investigators often run into trouble due to forensic tools' not performing well. The cause that leads to these situations may be the change in computer hardware or software after those tools' being put into the market. We call this phenomenon as that the forensic tools are not adaptive well to the real-world environment. Taking a well-known open source forensic tool, i.e. the TCT, as a prototype, this paper conducts research on the adaptability of computer forensic tools. The goal of the work is to devise methods of finding out the very reason when a tool does not work well on the field, and figuring out solutions to the situation. Experiments are carried out, which involve 8 releases of the Linux operating system, with 2 configurations for each release, totally covering more than 10,000 lines of source code. Reasons that lead to problems in the experiments are analyzed, and solutions are proposed to solve the problems.

Keywords Computer Forensics, Forensic Tools, Adaptability, Solutions

I. Introduction

With the deep penetration of computer networks into the society, computer devices are widely involved in criminal activities. Computer forensic investigations play more and more important role in handling law cases. Computer forensic tools are critical to computer forensic investigations. Most of these tools are in some sense platform specific, in that their functions may need support from the platform on which they operate. Quick development of computer hardware and software causes computer forensic tools unable to keep abreast with changes in platforms. Consequently, computer forensic tools may not work well on targeted platforms, which make investigators run into troubles. We refer to this phenomenon as an adaptability issue. That is to say, computer forensic tools can not adapt to a targeted investigation environment. Problems confront an investigator may be that a software tool can not be installed, can not run as expected or produces incorrect results [1]. Generally speaking, computer forensic tools currently in practical used may be roughly classified into two types, i.e. hardware tools and software tools. Common functions implemented by hardware tools include copying, erasure, conversion of disk interfaces, etc. One representative of this kind of tools is products from Logicube, an American company with authorization of the U.S. government. Common functions implemented by software tools include live investigation, key words searching, anti-deletion of files, evidence analysis, etc. Typical tools of this kind include Encase, FTK, Winhex, TCT, and so on. The Encase tool is a product of Guidance Software, an American company with authorizations of judicial departments. The FTK, or Forensics ToolKit, is a product of AccessData, an American company with rich experiences of police officers. The Winhex tool is a product of X-ways, a German company dedicated to file system forensics. The TCT, or The Coroner's Toolkit, is an open source tool developed by D. Farmer and W. Venema [2]. Among diverse computer forensic tools, we choose the TCT as a representative prototype to study adaptability of software forensic tools. In this paper, research is carried out through experiments, which involve 8 releases of the Linux operating system, each of which includes 2 configurations, totally covering more than 10,000 lines of source code. Through experiments and analysis, we try our best to find the very reason when a tool does not work well, and then propose and implement solutions to the adaptability problems.

The rest of this paper is organized as follows. Section II demonstrates adaptability problems that investigators may face in using forensic tools. Section III analyzes the very reasons that cause the problems.

Table 1. Main Functions of TCT Tool Programs

Function	Program	description
Data Gathering	grave-robber	Automatically collecting static and dynamic data from the system
Time Analysis	mactime	Collecting and handling MAC time attributes of files
Low-Level File System Utilities	unrm	Accessing a disk block by its disk number
	icat	Accessing file content by its inode number
	ils	Accessing file attributes by its inode number
File Reconstruction	lazarus	Reconstructing the structure of deleted file content
Low-Level Memory Utilities	pcat	Dumping the memory of a running process
Other C Program	file	Determining file type
	major_minor	Listing device major and minor number
	md5	Computing the md5 checksum for each file
	lastcomm	Showing last commands executed in reverse order (applied to BSD operating system)

Section IV proposes solutions to the problems. Section V discusses implementation of our solutions and result of test. Finally, Section VI briefly reviews related work and concludes the paper.

II. The Problems

The chosen prototype for our research, the TCT, is a computer forensic tool developed by D. Farmer and W. Venema. The first version of it was released in 2000. One of its distinguished features is that it supports live forensic investigation on a running computer [3]. So far, 19 versions of the TCT tool have been published, of which the latest one is TCT-1.19. In this paper, if not specifically mentioned, the prototype TCT tool is of this version. Operating systems that support the TCT tool include FreeBSD, OpenBSD, BSD/OS, SunOS, Linux, etc., and file systems that support it include FFS, Ext2, Ext3, etc. To install this tool on a system, Perl 5.004 or later version and a C complier are needed. The TCT tool consists mainly of 11 programs, whose functions are illustrated in table 1.

Using the TCT to do forensic analysis to Ubuntu9.04 (with Ext3 file system), we found that program icat and ils produce incorrect results, in which output data is not consistent with related contents of a designated inode. Testing these two programs on a clean Ubuntu 9.04 operating system and comparing results that they produced with low-level file systems data, we found that incorrect output results are caused by the TCT itself, which means that the TCT is not adaptive well to the Ubuntu 9.04 operating system.

During forensic investigation, the TCT should have performed well on the Ubuntu 9.04 operating system

Table 2. Experiment platforms and Testing Results of the TCT tool

OS Release (Kernel Version)	Release Date	Perl Version	File System	Testing Results of TCT Tool
Ubuntu9.04 (2.6.28)	2009.04	5.10.0	Ext3	Installable. Running results of icat and ils programs incorrect. Lazarus program not able to run.
			Ext4	
Ubuntu9.10 (2.6.31)	2009.10	5.10.0	Ext3	Failed to be installed. Errors appear in file /usr/include/ linux/ext2_fs.h when compiling
			Ext4	
Ubuntu10.04 (2.6.32)	2010.04	5.10.1	Ext3	Failed to be installed. Errors appear in file /usr/include/ linux/ext2_fs.h when compiling
			Ext4	
Ubuntu10.10 (2.6.35)	2010.10	5.10.1	Ext3	Failed to be installed. Errors appear in file /usr/include/ linux/ext2_fs.h when compiling.
			Ext4	
Ubuntu11.04 (2.6.38)	2011.04	5.10.1	Ext3	Failed to be installed. Errors appear in file /usr/include/ linux/ext2_fs.h when compiling.
			Ext4	
Fedora13 (2.6.32)	2010.04	5.10.1	Ext3	Failed to be installed. Errors appear in file/usr/include/ linux/ext2_fs.h when compiling.
			Ext4	
Fedora14 (2.6.35)	2010.10	5.10.1	Ext3	Failed to be installed. Errors appear in file /usr/include/ linux/ext2_fs.h when compiling.
			Ext4	
Fedora15 (2.6.38)	2011.04	5.10.1	Ext3	Failed to be installed. Errors appear in file /usr/include/ linux/ext2_fs.h when compiling.
			Ext4	

with Ext3 file system, but, in fact, it can not run correctly. In order to determine whether such kind of problems exist in other environments, we take the commonly used Linux operating system with Ext3 and Ext4 file system as the basic platform. In details, we choose Ubuntu release 9.04, 9.10, 10.04, 10.10 and 11.04 of the Debian series and Fedora release 13, 14 and 15 of the Red Hat series, with 2 configurations for each release, totally setting up 16 experiment platforms and covering more than 10,000 lines of source code. All programs of the TCT are tested on each platform. As a Perl compiler is necessary to install the TCT tool, different versions of Perl complier is tested on every platform. The test results are shown in table 2.

As is known from table 2, the TCT tool has adaptability problems on every platform. Specific analysis to reasons of the problems will be presented below.

III. Analyzing Reasons of the Problems

3.1 Preliminary Analysis

Generally speaking, when a forensic tool meets adaptability problems, the system on which the tool runs may issue some kind of messages. According to these messages, problem types can be judged, analysis scopes can be narrowed, analysis targets can be located and working procedures can be determined. Therefore, analysis in this phase plays a very important role in the whole process of analysis.

Experiment platforms and problems can be classified into two classes according to the testing results of the TCT tool. The first class is experiment platform with Ubuntu 9.04 operating system. The TCT meets problems in such kind of environment when it runs. The second class is experiment platform with Ubuntu 9.10-11.04 operating system or Fedora 13-15 operating system. The TCT meets problems in such kind of environment when it is to be installed.

On the first class of Experiment platforms, the TCT meets problems when program icat, ils and lazarus are run. The programs of icat and ils are designed in C language. When these programs run, they bypass the file system layer and read related contents of files through inode number, and they are closely related to low-level structure of file systems. To the case that programs produce incorrect running results on the first class of experiment platform, we have to inspect the internal working process of the programs and the structure of the low-level file system. The program of lazarus is designed in Perl language. It presents unstructured data to users in a viewable and manipulatable form. When the program runs on the first class of experiment platform, prompted messages say that the "$*" variable in source code of lazarus is unsupported. In this case, we have to inspect the Perl language and its version update state.

The TCT tool can not be successfully installed on the second class of experiment platform. The /usr/include/linux/ext2_fs.h file is said to be incorrect in compilation. This file in Linux operating system is used to indicate basic structure of super block, group descriptor and inode of the Ext2 file system. File system related programs in the TCT tool use this file to analyze data. To the case that the TCT meets installation problems on the second class of experiment platforms, we have to inspect the update state of the file in the operating systems.

Any computer forensic tools must satisfy the conditions of being properly installed and correctly running. The TCT can not be installed on the second class of experiment platforms, so the installation problem must be solved in the first place. Problems of running can be dealt with after it is properly installed. Analysis to the running problems that the TCT meets on the first class of experiment platform can be performed accordingly.

3.2 Analysis of Installation Problems

When a software tool is presented in source code, it usually needs to be compiled before being installed,

which is the case of the TCT tool. If problems occur when a tool is being compiled, specific compilation status messages are generated, which always point to version state of the compiler. Installing a software tool often involves a large number of files and data. The update state of configuration files and system environments must be taken into account when installation of a tool meets problems.

When the TCT tool is to be installed on different experiment platforms of the second class, problems that occur in the compiling period are the same. All errors point to the /usr/include/ linux/ext2_fs.h file in the system and prompt messages are specific. Without loss of generality, we take Ubuntu 9.10 as an example to do analysis in three steps.

First, compare file ext2_fs.h in the Ubuntu 9.04 operating system with the one in the Ubuntu 9.10 operating system. We found that Ubuntu 9.10 had updated this file. Second, copy header file ext2_fs.h in Ubuntu 9.04 to the TCT tool package and modify related source code. Problems that occur in the compiling period can be solved in this way. Third, test every functional component of the modified TCT tool one by one on the second class of experiment platforms. Test results show that the TCT can be successfully installed on all of the experiment platforms, but icat and ils produce incorrect results, while lazarus can not run.

Consequently, it can be seen that the TCT tool meets the same problems when running on both the first class and second class of experiment platforms. Detailed analysis to reasons for running errors of the icat, ils and lazarus program will be presented below.

3.3 Analysis of Running Problems

There are many reasons for problems that occur in the running of a program. A common one is that changes in lower level system data structures cause a program to generate incorrect results. To solve this kind of problems, it is required to understand the internal working process of the program and changes in lower level system environments. Another reason is that the update of an underlining system or its configuration files incurs the malfunction of a program. To solve this kind of problems, update states of the underlining system and important configuration files must be taken into account.

Running problems that the TCT tool meets in our experiments focus on the icat, ils and lazarus program. These problems can be classified into two types. One is of file-system related problems that cause the icat and ils program to generate incorrect running results. The other is of compilation-environment related problems that make the lazarus program unable to run.

3.3.1 Problems Related to File Systems

What the icat and ils program do are to access file contents and attributes through inode number. Source code analysis shows that their working process may be divided into three steps. First, open a disk block, read and analyze information of the super block. Second, search, read and analyze an inode. Third, output the results. Let’s analyze reasons for incorrect running results of the icat and ils program step by step.

(1)Read and Analyze the Super Block

The super block of a Ext3 or Ext4 file system is used to record basic information of the whole file system, including number of inode, number of data block, block size, block number of each block group, number of inode of each block group, etc.

After icat or ils open a disk block, data can be read directly from location with offset of byte 1024. They read the size of the ext2_super_block structure and analyze data according to that structure. Important information such as number of inodes, number of data blocks, size of a data block, offset of the number 1 block group descriptor, number of block number can be obtained.

To verify whether the TCT makes errors in this period, we compare the structure of the super block on experiment platforms. First, use the blkcat program of the TSK (The Sleuth Kit) to read directly number 0 data

block in the disk. Second, use the fsstat program to provide basic information of the file system. At last, compare data of the number 0 data block with offset of byte 1024 with super block information output from fsstat and the ext2_super_block structure, and seek changing data information.

By analysis, we found that compared with the super block of earlier Ext2 file system, the super block of Ext3 or Ext4 file system on the experiment platforms does not experience any change, which is put in the first data block with offset of byte 1024. The ext2_super_block structure in the ext2_fs.h file of the operating system is used to describe basic structure information of the Ext2 file system's super block. This structure is still applicable on experiment platforms. Thus, this step works correctly.

(2)Locate, Read and Analyze Inode

After icat and ils obtain basic information of file systems, they calculate the storage location of an inode, and read and analyze it. The working process can be divided into two steps.

Step 1, calculate the address of a block group descriptor according to super block information and size of the ext2_group_desc structure, and read and analyze it.

The block group descriptor is used to record basic information of a block group, including block bitmap address, inode bitmap address, inode address, the number of free blocks, the number of free inodes and the number of directory entries. Program icat and ils locate and analyze the block group descriptor by utilizing the ext2_group_desc structure in file ext2_fs.h.

In order to verify whether anything in the underlying structure of the block group descriptor has been changed, we use the blkcat program to read the number 1 data block of the disk directly, and compare block group information given by the fsstat program with that of the ext2_group_desc structure.

By analysis, we found that the block group descriptor of the Ext3 or Ext4 file systems on the experiment platforms has no change compared with the earlier Ext2 file system. The sizes are both 32 bytes, so the ext2_group_desc structure is still applicable on the platform. In this step, icat and ils work correctly.

Step 2, calculate the address of an inode according to the size of the block group descriptor and the ext2_inode structure, and read and analyze it.

Inodes are used to store attribute information of files, including file size, timestamp, data block address, etc. The ext2_inode structure in the ext2_fs.h file of the operating system is used to describe basic information of an inode in the Ext2 file system. Program icat and ils calculate and analyze the inode by utilizing the ext2_inode structure.

In order to verify whether anything in the underlying structure of an inode has been changed, firstly, we search the inode table of the first block group according to block group information and use the blkcat program give output. Secondly, we output information of the root directory (inode number 2) of the file system through the stat program. At last, compare the ext2_inode structure and stat's output with the output from the blkcat program, we seek address of inode number 2 and update state of the underlying structure of an inode.

Compared with the Ext2 file system, the inode structure of the Ext3 file system on the experiment platforms has no change [4]. But for compatibility with new version of file systems, the size of an inode is enlarged to 256 bytes from 128 bytes, of which the higher order 128 bytes stores file's inode information and the lower order 128 bytes don't store any data. Compared with the Ext2 file system, the inode structure of the Ext4 file system on the experiment platforms has a big change in that creation time of a file is added, data block address is expressed with extent, and the size of an inode is enlarged to 256 bytes. The ext2_inode structure is only 128 bytes in size and does not support the extent structure, which causes the icat and ils program unable to correctly locate an inode in the Ext3 or Ext4 file systems, and unable to explain an inode correctly in the Ext4 file system.

(3)Output Results

What should be done in this step is to output data obtained in the previous two steps to a terminal, which

does not involve any lower level data. Therefore, icat and ils have no trouble in running in this step.

3.3.2 Problems Related to Compiling Environments

Those programs of the TCT implemented in Perl language need the support of a Perl compiler of 5.004 or later versions. Perl compilers used on our experiment platforms satisfy all version requirements, but there still exist some problems.

By analyzing Perl compilers of 5.10.0 or later versions, we found that some improvements to the language had been made. With respect to compilation, support to the "$*" variable has been removed. In version before 5.10.0, the function of the"$*" variable is to make multiline matching of character strings. When it is set to 1, the multiline matching function is enabled. When it is not set or set to 0, multiline matching function is disabled. Cancelation of the variable causes malfunction of the lazarus program.

IV. Solutions to the Problems

Solutions to the problems have to be reasonably put forward according to reasons that cause the problems. They should be able to guide people to modify source codes of relevant programs if necessary.

Reasons that cause the TCT tool unable to work successfully on the experiment platforms are as follows. First, it depends on certain files of the operating system. Update of these files causes it unable to be installed successfully. Second, it uses system header files to determine the size of an inode. Enlarging the size of an inode in the underlying environments causes the icat and ils program unable to locate an inode correctly. Third, because the inode is analyzed according to system header files, change of the inode structure in the Ext4 file systems causes icat and ils unable to analyze an inode correctly. Fourth, because programs implemented in Perl language can only be handled by Perl compilers of earlier versions, later versions of Perl compilers on the experiment platforms cause the lazarus program run into compilation problems.

To the above four kinds of problems that the TCT tool meets on our experiment platforms, we put forward the solutions as follows.

(1)Add a new ext_fs.h header file to solve the problems that the TCT meets on the second class of platforms. In order to make the TCT more independent of the systems, the new header file ext_fs.h should be added to the TCT package and replace the former ext2_fs.h header file in the system. The ext_fs.h header file should at least include the structures of super block, data group descriptor and inode. In addition, the structure of the inode should be designed in such a common way that it can be used in both the Ext3 and Ext4 file system.

(2)Change the way to calculate the size of an inode so as to solve the problem of not being able to locate an inode correctly. The TCT tool utilizes the size of the ext2_inode data structure defined in the ext2_fs.h file to calculate the size of the inode, which is of 128 bytes. This is inconsistent with the case on the experiment platforms where an inode is 256 bytes in size. As we know, the attribute of inode_size in the super block correctly records the size of an inode of the file system, thus we change the method of the TCT to use data in the super block to calculate of the size of an inode.

(3)Add a new method to analyze inodes so as to solve the problem of not being able to correctly analyze an inode in the Ext4 file system. Compared with the Ext2 file system, an obvious change to the Ext4 file system is that "extent" is used to express the address of a data block [6]. An extent is used to express a piece of contiguous address space. In the Ext4 file system, when the number of extents is not greater than 4, all extents are stored in the inode and the data blocks to which they point store file contents. When the number of extents is greater than 4, only one extent is stored in the inode and the data block to which it points still stores file extents. Different contents of data blocks pointed to by extents may mean different data structures of extents. In table 3, structure A shows that the data block pointed to by an extent stores file contents, and structure B shows that the data block pointed to by an extent also stores file extents. Therefore, methods that analyzes inodes of the Ext4 file system

should be added to the TCT, so that the TCT can be applied to the Ext4 file system.

Table 3. Data structure of extent

Offset(byte)	0	1	2	3	4	5	6	7	8	9	A	B
Structure A	Logic number				The number of this extent		First block address (High)		First block address(Low)			
Structure B	Logic number				First block address (Low)				First block address (High)		Reserved bits	

(4)As to the problem caused by the Perl compilation environments, the "$*" variable for multiline matching in source code of the lazarus program can be substituted with other ways of expression.

V. Implementation and Test

5.1 Code Modification

Based on solutions mentioned above, we modify the source code of the TCT software package. The main points of our work are as follows.

(1)Add the new ext_fs.h header file to the tct-1.19/src/fstool directory. The header file includes structure ext_super_block, ext_group_desc, ext_inode and ext4_extent_block. The union type is used to express data block address in the ext_inode data structure, which can express both the Ext2/3 index structure and the Ext4 extent structure.

(2)Modify the tct-1.19/src/fstools/fs_tools.h file. Modification is made as follows. The /usr/include/linux/ext2_fs.h header file of the Linux operating system is substituted with the new ext_fs.h header file. The FS_INODE structure is modified and the extent structure of the Ext4 file system is added.

(3)Modify the tct-1.19/src/fstools/ext2fs.c file. Modification is made as follows. The method in the ext2fs_dinode_lookup function that determines the size of an inode is modified to use the inode_size attribute of the super block. Methods to identify file system types and to analyze the Ext4 extent are added to the ext2fs_copy_inode function.

(4)Substitute ext2 in all files with ext.

(5)Remove all statements of "$*=1" and "$*=0" in source code file of the lazarus program, and add "~m" before each regular expression that sits in between those two statements.

5.2 Testing and Results

We do the testing to the modified TCT tool on the 16 platforms mentioned above, which focuses on analysis of inodes in the Ext4 file system. As the experiment results show, the modified TCT tool can be properly installed at all the platforms and all the programs run correctly.

VI. Related work and Conclusions

There has been work to deal with problems of computer forensic tools for a long time. Usually, it is the responsibility of the developers to maintain or upgrade forensic tools when they fail to function correctly. What the developers do to this end is to update program codes, to re-analyze lower level data structures, or to add new functions [7]. Meanwhile, organizations exist that devote themselves to solve problems that forensic tools may incur by testing. For example, as early as in 2003, the CFTT, or Computer Forensics Tool Testing, began to launch

program to test computer forensic tools [8]. B. Carrier who is the developer of TSK, or The Sleuth Kit, once put forward a concept to deduce possible problems of forensics tools by using abstraction layer [9]. However, to solve the adaptability problems of computer forensic tools, there is still a long way to go. In this paper, we explore a way to shoot the target.

Taking the open source TCT tool as a representative, we anatomize the adaptability problems of software forensic tools, figure out the very reasons that cause the problems, and establish solutions to the problem. By conclusion, a general way to handle adaptability problems of software forensic tools may be summarized as the following two parts. First, perform preliminary analysis according to error status messages. While analyzing, identify the type of the problem, narrow the scope of the problem space, and locate the target of the problem. In addition, determine the basic procedure for analysis. Second, perform detailed analysis according to the type of the problem. The key issues that are worth mentioning are whether update of an operating system leads to changes of important system elements such as system configuration files or lower level data structures, and whether update of a compiler results in changes of language-related components.

Our experiments show that the above concept can guide us to solve the adaptability problems corresponding to the TCT tool effectively. Considering the common characteristics of software forensic tools, we believe that the concept derived from our work is of important value in tackling adaptability problems related to other software forensic tools.

Acknowledgements

The authors would like to thank the anonymous reviewers for their insightful comments that helped improve the presentation of this paper. The work of the paper was supported in part by the National Natural Science Foundation of China (61070192, 91018008, 61170240), Natural Science Foundation of Beijing (4122041), the Important National Science & Technology Specific Projects of China ("HGJ" Projects, Grant No. 2010ZX01042-001-002-002), and National High-Tech Research Development Program of China (2007AA01Z414).

References

[1] S. L. Garfinkel. Digital Forensics Research: The Next 10 Years. Digital Investigation, 7(2010), S64-S73, 2011.

[2] D. Farmer, W. Venema. The Coroner's Toolkit. http://www. porcupine.org/forensics/tct.html.

[3] D. Farmer, W. Venema. Forensics Discovery. Boston: Addison- Wesley Professional, 2005.

[4] K. D. Fairbanks, C. P. Lee, H. L. Owen III. Forensic Implications of EXT4. Proceedings of the Sixth Annual Workshop on Cyber Security and Information Intelligence Research, CSIIRW '10. ACM, New York, NY, USA, 2010.

[5] S. Tweedie, T. Y Ts'o. Planned Extensions to the Linux Ext2/Ext3 Filesystem. USENIX Technical Conference, Montery CA. 2002.

[6] A. Mathur, M. Cao, S. Bhattacharya, A. Dilger, A. Tomas, L. Vivier. The new ext4 filesystem: current status and future plans. Proceedings of the Linux Symposium. Ottawa, ON,CA, 2007.

[7] B. Carrier. The Sleuthkit : History. http://www.sleuthkit.org/sleuthkit /history.php.

[8] NIST. Computer Forensic Tool Testing (CFTT) . http://www. cftt.nist.gov.

[9] B. Carrier. Defining Digital Forensic Examination and Analysis Tools Using Abstraction Layers. International Journal of Digital Evidence,1(4),Winter 2003.