Effective Visualization of File System Access-Control

Alexander Heitzmann[1], Bernardo Palazzi[1,2,3], Charalampos Papamanthou[1], and Roberto Tamassia[1]

[1] Brown University, Department of Computer Science, Providence, RI, USA
[2] Roma TRE University, Rome, Italy
[3] ISCOM Italian Ministry of Communications, Rome, Italy
{aheitzma,bernardo,cpap,rt}@cs.brown.edu

Abstract. In this paper, we present a visual representation of access control permissions in a standard hierarchical file system. Our visualization of file permissions leverages treemaps, a popular graphical representation of hierarchical data. In particular, we present a visualization of access control for the NTFS file system that can help a non-expert user understand and manipulate file system permissions in a simple and effective way. While our examples are based on NTFS, our approach can be used for many other hierarchical file systems as well.

1 Introduction

The access control model employed by current-generation file systems, such as Microsoft Windows XP and Vista, is rather complex and often insufficiently documented. In a large file system with multiple users, it is rather tricky to understand which users/groups can access which files and with which permissions. Also, the effect of simple operations (such as copy and move) on the permissions of a file are difficult to anticipate and sometimes unintuitive. For example, consider a Windows user who changes the permissions of a certain file to make it not readable by others and later moves this file to another folder where the read permission is inherited from the parent folder. The user is unlikely to realize that after the move, the file is no longer protected. This is due to the fact that in a Windows NTFS file system, there are three types of permissions associated with a file: the *local* permissions for the file, the *inherited* permissions derived from the permissions of the parent folder, and the *effective* permissions, obtained as the union of the local permissions and the inherited permissions.

Inherited permissions have many advantages and have been adopted by several file systems.

Also, inherited permissions and other features of access control mechanisms can make answering questions such as "What group has access to which files during what time duration?" or "If I implement this policy, what conflicts this result?" very difficult [8].

Understanding file permissions and setting them to achieve desired file sharing and protection goals can be a daunting task for non-expert users and is non-trivial even for experts. A tool that helps users to understand how access-control permissions are determined and the effect of file system operations on file permissions would be extremely useful for both regular users and administrators.

We believe that an effective way to overcome difficulties of understanding file permissions is through visualization. Therefore, in this paper, we present our preliminary design of a visualization tool that displays access-control information in a way that is easily understandable and helps the user set the correct permissions to achieve file sharing and protection goals. Our visualization tool uses treemaps [6], a popular graphical representation of hierarchical structures based on a recursive decomposition of rectangles into sub-rectangles.

In Windows, advanced file system permissions are displayed as a list. Reeder et al. [9] propose using a square matrix to visualize the permissions of a file system and presents an example with changes of groups and users permissions. Montemayor et al. [8] present a solution for access control visualization based on representing the connections between groups, users, and resources, with a graph. The complexity of access control safety and the administrator's difficulty in dealing with it (which makes visualization of access control very important) is analyzed in [5]. The usability of access control systems is discussed in [3]. Some visualization solutions for access-control and file-sharing policies are presented in [10].

Treemaps were introduced in 1991 [6] as a method of representing a complex hierarchy in a compact space. Bladh et al. [1] provide a file system visualization based on treemaps in the 3D space. Interactive ways to explore a file system through visualization are presented in [4]. Stasko [12] gives an evaluation of different compact ways to represent hierarchical structures. The visualization of dynamic hierarchies is presented in [13]. Finally, in 1971, a method using a nested rectangle representation (that resembles treemaps but though not formally defined) to visualize program execution was presented [7].

2 Preliminaries

In this paper, we focus on the *access control list* (ACL) implemented in the Windows NTFS (New Technology File System). NTFS [11] allows to define access control information for each file system object. Using different security policies it is possible to allow or deny access to files and folders for determined users or groups. The file system driver manages all file system requests (i.e., create new files, open existing files, write to files. etc.) as the intermediary between the operating system and the storage device drivers. NTFS ACLs are composed of *access control entries* (ACEs). Each ACE allows or denies specific permissions (i.e., by a user or a group) to or from an object.

Starting with Windows 2000, NTFS allows to dynamically manage permission inheritance. That is, when you create a subfolder or a file in a NTFS folder, the child object not only inherits the parent's permissions but maintains a kind of link with its parent. Furthermore, parent's permissions are stored separately from any local permissions that are directly stored on the child. So for any changes performed on the parent folder, this method allows the child objects to automatically inherit the changes from their parents and to prevent from overwriting all the local permissions.

This approach allows an administrator or a user to manage a hierarchical tree of permissions that matches the directory tree. Since each child inherits permissions *recursively* from its parent. So it is possible to perform changes of permissions with little effort.

The main downside of *dynamic* inheritance is the increase of complexity and the possibility to have conflicting ACEs. The NTFS security module combines the specified permissions (i.e. local and inherited ACEs, allows or explicit denies) and decides whether to grant or deny the access to a user, a group, or other security entities. Microsoft introduced in Windows XP the *effective permissions* tab to help the administrator in the quite tricky task of understanding the effective permission for a user or a group on a specific file system object.

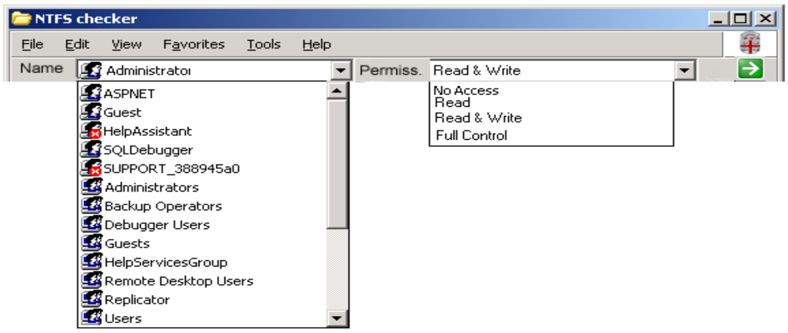
Treemaps (see, e.g., [2, 6]) were introduced in 1991 as a way to represent large hierarchical structures in a compact way. The main idea of creating a treemap can recursively be described as follows: Given a tree T with root r, assign a rectangle A to represent T. Then, for all the subtrees $T_1, T_2, \ldots, T_k$ of r, partition A into k rectangles $A_1, A_2, \ldots, A_k$ and assign $A_1, A_2, \ldots, A_k$ to $T_1, T_2, \ldots, T_k$. This process continues until it reaches the leaves, where it assigns distinct rectangles for every leaf of the tree. Given a tree with n nodes, a treemap can be constructed in $O(n)$ time using a bottom-up traversal.

Several algorithms have been proposed for assigning rectangles to subtrees. The standard method is based on the "slice-and-dice" algorithm, originally introduced in [6], which uses parallel lines to divide the rectangle assigned to a subtree T into smaller areas that correspond to the subtrees of T. It also alternates the direction of the parallel lines (horizontal/vertical) from one level the next, so that the change of levels is displayed. The standard treemap method often gives thin, elongated rectangles. A new method—the "squarified" algorithm— is presented in [2] to generate layouts in which the rectangles approximate squares.

3 Effective Access Control Visualization

In this section, we present the main features of the tool we have designed to assist administrators and users in better understanding and managing access-control of a hierarchical file system. The tool employs treemaps to visualize the file system tree. We use colors to distinguish the permissions of files and folders, and we indicate where a *break* of inheritance occurs with a special border around the relevent node in the treemap. The input to our tool consists of two items:

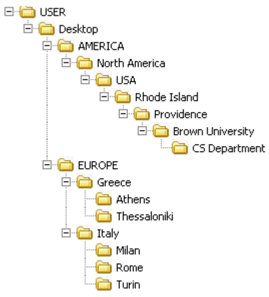
1. The "user" input, which indicates the user or group whose permissions we are interested in. In Figure 1, this is indicated with the label "name".
2. The "baseline" input, which basically indicates a certain combination of permissions upon which the color scheme of our visualization is based. In Figure 1, this is indicated with the label "permission".



Name: Administrator
Permiss. Read & Write

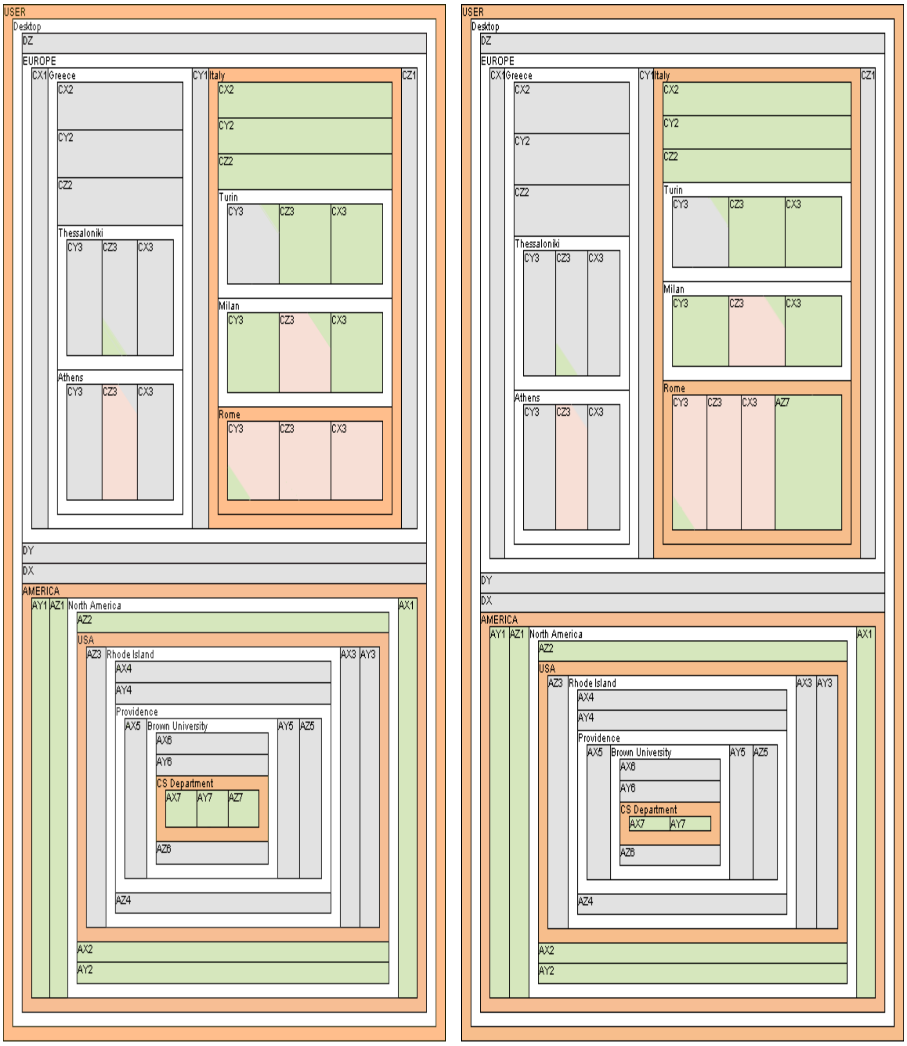
Fig. 1. The user interface of our visualization tool. The main screen consists of the "user" input and the "baseline" input.

In the current design, the baseline"input can take four values, namely the values **no access**, **read**, **read&write**, **full control**. These are sorted in "increasing permission" order. The user can also propose (and insert into the drop-down menu) another combination of permissions (e.g., **read&execute**) and the administrator is responsible for putting the new feature in the correct order (see Figure 1). The visualization tool reads this value and parses the file system tree, building the treemap using the slice-and-dice algorithm. For every file encountered, the associated node in the treemap is painted green, red, or gray, if the file's permissions are weaker (more restricive), stronger (less restrictive), or the same as those specified by the baseline, respectively. The tool could potentially use different shades of the same color to declare intensity of permissions. Finally, the tool draws an orange border around treemap nodes associated with files or folders where inheritance is broken.



folder	N.A.	R	R&W	F.C.	I
USER			✓		
Desktop			✓		
AMERICA		✓			
North America		✓			✓
USA		✓			
Rhode Island			✓		✓
Providence			✓		✓
Brown University			✓		✓
CS Department		✓			
EUROPE			✓		✓
Greece			✓		✓
Thessaloniki			✓		✓
Athens			✓		✓
Italy		✓			
Milan		✓			✓
Rome				✓	
Turin		✓			✓

Fig. 2. The directory tree that we are going to visualize with our tool, as visualized by Windows explorer. Beside the tree, we also show the effective permissions of each folder of the tree. **N.A.** stands for "no access", **R** stands for "read", **R&W** stands for "read and write", **F.C.** stands for "full control" and **I** stands for "inheritance".



(a) (b)

Fig. 3. Treemap (a) shows the access control permissions for user *administrator*. User *administrator* has permission **R** on file **AZ7** because the color is light green, indicating a weaker permission compared with the baseline **R&W**. One can see that these permissions were inherited by *CS Department* because it is the first orange parent folder, indicating a break of inheritance. Treemap (b) illustrates the result of moving file **AZ7** from the *CS Department* folder to the *Rome* folder. The permissions of this file, indicated by the light green color, are preserved after the move. Furthermore, the size of the rectangle associated with the moved file increases to accentuate the move. Also, the color of the top right corner (the inherited permissions) of file **AZ7** is light green because after the move there is no inheritance from the parent.

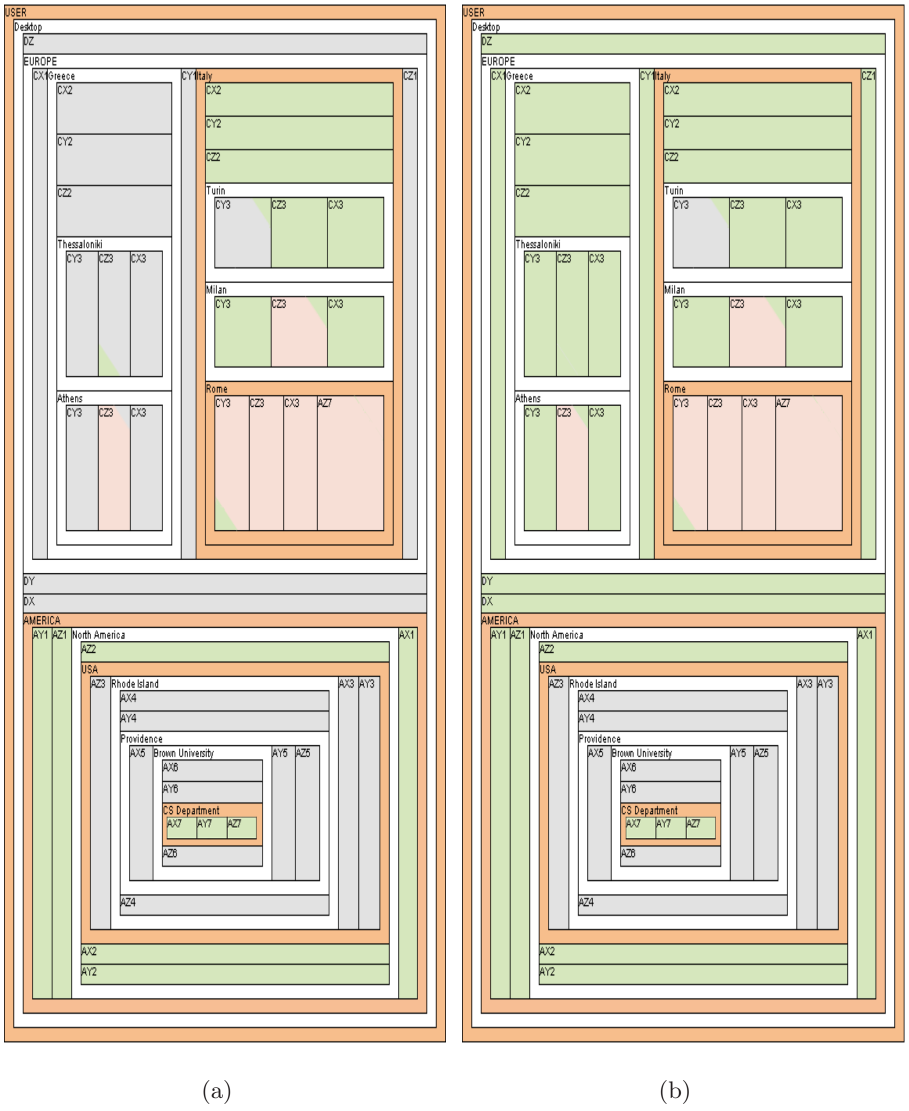


Fig. 4. Treemap (a) shows the result of copying file **AZ7** from the *CS Department* folder to the *Rome* folder. The permissions of the file change, as indicated by the light red color, because of the inheritance from the destination folder. Treemap (b) shows the result of changing the permissions of the *USER* folder from **R&W** to **R**. This change propagates down to descendant files and folders until there is a break of inheritance. Note that file **CZ3** in the *Thessaloniki* folder changes its color from *grey* to *light green* because the local permission (left bottom corner) has a level that is lower than that of the inherited permission (top right corner). It is possible to see the opposite behavior in file **CZ3** in the *Athens* folder, where the inherited permission changes from *grey* to *light green* but the color of the rectangle remains *light red* because the level of the local permission is greater than that of the inherited permission.

We believe that this scheme makes it is easy to gain a general sense of current permissions of the file system as far a certain user is concerned. Furthermore, a more detailed understanding can be achieved simply by exploring the treemap more thoroughly. For example suppose that a file is moved from a folder that has weaker permissions than the baseline to a folder that has stronger permissions than the baseline. The administrator, by using our tool, will be able to notice that difference (since a small green area will appear in a greater red area). There is no longer a need to manually (by exploring the directory with `cd` commands) find files with changed permissions, a task that quickly becomes arduous as more users and other commands such as `copy` or `cacls` are taken into consideration.

We show examples of using our tool to visualize the permissions of the directory tree of Figure 2. Figure 2 shows a directory tree and the effective permissions of every folder contained in this tree. We show in the table of Figure 2 four kinds of permissions, namely the permissions **no access**, **read**, **read&write**, **full control**. Also in the table of Figure 2 we have a column that indicates whether the certain folder inherits the permissions or not (the last column).

In Figure 3(a) we see the representation of our file system with the treemap colored with colors according to permissions, as defined before. In Figure 3(b) we see the treemap layout of the file system after moving a file into a directory that has different permissions from the file. Also, in Figure 4(a) we see the treemap layout of a copy operation and in Figure 4(b) we see the treemap layout of the file system where the permissions of the root node of the directory have changed. Also note that in the presented visualizations we distinguish between the *local* and *effective* permissions. Namely, if the local and effective permissions coincide the tiles are painted with only one color. When this is not the case, we use the upper right corner to indicate the inherited permissions and the bottom left corner to indicate the local permissions. In this way we have a good overview of the permissions that correspond to a file. Note that the figures do not present the break of inheritance of a file since this will clutter up the space. The frames have been produced with the software from University of Maryland (`http://treemap.sourceforge.net/`), where we use the slice-and-dice algorithm for the layout and the increased border option to better display the directory structure.

4 Conclusions and Future Work

In this paper, we have presented an effective method to visualize file system access control. We have outlined the design of a tool that visualizes both effective and local permissions and inheritance interruption for the Windows NTFS file system. Work already in progress is the implementation of a full prototype of our system and to perform user studies to evaluate our approach. As future work, we plan to develop further variations of the treemap layout to display additional file permission information. Also we plan to investigate the application of our technique to visualize the *tree-walking* protocol result used in the *RFID singulation* problem.

Acknowledgments

This work was supported in part by the U.S. National Science Foundation under grants IIS–0713403 and OCI–0724806, by the Kanellakis Fellowship at Brown University, and by the Italian Ministry of Research, grant number RBIP06BZW8, project FIRB "Advanced tracking system in intermodal freight transportation".

References

[1] Bladh, T., Carr, D.A., Schol, J.: Extending tree-maps to three dimensions: A comparative study. In: Masoodian, M., Jones, S., Rogers, B. (eds.) APCHI 2004. LNCS, vol. 3101, pp. 50–59. Springer, Heidelberg (2004)

[2] Bruls, M., Huizing, K., van Wijk, J.: Squarified treemaps. In: Proc. of Joint Eurographics and IEEE TCVG Symp. on Visualization (TCVG), pp. 33–42 (2000)

[3] Cao, X., Iverson, L.: Intentional access management: making access control usable for end-users. In: Proc. of Int. Symposium on Usable Privacy and Security (SOUPS), pp. 20–31 (2006)

[4] Foster, J., Subramanian, K., Herring, R., Ahn, G.: Interactive exploration of the AFS file system. In: Proc. of the IEEE Symposium on Information Visualization (INFOVIS), p. 215 (2004)

[5] Jaeger, T., Tidswell, J.E.: Practical safety in flexible access control models. ACM Trans. Information Systems Security 4(2), 158–190 (2001)

[6] Johnson, B., Shneiderman, B.: Tree maps: A space-filling approach to the visualization of hierarchical information structures. In: Proc. IEEE Visualization, pp. 284–291 (1991)

[7] Johnston, J.B.: The contour model of block structured processes. SIGPLAN Not. 6(2), 55–82 (1971)

[8] Montemayor, J., Freeman, A., Gersh, J., Llanso, T., Patrone, D.: Information visualization for rule-based resource access control. In: Proc. of Int. Symposium on Usable Privacy and Security (SOUPS) (2006)

[9] Reeder, R., Bauer, L., Cranor, L., Reiter, M., Bacon, K., How, K., Strong, H.: Expandable grids for visualizing and authoring computer security policies. In: Proc. ACM Conf. on Human Factors in Computing Systems (CHI), pp. 1473–1482 (2008)

[10] Rode, J., Johansson, C., DiGioia, P., Filho, R.S.S., Nies, K., Nguyen, D.H., Ren, J., Dourish, P., Redmiles, D.F.: Seeing further: extending visualization as a basis for usable security. In: SOUPS, pp. 145–155 (2006)

[11] Russinovich, M.E., Solomon, D.A.: Microsoft Windows Internals, 4th edn. Microsoft Windows Server $^{\mathrm{TM}}$2003, Windows XP, and Windows 2000 (Pro-Developer). Microsoft Press, Redmond (2004)

[12] Stasko, J.: An evaluation of space-filling information visualizations for depicting hierarchical structures. Int. J. Hum.-Comput. Stud. 53(5), 663–694 (2000)

[13] Wilson, R.M., Bergeron, R.D.: Dynamic hierarchy specification and visualization. In: Proc. of the IEEE Symposium on Information Visualization (INFOVIS), p. 65 (1999)