DTEVISUAL: A VISUALIZATION SYSTEM FOR TEACHING ACCESS CONTROL USING DOMAIN TYPE ENFORCEMENT

Yifei Li, Steve Carr, Jean Mayo, Ching-Kuang Shene, Chaoli Wang
Department of Computer Science
Michigan Technological University
Houghton, MI 49931
(906)487-2209
yifli@mtu.edu, carr@mtu.edu, jmayo@mtu.edu, shene@mtu.edu, chaoliw@mtu.edu

ABSTRACT

This paper describes DTEvisual, a visualization system that leverages Domain Type Enforcement (DTE) for access control education. Domain Type Enforcement (DTE) is a powerful abstraction for teaching students about policy complexity and application of the principle of least privilege, mandatory access control and modern models of access control. DTEvisual facilitates graphical depiction, construction, and modification of a DTE policy. It also allows isolation of selected portions of a depicted policy. Finally, a query subsystem identifies the set of files that can be accessed by a specified process under the depicted policy. We anticipate this tool will be useful for classroom presentations, homework assignments, and self-study.

1. INTRODUCTION

Access control systems have become significantly more sophisticated in order to meet modern security requirements. An example is SELinux [9, 10, 11], a version of Linux developed by the National Security Agency. SELinux supports non-traditional models of access control, including Type Enforcement [4], Multilevel Security [2, 3], and Role-based Access Control [12]. These modern systems are very complex. A strict access control policy can contain tens of thousands of rules [9]. As another example, Windows attaches access control lists to objects. Windows access control lists now comprise up to 30 different privileges for operations on about 15 different kinds of objects [8]. It is important for educators to prepare students to deal with the complexity of these modern systems.

This paper presents DTEvisual, a visualization system for teaching access control using Domain Type Enforcement (DTE) [1]. Via DTE, a user essentially groups active entities (e.g., processes) into domains; groups passive entities (e.g., files) into types; and specifies the kind of access each domain has to objects of a given type. A DTE policy is expressed in Domain Type Enforcement Language (DTEL). DTE has several characteristics that make it useful for education in access control [5]. First, DTE facilitates graphical expression of policies. Further, it allows representation of policies that conform to other access control models. DTE exposes students to a process-oriented paradigm for access control, drawing them into the application of the principle of least privilege naturally. It allows the policy designer to balance

policy complexity against application of the principle of least privilege. Finally, it supports relatively concise expression of an access control policy via DTEL.

DTEvisual was developed to make access control education using DTE more practical. Using DTEvisual, access control policies expressed in DTEL can be depicted, developed, and modified graphically. An interactive graphical exploration mode allows a student to isolate selected portions (a domain or a type) of a complex policy. A query subsystem allows students to determine the access individual processes have to specific files under the depicted policy. DTEvisual was used within an elective senior-level operating systems course. Based on this use, we anticipate the tool will be very useful for classroom presentations, homework assignments, and self-study.

2. DTEVISUAL DESCRIPTION

DTEvisual supports two types of text files: DTE specifications given in DTEL (*.dte) and visualization descriptions (*.dtevis). The latter records a DTE specification and the layout of the general and type graphs. The user may import a DTE specification, visually modify it, and save the result to a dtevis file with graph layout or export DTEL to a dte file without graph layout. DTEvisual also allows the user to visually create a DTE specification from scratch, and save or export the design to a dtevis or dte file, respectively. When the user imports a DTE specification, DTEvisual uses a graph layout algorithm [7] to determine the positions of domain and type nodes. This layout algorithm can be interrupted using the space bar.

System Toolbar

DTEvisual has a system menu that includes menu items such as Edit, Settings, and View. The user may right click a node/edge to bring up the associated context menu for faster access to frequently used menu items. Commonly used operations are available in the system toolbar. The toolbar has 21 icons in five groups (Figure 1). Group A has five I/O-related icons (e.g., importing and exporting a dte file), group B has seven editing operators (e.g., moving nodes, highlighting, and adding domain and type nodes), group C has four shortcuts (e.g., opening the query window), group D has three icons for the user to zoom in and out and enter/exit the full-screen mode, and group E has two icons for controlling query animation.

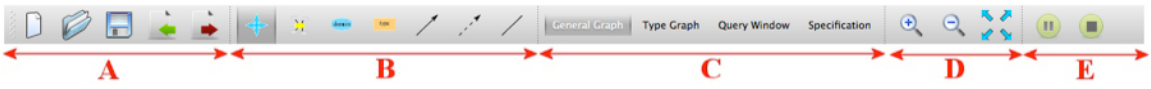


Figure 1

Domain and Type Graphs

Ellipses and rectangles in a general graph (Figure 2(a)) denote domain nodes and type nodes, respectively. Thick solid and dashed directed edges between domains are ‘auto’ and ‘exec’ transitions, respectively, and edge labels are entry points.

Domain and type nodes are connected with undirected edges, and each edge label indicates the permission of a domain has on a type. A type graph is displayed with a radial tree (Figure 2(b)) [13]. Directories and files at the same level are placed on a dotted circle with static types shown in double circles, and a trailing slash is used to differentiate between files and directories.

(a) (b)

Figure 2

Since a general graph may become cluttered when the number of nodes increases, DTEvisual allows the user to highlight certain parts of the graph to alleviate this problem. Two types of highlighting are available. Global highlighting affects all selected items (e.g., edge labels and type nodes), and local highlighting only affects user-defined items. Global highlighting is controlled from the View menu. The user may display (1) only domain nodes and adjacent edges, (2) labels for edges between domain nodes, and (3) labels for edges between domain nodes and type nodes. Local highlighting is available under the highlighting mode through the toolbar. The first time the user clicks a node or an edge, the selected item and its adjacent edges and nodes are highlighted while the rest is grayed out, the second click on the same item only grays out the item itself and its associated edges and nodes, and the third returns to the normal display.

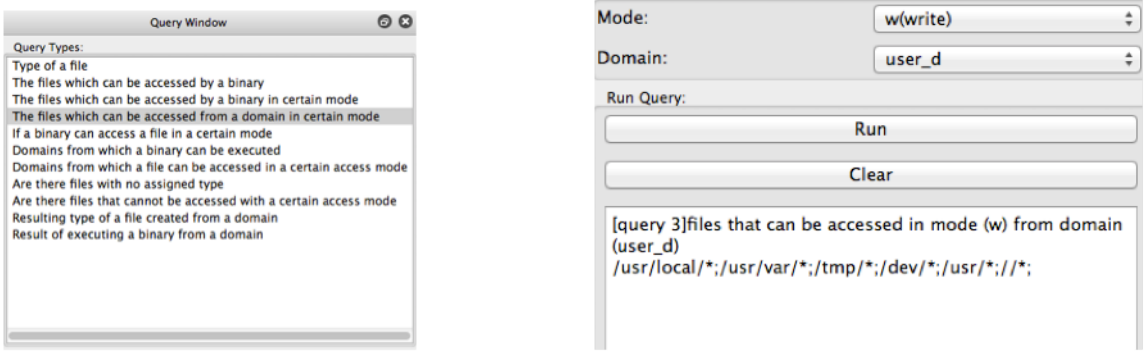
Graph Editing

The user may drag a node in a general graph to change its position. A domain (or type) node is added by clicking at the desired position and a non-empty but unique name is required for the new node. Dragging from a node to the other adds an edge; however, no edge will be added if the user attempts to add an invalid type or duplicate an edge. An appropriate label must be added to each newly created edge. For example, the label of a domain-type edge should indicate valid permissions that only contain characters ‘c’, ‘r’, ‘x’, ‘w’ and ‘d’. The context menu of a node/edge permits the user to delete or change the name of that node/edge. Once a node is deleted, all adjacent edges are also deleted. DTEvisual allows the user to modify a type assignment statement and the modifications will be immediately reflected in the type graph.

For a type graph, the user may change the name, type or flag of a node, delete a type, and assign the type to a child node. All operations are accessed from a node's context menu. DTEvisual performs various checks to ensure no semantic errors are introduced when the user edits a type graph. When the type of a node is changed, DTEvisual checks each of its children to ensure if there is any child which inherits the node's type, and those who are affected will receive the new type. The user may create a new type in the process of changing types, and a node representing the new type will be automatically added to the general graph. A node inherits its parent's type when its type is deleted. If the inherited type is different from its original one, it is treated as if the node's type has been changed. The ability of adding the '-s' flag to a node is disabled if there exists a node among its children which has a different type, and the ability of removing '-r' flag is disabled for those nodes with children (i.e., directories) because directories differ from files by the existence of the '-r' flag. The same principle is used when the user adds a child to a node: the child and its parent have the same type if its parent has the '-s' flag set on, and no child can be added to any node without the '-r' flag.

Queries

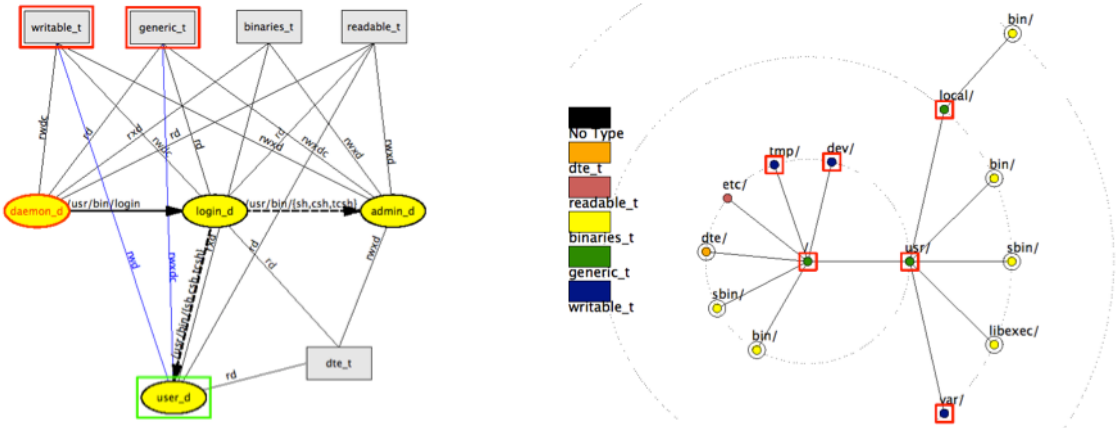
DTEvisual can perform 10 different queries (Figure 3). The user brings up the query window (Figure 3(a)) through the toolbar or the context menu to start a query session, provides the query input, and clicks the Run button to execute the query. The query result is shown in the output frame (Figure 3(b)). By default, DTEvisual shows a step-by-step animation of how a query is carried out. The user may enable/disable or change the speed of the query animation with the Settings menu, and stop and pause/resume an animation process.



(a) The Query Frame (b) The Input/Output Frame

Figure 3

Figure 4 illustrates a query animation. The query in question is to find the files writable by any binary from domain user_d (Figure 3). The types to which domain user_d has write access to are indicated by a red framed box. The edges between the relevant domain and type nodes are shown in blue color (Figure 4(a)). Then, the system displays the type graph and highlights the files/directories whose types are among the one found in the previous step (Figure 4(b)).



(a) (b)

Figure 4

3. CLASSROOM EXPERIENCE

DTEvisual was used in a senior-level operating systems elective. Students in this course have already taken courses in systems programming and concurrent computing. The course had an enrollment of ten. This course covers scheduling, deadlock, memory management and virtual memory, file systems, and operating system security. Topics in access control included the Bell-LaPadula model, DTE, and RBAC (in this order).

DTEvisual was used to present examples of DTE policies graphically. This was especially useful for modifying policies during classroom discussions. Students also used the system to complete homework problems.

With a tool that simplifies presentation, we were able to leverage DTE for several lessons in access control. We used DTEvisual to present a DTE example policy that restricts user processes from writing to the system binaries [14]. This motivated a discussion on tradeoffs between policy complexity and application of the principle of least privilege. The initial example was modified to confine a specific binary to a specific set of files. This further illustrated operation of a DTE specification. We also used this example to demonstrate the inadequacy of UNIX discretionary controls to achieve this same confinement. We then explored expression of a Bell-LaPadula model using DTE. In addition to other problems, students were given an assignment to depict (with paper and pencil) an RBAC policy that was expressed in DTEL, and to modify that DTEL policy to extend the given RBAC policy. This helped them to understand the use of a hierarchy for expressing permissions (since there is no inheritance in DTE) and, at a higher level, how different models are suited to different access control problems.

This first use of DTEvisual was given as extra credit. Not all students (in an already small class) participated, so we decided to forego statistical analysis of the student evaluations. The comments we received were uniformly positive about (1)

the usefulness of the general graph and type graph for understanding a policy (mean 4.5 and standard deviation 0.58), (2) the utility of the tool for finding mistakes in a DTE policy (mean 4.0 and standard deviation 0.82), (3) the use of DTE to increase understanding of the balance between the principle of least privilege and policy complexity (mean 4.5 and standard deviation 0.58), and (4) the use of DTE to better understand the RBAC and Bell-LaPadula models (average 4.0 and standard deviation 0.82). Students also gave high marks to representation and layout of various graphs, and the uses of font sizes, labels, and colors with averages no less than 4.25 and standard deviations between 0.5 and 0.58.

4. FUTURE WORK

Our experience shows that DTEvisual has the potential to be developed into a full blown visualization and pedagogical environment for learning and teaching access control. DTEvisual may be extended in several ways. First, DTEvisual is a quick prototype implemented in Python with some tools such as Qt. As a result, its efficiency can be improved by using a programming language such as C++. Second, to support large and realistic access control systems, new features must be added to address issues created by the large number of nodes and edges in the general and type graphs. We plan to use a Focus+Context technique so that the user is able to see the objects of primary interest in full detail while maintaining an overview of all surrounding information. Third, we intend to redesign the query system to become an open structure so that an instructor is able to add new and remove existing queries. Fourth, DTEvisual can be extended to become a programming+visualization environment [6]. We will use class libraries to intercept requests to the access control system in order to indicate what would be the response on a system that applied the DTE policy. These class libraries will at the same time provide information to the visualization system. In this way, the user is not only able to visualize DTE related activities in real time and receive feedback from the visualization system, but also use the system to debug code and policies.

5. CONCLUSIONS

DTEvisual provides a system that makes teaching access control using DTE more practical. It facilitates graphical depiction, construction, and modification of a DTEL policy. It has an interactive graphical exploration mode that can isolate selected portions of a depicted policy. Finally, a query subsystem allows precise determination of the set of files that can be accessed by a specified process. We anticipate this tool will be useful for classroom presentations, homework assignments, and self-study. Better education on modern access control systems should encourage more widespread adoption of sophisticated approaches and have a significant impact on system security for the systems ultimately managed by these students. The system and example policies are available for download from http://www.cs.mtu.edu/~jmayo/dtevisual/.

REFERENCES

[1] Badger M. L., Sterne D. F., Sherman, Walker K. M., Haghighat S. A., Practical domain and type enforcement for UNIX, *Proceedings of the 1995 IEEE Symposium on Security and Privacy*, 66-77, 1995.
[2] Bell D., LaPadula L., Secure computer systems: mathematical foundations, Technical Report MTR-2547, Vol. 1, MITRE Corporation, 1973.
[3] Biba K. J., Integrity considerations for secure computer systems, Technical report MTR-3153, Rev. 1, MITRE Corporation, 1977.
[4] Boebert W. E., Kain R. Y., A practical alternative to hierarchical integrity policies, *Proceedings of the Eighth National Computer Security Conference*, 1985.
[5] Carr S., Mayo J., Teaching access control with domain type enforcement, *Journal of Computing Sciences in College*, 27, (1), 74-80, 2011.
[6] Carr S., Mayo J., Shene C. K., ThreadMentor: a pedagogical tool for multithreaded programming, *Journal of Educational Resources in Computing,* 3, (1), 2003.
[7] Davidson R., Harel D., Drawing graphs nicely using simulated annealing, *ACM Transactions on Graphics*, 15, (4), 301-331, 1996.
[8] Govindavajhala S., Appel A. W., Windows access control demystified, Technical Report TR-744-06, Princeton University, 2006.
[9] Mayer F., Macmillan K., Caplan D., *SELinux by Example: Understanding Security Enhanced Linux*, Upper Saddle River, NJ: Prentice Hall, 2007.
[10] McCarty B., SELINUX: NSA's open source security enhanced linux, O'Reilly Media, 2004.
[11] National Security Agency, Security Enhanced Linux – CSA/NSS, http://www.nsa.gov/research/selinux/index.shtml, Jan. 15, 2009.
[12] Sandhu R. S., Coyne E. J., Feinstein H. L., Youman C. E., Role-Based Access Control Models, *Computer,* 29, (2), 38-47, 1996.
[13] Tollis I. G., DiBattista G., Eades P., Tamassia R., *Graph Drawing: Algorithms for the Visualization of Graphs*, Upper Saddle River, NJ:Prentice Hall, 1998.
[14] Walker K. M., Sterne D. F., Badger M. L., Petkac M. J., Sherman D. L., Oostendorp K. A., Confining root programs with domain and type enforcement (DTE), *Proceedings of the 6th conference on USENIX Security Symposium, Focusing on Applications of Cryptography – Volume 6*, 6, 21-36, 1996.