ACCESSIBLE ACCESS CONTROL: A VISUALIZATION SYSTEM FOR ACCESS CONTROL POLICY MANAGEMENT

Man Wang
Michigan Technological University, manw@mtu.edu

ACCESSIBLE ACCESS CONTROL: A VISUALIZATION SYSTEM FOR ACCESS CONTROL POLICY MANAGEMENT

By

Man Wang

A DISSERTATION

Submitted in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

In Computer Science

MICHIGAN TECHNOLOGICAL UNIVERSITY

2019

This dissertation has been approved in partial fulfillment of the requirements for the Degree of DOCTOR OF PHILOSOPHY in Computer Science.

Department of Computer Science

Dissertation Co-advisor:	*Dr. Jean Mayo*
Dissertation Co-advisor:	*Dr. Chaoli Wang*
Committee Member:	*Dr. Ching-Kuang Shene*
Committee Member:	*Dr. Steven Carr*
Department Chair:	*Dr. Zhenlin Wang*

Contents

List of Figures

List of Tables

Acknowledgments

I would like to express my gratitude to my advisors, committee members, coauthors, and friends and family. Without their guidance and support, it would never be possible for me to accomplish my research work throughout all these years.

I would like to thank my advisor Dr. Jean Mayo for her advise and encouragement. She helped consolidate my background in Computer Security and also allowed great freedom for my own exploration. She gave a lot of thoughts into what we could bring to the research community, and what is the best for the students in teaching. I appreciate every insightful discussion with her. Her work ethics and her wisdom in life will always be influencing me.

I would also like to thank my co-advisor Dr. Chaoli Wang. He is the special person who lead me into the Computer Graphics and Data Visualization research community. He made me aware of what an excellent researcher is like, and how I should work towards being one by myself. Working with him has granted me many opportunities to collaborate with researchers in the field, and out-reaching activities to the general public.

My appreciation also goes to Dr. Ching-Kuang Shene and Dr. Steven Carr. Dr. Shene has always been resourceful and a great help not only in Computer Science subjects, but also in Mathematics and Data Analysis. His professional experience and advice has shaped my research thinking in a profound way. Dr. Carr has been an expert in field of Computer Security for many years. His work and the collaboration he brought into our group had been a valuable experience. I appreciate every single piece of advice he ever gave to me and our research group.

At last, I am so grateful that I have many good people as my friends, Mengchan Wang, Alex Klinkhamer, Jun Ma and Jun Tao. They share my joys when I am happy, and are always there for me when I need help. They made me who I am today. I also owe my gratitude to my family. I am so lucky to have them in my life; they are the most supportive parents and sister that I could ever wished for. Their encouragement and unconditional love is what drives me forward every single day.

Last to the last, I would like to thank the winter in Houghton. It lets me know that there is something being staying at home is much more fun than going out. It is probably what shaped me into a competent researcher, and also the reason why the local people are so kind and warm-hearted.

Abstract

Attacks on computers today present in many different forms, causing malfunction of operating systems, information leakage and loss of business and public trust. Access control is a technique that stands as the last line of protection restricting the access of users or processes to resources on computers. Throughout the years, many access control models have been implemented to accommodate security requirements under different circumstances. However, the learning of access control models and the management of access control policies are still challenging given its abstract nature, the lack of an environment for practice, and the intricacy of fulfilling complex security goals. These problems seriously reduce the usability of access control models.

In this dissertation, we present a set of pedagogical systems that facilitates the teaching and studying of access control models, and a visualization system that aids the authoring and analysis of access control policies. These systems are designed to tackle the usability problems in two steps. First, the pedagogical systems were designed for new learners to overcome the obstacles of learning access control and the lack of practicing environment at the very beginning. Contrary to the traditional lecture and in-paper homework method, the tool allows users to write/import a policy file, follow the visual steps to understand the concepts and access mechanisms of a model, and conduct self-evaluation through Quiz and Query modules. Each of the four systems is specifically designed for a model of the Domain Type Enforcement, Multi-level Security, Role-based Access Control, or UNIX permissions. Through these systems, users are able to take an active role in exploring the effect of a policy with a safe and intact underlying operating systems. Second, writing and evaluating the effect of a policy could also be challenging and tedious even for security professionals when there are thousands of lines of rules. We believe that writing an access control policy should not include the complexity of learning a new language, and managing the policies should never be manual when automatic examination could take the place. In the aspect of policy writing, the visualization system kept the least number of key elements for specifying a rule: user, object, and action. They describe the active entity who takes the action, the file or directory which the action is applied to, and the type of accesses allowed, respectively. Because of its simple form without requiring the learning of a programming-like language, we hope that specifying policies using our language could be accomplished effortlessly not only by security professionals, but also by anyone who is interested in access control. Moreover, policies can often be left unexamined when deployed. This is similar to releasing program which was untested, and could lead to dangerous results. Therefore, the visualization system provides ways to explore and analyze access control policies to help confirm the effect of the policies. Through interactive textual and graphical illustrations, users could specify the accesses to check, and be notified when problems exist.

Chapter 1

Introduction

Computers nowadays are subject to attacks of various kinds, which have caused compromised system safety, failed computer operations, and jeopardized information, property, and privacy. Access control is a technique that guards the integrity, confidentiality, and availability of data present on computers by regulating the permissions users have to resources. It runs a set of rules, which constitute an access control policy, to stop unauthorized users from accessing data resources, but at the same time, makes sure that authorized users can obtain data with granted permissions. Many access control models have been developed to control access to resources under different scenarios. To make sure a policy fulfills certain security goals, one should choose an appropriate model, write up a policy conforming to the model's format, and make sure the rules collectively achieve the set security goals. This process requires a designer to have the knowledge of access control models, the language of each model, and the individual and collective meanings of each line of rules.

Access control has been covered in some popular textbooks [14, 66] and taught in computer security related courses in different institutes. Usually, it is taught in the traditional whiteboard-and-slides classroom setting, where the instructors play an active role in introducing the subject along with in-class practices and discussions. Due to the limited class time, it is common that simple policies are used as examples for introducing access control models. Explaining more complicated policies used for real system administration is usually not feasible. Students also do not have the opportunities to design policies in class. Thus, the simple policies tend to produce an illusion that access control is easy to master and designing policies does not require much effort. It is not until the homework or exam is returned that the students realize that there is a misunderstanding of the access control models. It is also difficult for instructors to revisit the topic and keep up with the course schedule at the time when these problems are realized. Most commonly, security policymakers and implementers do not have the luxury of taking security courses offered by institutions. Brief training usually helps but still hardly suffices for the professionals to be proficient in the properties of various access control models.

Besides learning access control, the use of access control can also be challenging.

A policy used in large organizations and companies can contain thousands of lines of rules or even more, which makes policy creation and management challenging tasks. Usually, system administrators resort to many tools for policy authoring and analysis [8, 16]. But issues arise when the tools are employed. The output of a tool may be hard to interpret from the input; the outputs from different tools may not be compatible with each other [16]. Therefore, system administrators tend to write up their own command-line scripts instead of using those tools [16]. As a result, the use of access control models is significantly reduced.

To popularize the use of access control, we design a system with visualization and a graphical interface that makes access control easy to learn and use by everyone, in particular for individuals with no security or technical background. The system incorporates two commonly used access control models and is comprised of three major parts: 1) a package of visualization tools to help the learning of Domain Type Enforcement [3], Bell-Lapadula model [9] of Multilevel Security, Role-based Access Control [31], and UNIX permissions access control models; 2) a policy authoring component where policies can be designed using the language of user-object-action and graphical interfaces; 3) a policy analysis component where policies are presented through visualization, and problematic rules can be detected and resolved.

1.1 Background

This section provides the basic background of access control involved in our work. First, the key concepts and the types of access control are introduced. Then some access control models, in particular, Domain Type Enforcement, Multi-level Security, Role-based Access Control, and UNIX permissions, and their properties are discussed in detail.

1.1.1 Access Control

Corporations, organizations, and data centers nowadays possess a wealth of intellectual property that contains information of high-value and may attract attacks from external parties. These attacks take the form of exploiting access to the information. Access control is a technique to ensure that both internal and external users of an organization have no more than the necessary access to resources to perform their own tasks. It restricts a subject's operation to objects using a set of rules. A subject is an active entity that can either be a user or a user-invoked process. An object is a passive entity such as a file, a directory, some memory segments or other computer resources.

Based on how access control applies access constraints, the Trusted Computer System Evaluation Criteria of the Department of Defense categorized access control into mandatory access control and discretionary access control. Mandatory access control (MAC) sets up a central policy where rules that lead to allowing or disallowing operations of users to objects are defined. The policies are usually written and enforced by the system administrators or security specialists, and the end users will not have the authority to modify them. When an access attempt is made to an object, a reference is made to the central policies for a decision. Discretionary access control (DAC), on the other hand, allows an end user to set the permissions subjects may have to the objects (s)he owns. It gives the object owners the flexibility and convenience to set up permissions and has been widely implemented in most mainstream operating systems. A common example is the UNIX permissions, where the determination of access to objects is based on the object ownership of the subject and the group it belongs to. Most commonly, MAC and DAC are used together in modern operating systems. MAC regulates access to objects in a macro way, and DAC allows the object owners the flexibility of sharing or passing to other users the permissions to their own objects.

1.1.2 Domain Type Enforcement

Domain Type Enforcement (DTE) is a kernel-level access control model that is configurable through a specification written in the Domain Type Enforcement Language (DTEL). It separates resources on systems into active entities and passive entities. Active entities, such as processes, are tagged into different domains; passive entities, such as files, directories and memory segments, are grouped into different types. The access of a user or a process a user is running to the objects is determined by the permission the user's domain has to the type of the object.

Operating systems allow one process to affect or generate another process: a process can pass signals to another process to conduct some task; a process can also create another process. The resulting processes in these two scenarios sometimes can be in a different domain than the domain the initial process belongs to. To represent these scenarios, DTE also allows transitions between domains through `auto()` and `exec()`. The `auto` transition is a mandatory transition that will be conducted if an entry point program into a domain to which the current domain has `auto` access is executed through system call `execve`. The `exec` transition is a user-requested transition on system call `sys_dte_execve`. This transition allows the process to stay in the same domain or transition to a different domain upon request.

1.1.3 Multilevel Security

Multilevel Security (MLS) was initially developed for the need to regulate access of users with different security clearances to information in security classifications. The Bell-LaPadula (BLP) model [9], as an implementation of MLS, was developed to meet the security requirements of the US military on their time-sharing mainframe systems. In the BLP model, subjects and objects are labeled with security levels, which consist of a clearance and a set of categories. The clearance indicates a subject's security clearance or an object's level of secrecy (e.g., "Unclassified", "Confidential", "Secret", "Top Secret") and has an inherently hierarchical relation (e.g., "Secret" has a higher secrecy level than "Confidential"). The category shows the content to which a subject has access or an object is related. Each subject or object is assigned to a set of categories. For example, a subject can be part of the Technical group and is also in the Management group. So the category set of the subject is ("Technical", "Management"). In the case of a technical report which belongs to the Technical group, the category set of the report is ("Technical"). The access to objects then is determined by a comparison between the security levels of a subject and an object. We use (L, C) to represent security levels, where L is the clearance, and C is a subset of a comprehensive set of categories in a system. (L, C)'s are compared using the dominates relation $\geq$. $(L_1, C_1) \geq (L_2, C_2)$ if and only if L_1 has a higher or same level of secrecy than L_2, and C_2 is a subset of C_1. The basic principles of BLP define that an object is readable by subjects that dominate the object (no read up), and writable by subjects it dominates (no write down). In this way, it is guaranteed that sensitive information on a system can only be read by people with enough security level, while people with lower security level can contribute new information to the objects with equal or higher security levels.

1.1.4 Role-based Access Control

Role-based access control (RBAC) is an access control model that was developed to associate accesses to resources based on a user's role within an organization. The first RBAC model was proposed by Ferraiolo and Kuhn [30], and an RBAC framework [57] was later developed and accepted as a U.S. national standard [31]. The RBAC model has been widely used for providing enterprise security and developing identity management products. The NIST RBAC model has four levels: the Core RBAC, the Hierarchical RBAC, the Statically Constrained RBAC, and the Dynamically Constrained RBAC. As described in detail below, these models are arranged in a sequence with increasing capabilities. That is, the model at each level has the functionality from the prior level as well as the additional capabilities from their level.

1.1.4.1 Core RBAC

The Core RBAC defines the essential elements of RBAC. The basic concept of RBAC is that roles are defined based on job positions within an organization, and users' access to objects is operated through roles. That is, users' membership to roles defines their access to objects. The user-role assignment and role-object permission assignment is many-to-many, which allows great flexibility and granularity of user to role and permission to role assignment. During a session, a user can activate a subset of assigned roles; a user can also switch activated roles among sessions.

1.1.4.2 Hierarchical RBAC

On top of the Core RBAC, the Hierarchical RBAC is constructed with additional hierarchical relations among roles, which represents job function structures within organizations. The hierarchy shows the seniority among roles mathematically through a partial order where senior roles inherit junior roles. Within this relation, "senior roles acquire the permissions of their juniors, and junior roles acquire the user membership of their seniors" [31]. General role hierarchy allows an arbitrary partial order for the representation of role hierarchy (multiple inheritances), while a restricted role hierarchy only allows a tree structure (single immediate descendant).

1.1.4.3 Statically Constrained RBAC

To this point, roles within an RBAC framework can form hierarchical relations, and users can be assigned to roles freely. In practice, roles, which can supervise one another or play important parts in a common task, are related in a more complex manner. In the case of distributing medicine to a patient, a prescription is needed from a doctor and taken to a pharmacist. Here, a doctor can provide a prescription, but does not have the right to dispense the medication to the patient by laws and regulations. Therefore, one is only allowed to either be the doctor or the pharmacist. Otherwise, it is possible to let the patient unknowingly abuse drugs. Scenarios like this introduce the principle of Separation of Duties, where more than one person is required to complete a task. The principle acts as an internal control to prevent fraud and error throughout the process. The Statically Constrained RBAC, on top of the Hierarchical RBAC, supports the principle of Separation of Duties by restricting user assignment to certain roles under all circumstances.

1.1.4.4 Dynamically Constrained RBAC

Dynamically Constrained RBAC also is an implementation of the principle of Separation of Duties. Instead of disallowing users being assigned to some set of roles at all times, it restricts the roles a user can be at a certain period of time. For example, a user can be a bank teller and also have an account at a bank. However, this user can only play one role at a time to prevent illegal money operations. Furthermore, there could be a restriction of the maximum number of users assigned to a role. Say, for the position of project manager, it is reasonable to allow only one person assigned to this position at a time. This provides the flexibility of changing the manager when one is absent and also guarantees that there is only one person in charge of decision making.

1.1.5 UNIX Permissions

UNIX permissions is the fundamental access control system that comes in the package with UNIX-based operating systems, and has been the most widely used access control model. In UNIX-based systems, each file and directory is associated with its owner, owner's group, and a 9-bit permission set which is decided by the process that creates it. The permission set shows the permission of the owner, members of the owner's group, and everyone else in three of 3-bit parts. Hence, the 3-bit parts are named the user bits, the group bits, and the other bits, respectively. Within each part, the three bits represent read, write and execute permissions. Through setting the bits, rights can be assigned to read a file, write a file, and execute a file (i.e., run the file as a program). Therefore, the UNIX permissions leverage the owner and owner group of an object, and the permission set to collectively determine the access to the object. It gives the owners the flexibility of restricting and sharing the access to files of their own. That is, it can also work with mandatory access control models to provide fine-grained control for file owners under system-wide security specifications. The following parts explain the mechanism of UNIX permissions that includes the notations of permission bits, the process of determining access to an object without and with directory traversal.

1.1.5.1 Letter and Octal Notations

The 9-bit permission set can be expressed in two types of notations: letter notation and octal notation. The letter notation uses “r”, “w”, and “x” in order to represent read, write and execute permissions in the user, group, and other bits. A letter in place means the relative permission is granted, otherwise, a dash is in place meaning that permission is disallowed. Therefore, “rwxrwxrwx” means the owner, group and others all have read, write and execute permission to the object; “rwxr-xr–”, on the other hand, means that the owner has all permissions, the members of owner’s group can read and execute the object, and all other users can only read the object.

If we write all 3-bit parts with 1 representing the letter (allowed permission) and 0 representing the dash (disallowed permission), then the permission set is in its octal notation. In the example of “rwxrwxrwx”, we have “111 111 111”. So the octal notation is “777”. Similarly, for “rwxr-xr–”, the octal notation is “754”. The permission set of an object can be viewed and modified through the UNIX command ls -l and the chmod, respectively.

1.1.5.2 Access to Objects

The access to an object is decided with a combined consideration of its owner, owner's group and permission set. For files and directories, read, write and execute permissions have different meanings:
For a file,

- Read: Read the content of a file.
- Write: Write the content to a file.
- Execute: Execute the file as a program.

For a directory,

- Read: Allow the listing of the files and directories under the directory.
- Write: Create or delete the files and directories under the directory.
- Execute: Go through the directory. This permission is crucial if a user needs to have access to the files and directories under the directory.

When determining one's access to an object, the owner and the owner's group of the object are compared to the user who makes the access request. If the user is the owner, then the user has the permissions indicated in the user bits. Otherwise, the user is checked upon its membership of the owner's group. If it belongs to the group, then the group bits are used for access evaluation. If not, the user has the permissions specified in the other bits. When the directories containing the object of interest are considered, extra evaluation of the execute permissions of all directories should be included as well to check if the user is able to reach the object.

1.2 Challenges

In the past few years, with the increasing recognition of protecting digital resources, more and more sophisticated access control components have become available in operating systems. Many Linux distributions also contain packages of access control components such as AppArmor and SELinux. Fedora and the RHEL distributions come with the security components initially enabled. However, while data breach incidents keep pushing the need for systems and information with stronger security protection, access control only managed to provide limited help in improving the integrity, confidentiality, and availability of digital resources.

One reason for this is that security is usually considered secondary by end users compared to completing their primary jobs on time [81]. It is common that limited time is spent on incorporating proper access control, and it is usually conducted if access control is a required work. But even when access control is being used, there are problems. End users, unlike security professionals, are nontechnical individuals who use access control tools to fulfill macro security requirements and lack the knowledge and experience of access control models. Reeder et al. [45, 52] found that the end-users can not correctly interpret the changes in access control lists through the traditional ACL user interface. It is also interesting that even experts do not use ACLs due to the same reason [11]. Moreover, RBAC, the most widely used access control model, suffers from the same usability problem. It takes substantial time and effort of end users to learn and use RBAC. Therefore, another obstacle to accurate access control policies is human error [2, 32, 45], which is brought by the lack of knowledge of the access control model in use.

Many access control languages and tools were developed to make policy authoring easier. However, the usability of access control did not show significant improvement. In the case of RBAC, most tools are designed for system administrators and security professionals [18]. Some of them have a simple user interface but require prior knowledge of the model for users to understand the terms in the user interface. XACML, as a dialect of XML, is used to specify attribute-based policies. Its use not only requires the understanding of the access control model itself, but also the knowledge of the language syntax if the fact that writing an XACML could be rather tedious with numerous nested tags is ignored.

With more and more nontechnical individuals and groups are getting involved in setting up access control within organizations, current tools did not help much in making access control easy to learn and use for the protection of their resources. From the previous work, we see that 1) users spend little time learning the models and the models are difficult to learn even for expert users; 2) when they have to learn or use access control, the existing tools are not designed for their use as too much prior experience is needed; 3) some tools suffer from usability problems even when used by security professionals. In summary, the current status shows a lack of a set of tools with decent usability and systematic design to kick start the learning of access control models and ease the authoring and management of access control policies.

1.3 Methodology

To resolve the above issues, we developed a system to facilitate both the learning and policy management of access control models. The system consists of three parts: 1) a set of pedagogical tools for the commonly-used access control models, 2) a policy authoring component where policies can be designed using user-object-action language and graphical interfaces, and 3) a policy analysis component where policies are presented through visualization, and misconfiguration can be detected and resolved.

The current tools in the field have shown a lack of focus on aiding the learning of the models and a shortfall in covering multiple models. We design our pedagogical system to have a set of tools, each of which is developed specifically for one well-known model, in a unified design structure. As policy language aids correct specification over individual permission setting [37], our system will have a language for each of the access control models. These policy languages will have similar syntax across models and be in the form of a combination of natural language and programming language that inherits the simplicity of the former and the precision of the latter. Policies written in those languages are then taken as the input of our tools. In the presentation of the policy properties, we opt to show the components and their relationship of a model in a graphical manner. As thousands of lines of rules in one policy could introduce large numbers of entities and complex relationship, users can be easily overwhelmed if exposed to the information all at once. Hence, a method that shows users the only necessary information at each step and provides response and guidance to the users' action is needed. To this end, incorporating visualization is an appealing approach for this goal as it allows temporal and structural presentation of entities and provides a dynamic response to users' actions that could enhance the understanding of the material [54]. Lastly, we believe that it is important to include a way to exercise and examine their understanding as the role of class tests since there is a significant amount of new learners who do not have access to those courses and help from others. Therefore, they would miss the opportunity to test their understanding from which some unrecognized misperception could be found out.

Our policy authoring and analysis tool focuses on easing the process of composure and management of policies for new learners and security professionals. This component would take a policy file and report property analysis to the users. We use RBAC as the implementation model, but simplify the method of specifying access control rules. For new learners, we have a graphical interface that allows users to enter values of policy components through the user-object-action template. In this way, new learners are exempt from the work of learning RBAC model language and would have an easier start with a template that is akin to the common access control requirements. For experienced users, we support the import of a policy file written in our user-object-action language, which is designed to be a possibly more efficient way of writing policies of large scales once the language becomes familiar to them. Due to the fact that policies are usually designed and implemented by more than one person, rules that affect the access to an object can be difficult to locate and interpret altogether. Even if the rules are tested afterward, which is a rare practice, understanding the effective permission and whether problems exist among rules are still obscure to users. We design the tool to directly depict the effective permissions for users, and automate the process of detecting and reporting issues within the policy. Through visualization, users could have step-by-step guidance of what has been affected by their operations, and further get to know possible problems and make informed decisions. We plan to provide two levels of detail for the illustration of a policy for different users. For users who are only interested in getting a security module set up, the effect of the policy is shown while how the effect is achieved is hidden. The check of access can be accomplished in both textual property checker and graphical representations where no underlying model information is displayed. For users who are interested in knowing more, such as the details of which rules are in effect, how RBAC components affect the access results and whether and why any issue exists in the policy, our tool will provide interactive visualization to notify the problems to users. In this way, new learners will be able to observe the analysis of the policy rules in detail while experienced professionals, whose primary goal is to finish the task with the least amount of time, could opt to directly test the policy and resolve problems to obtain immediate results.

1.4 Organization

This dissertation is structured as follows. Chapter 2 introduces the works that have been accomplished in the fields of access control authoring and analysis, and discusses our approaches to tackle the existing problems. Chapter 3 presents a set of pedagogical systems we designed for new learners to study individual access control models. Chapter 4 introduces ACvisual, another visualization system for policy authoring and analysis provided for both new learners and security professionals. Chapter 5 includes two pieces of software that facilitate the teaching of Fluid Dynamics, FlowVisual for two- and three-dimensional flow fields. Lastly, chapter 6 presents the conclusion of our work and findings, and future directions for improvement.

Chapter 2

Related Work

In this chapter, we first investigate the use of access control in large corporations and organizations, followed by a discussion of the languages for access control policy design. Then we review the tools that were developed to help facilitate the teaching and learning of access control and security models, and the authoring and analysis of access control policy both in the academic and industrial environment.

After the occurrence of system incursion and data compromise, companies and organizations nowadays have an increased recognition of the importance of controlling access to the intellectual property on the computer systems to protect their privacy, safety, finance, business, and so on. Access control is developed for this purpose, and many advanced models, such as DTE, MLS, and RBAC, were implemented and packaged with operating systems. A good amount of languages and tools have been developed to make access control models easier to learn and use. However, UNIX permissions is still many administrators' favorite choices, even when it comes to protecting system data.

2.1 Access Control Languages

EXtensible Access Control Markup Language (XACML), is a specialization of XML for composing security policies. It is taken as the standard access control policy language in many tools and operating systems. For large organizations where policies may be enforced in a distributed manner, XACML-based policies have the advantage of high re-usability in implementing effective and consistent access control mechanism. Similar to XML, the drawback of XACML is that it is a strict language where numerous tags should be used to specify attributes. The concepts of the rule, policy, and policy set are supported. Each of them can be an intensively nested tagged property, and the latter is a collection of the former concept, which makes writing an XACML policy tedious and error-prone.

Another widely-acknowledged language is provided by Security Enhanced Linux (SELinux). SELinux is a Linux kernel security model that provides a means of fine-grained and cross-model mandatory access control. It sets up access control using configuration files that describe a security policy. SELinux is highly configurable and can handle a large number of applications as well as system resources with rather fine resolutions. But high configuration also brings high complexity as more resources and conditions are considered. SELinux could also be rather challenging to start with given a rich collection of tools and commands it provides. This is because its ability to set up rules of multiple access control models requires prior knowledge of those models. Additionally, the fact that SELinux uses a prototype of mixed models for policy writing could cause more confusion than using separate languages for individual models, which may be further less error prone.

2.1.1 Our Approach

Unlike XACML and SELinux being programming type of languages, the languages in our tools are designed with a common goal of being as close to the original structure of the model to express as possible. In the set of pedagogical systems for individual access control models, a language specifies the key components of a model. Take UNIX permissions for instance, a policy is consisted of specifying 1) a root directory; 2) a list of users; 3) a list of users assigned to each group; 4) the permission bits and user and group owners of each object in the form similar to the output of command "ls -l". Even though each model has a distinctive language, the languages fill in the key components in a linear manner, which could also help familiarize user to the structures of the access control model during the policy writing. In the policy authoring and analysis tool, users are provided with a user-object-action template, where each rule specifies users who can access the file or directory objects through what is allowed in the action. This type of specification resembles the form of the common security requirements in an organization, and allows the explicit assignment of each component. Both the model-specific and the user-object-action languages exempt users from unnecessary complexity of learning a programming or script language, thus could possibly let new learners and security professionals alike focus on the core task of setting the correct policies. To make writing policies even easier, the tools also provide graphical interface to build specifications from scratch.

2.2 Tools for Security Education Using Visualization

Visualization has been applied to the teaching and demonstration of access control models. Schweitzer et al. developed a visualization system to enable active learning about the HRU (Harrison, Ruzzo, Ullman) and Take-Grant models of access control [61]. The system teaches the concepts at an abstract level combined with instructor demonstration and student interaction. It saved students from getting into the unnecessary details of the underlying system. Compared with no assistance of the tool a year ago, the average score of homework questions increased from 33% to 93%, which strongly indicates that the tool was beneficial in reinforcing the learning of the concept. One drawback of the tool is that it is only limited to be effective in the in-class study.

Visualization has also been applied to Computer Security pedagogy. Crandall et al. [22] presented a Java applet that demonstrates in programming courses that how and why buffer overflow occurs in C programs. The tool explains why some defense mechanisms should be applied in different scenarios. The US Air Force Academy has developed a set of interactive classroom visualization tools to increase the background knowledge and experience of information security of the Computer Science majors as well as all cadets [61]. One of these tools [26] was developed by students studying Software Engineering course to demonstrate Public Key Infrastructure. The development experience deepened the developers' understanding of the topic. The tool itself helped the non-computer science students learn better. Another tool [59] visually demonstrates the sequence of passed information and computations of arbitrary protocols step by step. The effects of attacks, such as eavesdropping, forged messages, communication blocks, and message replays, could be manipulated through interactive protocols. This tool could be used in a laboratory exercise, and also is applicable in classroom environment. A set of tools was also designed to help the teaching of Cryptography in Information Assurance curriculum [58]. The tools aim to address the lack of understanding of the basic concepts and both historical and current cipher algorithms. They can be used in the classroom and as out-of-class exercises. Students' reaction to the tool demonstrations were quite positive. During the class, students were more engaged and showed better understanding over questions and concepts that had challenged them when only taught through traditional lectures. However, these tools are not intended for new learners without any instructional materials.

2.2.1 Our Approaches

Many tools were developed for topics of Secured Coding, Cryptography, and Cyber-attacks. But there are not many for access control models when it is a subject that students found challenging based on our teaching experience. This is due to the fact that traditional lectures can teach the ideas, and the homework questions could not reach a scenario of a realistic scale. Since the ideas are abstract in nature and designing and setting appropriate policies is an art based on practice, there is a need to fill the gap between learning through reading and effective learning by practice. We design our tools so that students are given an interactive visual presentation of concepts and their relationship, freedom to create and explore concrete policies at different scales on their own, simple to comprehensive depictions of access evaluation, and self-tests for assessment. The tools are designed as assistance to in-class teaching, as well as a portable program for self-learning.

2.3 Tools for Security Policy Management

RBAC/Web is an RBAC administering tool developed for networked servers using WWW protocols [7]. Through RBAC/Web, users can be assigned to roles and build the role hierarchy through a graphical interface. It supports UNIX as well as Windows NT environments. The advantage of RBAC/Web is that it can be run directly on servers without any server internal changes or source code access. RGP-Admin [6], developed by the same team, is another tool for specifying RBAC policies. It introduced an independent concept of "Object Access Type" from objects for access control specification. The tool provides a graphical interface for users to create/-modify the role/permission association through"Object Access Type", view existing "Object Access Type", and view a role's permission to objects. While "Object Access Type" does classify objects into different categories, the new concept also increases the complexity of managing RBAC. First of all, how many and what "Object Access Type" should be could be difficult to determine. This concept is similar to the concept of role. The only difference is that one is for organizing users while the other is for objects. Secondly, as selecting good roles is still a research topic today [41], adding this role-like concept could double the amount of intricacy in the RBAC model. These two tools were built on top of the RBAC model structure with a more user-friendly interface. However, neither of them provides a way of teaching user the model or any interactive feedback during policy writing. As the improvement is solely based on better interface design, it fails to guarantee the learning of the model and the quality of the resulting policy.

Access Control Policy Tool (ACPT) [38] was developed by the Computer Security Division of NIST and North Carolina State University. It provides four aspects of functionality: 1) providing a graphical interface template for policy writing; 2) providing access control property conformance check; 3) providing a complete test suite; 4) providing automatic generation of XACML output files for verified policies. Through these functionalities, ACPT reduces the challenge and increases the correctness of writing an access control policy. It also exempts users from the tedious conversion from designed access control rules to the usable XACML policies. ACPT supports the RBAC and MLS model as well as the user-object-action rules, which is specified in the form of whether a subject can perform certain actions to some objects. Though new learners can opt to use the user-object-action rules as a generic way of policy writing, RBAC, MLS or any other access control models is left unknown to them. With the convenience ACPT brings, it still requires knowledge of an access control model if policies must be composed under the model's framework.

Some tools incorporated visualization for the illustration of the mechanism of access control models. DTEedit/DTEview [34] are tools that were developed to help the understanding and administration of the DTE model. The file type analyzer in DTE-view allows a user to traverse from a specified directory and show the types of the objects along the path interactively. The process analyzer simulates the initiation of a system. Possible domains to enter in the next step are shown and could be further selected upon the user's request. Then the detail of the types that the selected domain has access to will be displayed. DTEview can also analyze a policy and give out warnings when a potential security compromise is detected. The tool can help a user walk through and analyze a DTE policy in detail, but does not serve a pedagogical purpose. Expandable Grids [52] is a tool that shows effective access control using an interactive matrix given a policy. Permissions can be directly edited through the graphical interface, and effects to other grids will be reflected immediately. Their user study shows, with a control group using the Windows XP file permission interface, that the Expandable Grid interface allows the users to write policies with higher efficiency and correctness. The positive results were obtained through testing a range of fundamental policy authoring operations and held for policies of both small and large scales.

More policy management tools such as Adage [87] and IAM [20] were designed for manipulating RBAC model, access control on the web and effective access control policy for WebDAV. Both of the tools have been tested in a user study. Adage showed that novice users were able to effectively manipulate many access control properties such as use of groups, attributes, constraints, and rules without any guidance. The user study of IAM showed that it achieved more accurate manipulation and better usability than many traditional tools.

Commercial tools were also developed to facilitate access control on systems. Permissions Analyzer [64] analyzes a user's effective NTFS permissions for a specific file or folder. It takes the network share access into account, then displays the results in a nifty desktop dashboard. Eiciel [48], as a built-in component of the GNOME file manager, lists users and groups of an opened file and helps to manage the ACLs for those users/groups on that file.

2.3.1 Our Approaches

These past works improved interface design and policy visualization to make access control usable, but were all designed for technical administrators. For novice users who need to apply security to their work but lack the security background, the existing tools advance forward to the policy management part and have ignored the importance of explaining the models. So the tools are hard to pick up and thus of less help. We believe that it is important to design a tool that can make learning the access control models easier, so that the understanding could further help the production of quality policies. In order to make access control usable for users with different levels of experience, we integrate policy writing with an easy-to-understood template and policy analysis with both textual and graphical representations. Further, we provide the cyclic role inheritance and violation with separation of duties detection, and access property checker for the enforcement of policy correctness.

Chapter 3

Model-Specific Pedagogical Systems

This chapter introduces the first part of our work, a set of pedagogical systems designed to facilitate the teaching and learning of access control models. Individual systems, namely DTEvisual, MLSvisual, RBACvisual, and UNIXvisual, were developed for the most commonly used models of DTE, MLS, RBAC, and UNIX permissions. The visualization systems take a policy as input and provide a highly interactive illustration of the entities and the relations within an access control model. Model-specific languages with simple syntax and intuitive semantics are also designed for policy authoring. This makes it easier for users to experiment with policy writing and to focus on how each model works instead of taking extra efforts to get familiar with the configuration of a specific model. The following sections introduce each of the tools in the aspects of their model-specific languages, the design of the major components, and the user evaluation results. We will describe UNIXvisual in details as UNIX permissions is most widely used access control model. At last, a framework is introduced for designing pedagogical tools in the field of access control.

3.1 DTEvisual

DTE access control model allows processes with similar access requirements to be collected into domains, and objects to be assigned to different types. The access rights of each domain has to be associated to objects of a given type, and thus facilitates tight control over policy granularity. DTEvisual [44] is the first of the series of tools under the "Accessible Access Control" NSF project developed by my group member YiFei Li, and maintained by myself. The tool designed to ease the learning and teaching of the DTE model. It supports two types of textual input/output files: DTE specification (*.dte) and visualization description (*.dtevis). The DTE specification could be directly composed in the DTEL syntax; the visualization description can be composed using DTEvisual through graphical editions. It records the DTE specification in graphical entities as well as the layout of the visualization. This provides an advantage of storing a reusable depiction that users could refer to later to build jobs step by step.

3.1.1 Language

DTEvisual simply incorporates the existing DTEL as its policy language, given that it is an intuitive and standard language for DTE policies. DTEL uses implementation-level structure to specify the properties and constraints for security settings. A DTE policy usually contains four sections. The syntax for rules in each section is as follows:

1) Type Definition

   Declares all object type names in a system, which may be used in other parts of a DTE policy. The statement starts with a keyword `Type` followed by an unordered list of object types. The names of types follow the convention of having a suffix of "_t" as an indication that it is an object type.

   ```
   Syntax format: Type typeId1_t, typeId2_t, typeId3_t,...;
   ```

2) Domain Definition

   A domain is defined as a list of tuples. The statement starts with keyword `domain` and is followed by the domain name and its execution environment. The execution environment is comprised of three parts. The first part is a list of entry point programs. These entry point programs are identified by pathnames, one of which a process must execute in order to enter the domain. For instance, the entry point program of domain `login_d` could be `/usr/bin/login`. The first line of the domain definition then is:

```
domain login_d = (/usr/bin/login),
```

The second part is the access rights to some object types. Say if `login_d` can read the objects and go through the directories of `readable_t`. The domain definition so far is:

```
domain login_d = (/usr/bin/login),
          (dr->readable_t);
```

The third part is to specify transitions between domains. There are two types of transitions. The `auto` transition is a mandatory transition that will be conducted if an entry point program into a domain to which the current domain has `auto` access is executed through system call `execve`. The `exec` transition is a user-requested transition on system call `sys_dte_execve`. This transition can allow the process stay in the same domain or transition to a different domain upon request. If `login_d` can switch to domain `user_d` through `exec` transition, then the entire definition of `login_d` is:

```
domain login_d = (/usr/bin/login),
          (dr->readable_t),
          (exec->user_d);
```

Additional access to other object types could be added before the line of domain transition in the same syntax of the second line.

3) Initial Domain

Initial domain specifies the domain the system starts with.

```
Syntax format: initial_domain = login_d;
```

4) Type Assignment

The type assignment associates a type with one or more objects under the restriction that each object can only be assigned to one type.

```
Syntax format: assign [OPTION] typeId_t object1, object2,...;
```

The OPTION could be a combination of the following flags:

(a) `-s` indicates that the type assignment is static. It means that the type associated with the pathname cannot be changed at runtime, even if the object is renamed.

(b) `-r` indicates that the type assignment is recursive. That equivalently means that the type is assigned to both the objects listed in the statement and all objects in those directories.

A type assignment can be overridden if there are multiple type statements referring to an object. The effective type of the object is determined by the type assignment with the object of the longest path prefix. For instance, there are type assignments:

```
assign -r unprotected_t /;

assign -r -s dte_t /dte;
```

Then the type of `/dte/config.txt` is `dte_t` instead of `unprotected_t` as `/dte` is a closer ancestor.

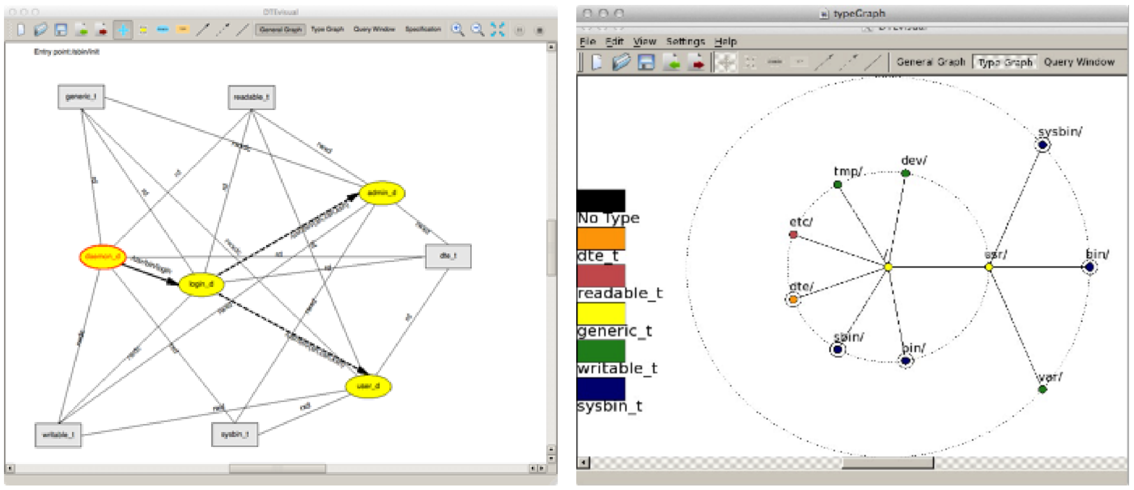
A complete example policy is as below:

```
type unprotected_t, passwd_t, dte_t;
domain unprotected_d = (/sbin/init),
        (cdrwx->unprotected_t),
        (exec->passwd_d);
domain passwd_d = (/usr/bin/passwd),
        (crw->passwd_t);
initial_domain = unprotected_d;
assign -r unprotected_t /;
assign -r -s dte_t /dte;
assign passwd_t /etc/passwd, /etc/shadow;
```

Figure 3.1: DTE policy in DTEL

3.1.2 Visualization System

DTEvisual supports two types of textual input/output files: DTE specification in DTEL (*.dte) and visualization description (*.dtevis). The DTE specification could be directly composed in the above syntax or composed using DTEvisual through graphical editions. The visualization description records the DTE specification in graphical entities as well as the layout of the visualization. This provides an advantage of storing a reusable depiction that users could refer to later to build jobs step by step.



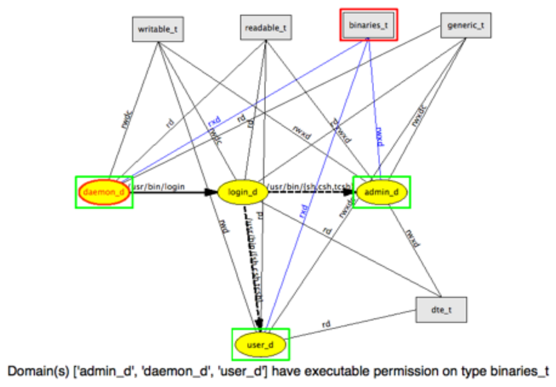
(a) General Graph (b) Type Graph

Figure 3.2: Visualization in DTEvisual

DTEvisual consists of two major graphs: the General Graph and the Type Graph. The General Graph depicts the domain definition, the domain-to-type permissions, and the domain-to-domain transitions. In Figure 3.2 (a), domains are represented by ellipses, and types are represented by rectangles. Permission a domain has to types are depicted in undirected edges between an ellipse and rectangles. The text labels on the edges specify the permissions, where `c`, `d`, `r`, `w`, and `x` denote create, pass through (a directory), read, write and execute, respectively. The `d` permission applies to directories and is similar to the `x` permission to directories on UNIX. The `r`, `w`, and `x` permissions are similar to `r`, `w`, and `x` on UNIX. The `c` permission indicates the default type for objects created by processes in the domain. Solid and dashed arrows represent `auto` and `exec` transitions between domains, respectively. The Type Graph depicts the type assignments to objects and provides an overview of the distribution of all types in a system. In Figure 3.2 (b), objects are represented as circular nodes and structured in a radial tree [12] where the center is the root directory, and directories and files at the same level are placed on one circle. Each type has a unique color and is listed on the left side of the visualization area. The object nodes associate with their types by sharing their colors.

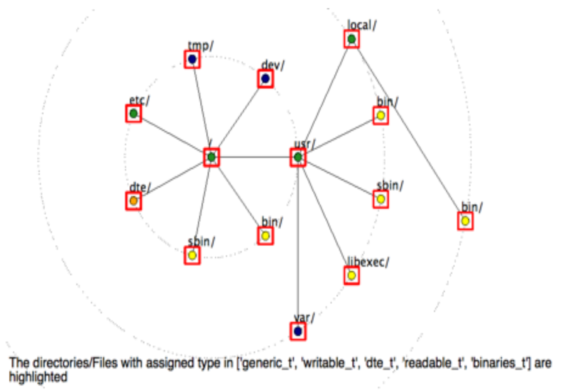
DTEvisual allows users to highlight part of a graph for focused analysis. There are global and local highlighting available. From the global highlighting, one could make visible gradually 1) all domain nodes and their edges, 2) domain transition edges, and 3) domain-to-type edges. The local highlighting, on the other hand, is user-click triggered. A user can start the highlighting by clicking any node or edge and only the clicked item and the nodes and edges connected will be shown with the rest of the graph grayed out. A second click on the same item will show the opposite. The third click will bring the visualization back to the normal display. DTEvisual also supports graphical policy editing through the context menu of adding, moving, modifying and deleting items. Editions are checked with syntax references and reflected on the visualization immediately.

To facilitate self-learning of the model, a query subsystem is designed to provide solutions to 10 commonly asked questions. The questions are listed in a side widget. When running a query, the answer text is displayed in that widget and an animated visualization provides a detailed explanation of the steps leading to the solution. Users may enable and disable the animation and its speed. The animation process could be paused or resumed at any time. Figure 3.3 has the example animation from two queries.



Domain(s) ['admin_d', 'daemon_d', 'user_d'] have executable permission on type binaries_t

(a) Query 1



The directories/Files with assigned type in ['generic_t', 'writable_t', 'dte_t', 'readable_t', 'binaries_t'] are highlighted

(b) Query 2

Figure 3.3: Query Subsystem

3.2 MLSvisual

MLSvisual[1] is the second tool we developed in the same series for the Bell-Lapadula model. It follows a similar framework of using the same types of input and output files, representing textual policies through visualization, and providing solutions through an embedded query system. In addition, MLSvisual implements a Quiz module where users can conduct a test to evaluate their understanding of the model.

[1]The material contained in this section was previously published in proceedings of the 2014 ACM Conference on Innovation and Technology in Computer Science Education [73]

3.2.1 Language

Unlike DTE, MLS does not have a standard high-level language to specify policies. We designed a language of this kind in a similar syntax with that of the DTEL to minimize the learning overhead. The syntax is as below:

1) Clearance Statement

   Define one or more clearance levels, which are then available to other parts of an MLS specification. Clearance Statement starts with keyword `clearances` followed by a list of all security levels from low to high sensitivity.

   ```
   Syntax format: clearances: clearance1<clearance2<...
   ```

2) Category Statement

   Define one or more categories, which are then available to other parts of an MLS specification. The Category Statement starts with keyword `categories` followed by an unordered set of categories separated by commas.

   ```
   Syntax format: categories: category1, category2...
   ```

3) Object Security Assignment

   Assigns security levels to objects in file systems. The statement consists of the keyword `assign`, a security level, optional recursive flag, and a list of objects.

```
Syntax format:

assign clearance:category1:category2:... [-r] object1, object2,...
```

The recursive flag has the same meaning as that in the DTEL, and an object's security level is determined by the assignment with its more recent ancestor.

4) User Security Assignment

   This type of statement assigns security levels to users. A statement contains the keyword `users`, a security level, and a list of users.

```
Syntax format:

users clearance:category1:category2:... user1, user2,...
```

Figure 3.4 has a complete example of MLS policy.

```
clearances: secret < topsecret < seven

categories: michigan, texas

assign secret:michigan  -r /usr, /top
assign seven:michigan     /usr/somefile

users secret:michigan:texas  fred, ethel
users seven:texas                ricky, lucy
```

Figure 3.4: MLS policy

3.2.2 Visualization System

MLSvisual supports the import and export of textual specification files (*.mls) and visualization files (*.mlsvis). A specification file is a human-readable, text-based specification written in the above language. A visualization file stores the graphical entities and their layout so that the visualization can be restored for later use. These two types of file can be converted to each other through the visualization system.

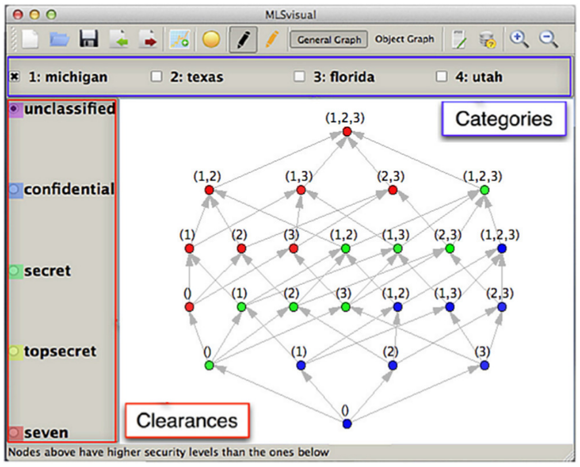
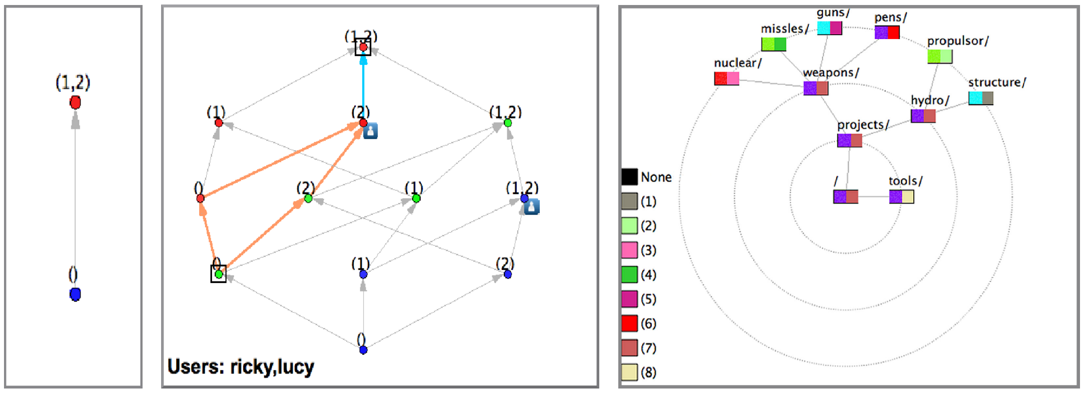


Figure 3.5: Main Window (© 2014 ACM. Reprinted by permission.)

Figure 3.5 shows the interface of MLSvisual. The main area is divided into a toolbar, a clearance section, a category section, and a visualization section. The clearances are arranged in the least to the most sensitive order from top to bottom with different colors. The unordered categories are associated with numbers. With the colors representing the clearances and the numbers representing the categories, a security level, which is comprised of a clearance and a category set, can be denoted as a colored node labeled with a set of numbers. The lattice in the visualization area is built from the dominates relation between security levels.

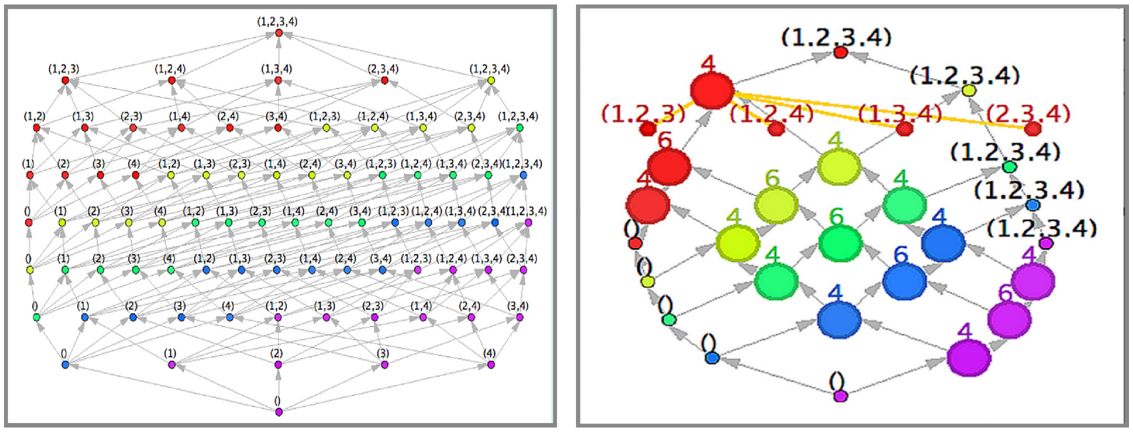
MLSvisual contains the Whole Graph, the General Graph, and the Object Graph. The Whole Graph draws a lattice of the dominates relation among all security levels. The General Graph shows the relations between security levels of the user's interest. It can be used to demonstrate the change of dominates relation between security levels as they are put into use one by one. The Whole Graph and the General Graph together provide overall and partial views of the relationship among security levels to facilitate a full understanding of a policy. The Object Graph depicts the security level assignment to objects (Figure 3.6 (c)). This graph has a number of concentric circles with the center being the root directory. The circles with increasing radii represent directories of increasing directory depth. The nodes in the graph are objects and the edges represent the membership of the directory. Each node is a rectangle with two colors. The left color indicates its clearance and the right one shows the category based on the color-category correspondence in the legend.



(a) (b) (c)

Figure 3.6: General Graph and Object Graph. (a) and (b) show two different General Graphs. (c) shows the Object Graph. (© 2014 ACM. Reprinted by permission.)

In the General Graph, there are two ways to explore the relation between security levels. One is to add one security level node at a time. The other one is to specify the starting and ending nodes and generate the full directed graph of all in-between security levels at once. The first method draws an edge directly between nodes when they are related under the dominates relation. The lattice is updated when more nodes are added into the visualization. This shows how dominates relation works in progress. The second method shows a full graph including the intermediate levels. This is useful when investigating the reachability of two given nodes, the possible paths and the involved security levels. It also avoids the overwhelming and repetitive operations of adding nodes one by one. Figure 3.6 (a) shows two nodes and their relationship. Figure 3.6 (b) shows the resulting lattice after applying the second method. The security level node with an icon of human outline indicates there are users assigned to the security level. When mousing over the node (e.g., the red node with label (2) in Figure 3.6 (b)), the users' names are displayed, and its permissions to other security levels are shown in highlighted paths. These subjects can write to the nodes along the blue paths and can read the nodes along the orange ones. Combined with the Object Graph, the permissions those users have to certain objects can be identified. As a fine-grained policy may introduce large numbers of clearances and

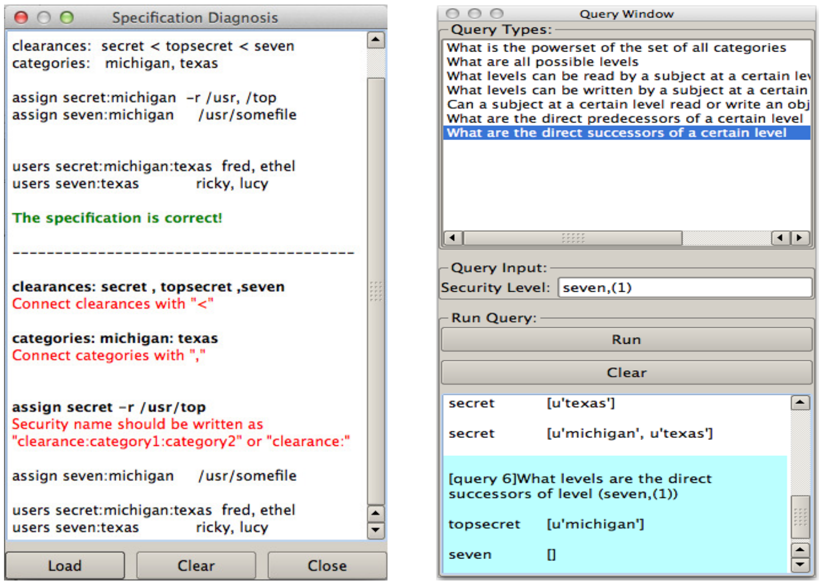


(a) (b)

Figure 3.7: Whole Graph. (a) shows the graph without Grouping. (b) shows the graph with Grouping. (© 2014 ACM. Reprinted by permission.)

categories, the lattice of security levels in the Whole Graph can be rather cluttered. We apply node grouping to the nodes of the same clearance at the same depth of the complete lattice and label the group node with the number of the nodes contained. This group node can be expended and replaced by a contained node. Figure 3.7 (a) and (b) show a Whole Graph without and with grouping.

MLSvisual also supports an edit mode, where clearances and categories as well as the security level assignments to subjects and objects can be edited. This mode allows users to modify a policy through visualization and aims to help them design policies to fulfill specific security requirements. In addition, Specification and Exercise



```
clearances:  secret < topsecret < seven
categories:  michigan, texas

assign secret:michigan  -r /usr, /top
assign seven:michigan     /usr/somefile


users secret:michigan:texas  fred, ethel
users seven:texas            ricky, lucy

The specification is correct!

----------------------------------------

clearances: secret , topsecret ,seven
Connect clearances with "<"

categories: michigan: texas
Connect categories with ","


assign secret -r /usr/top
Security name should be written as
"clearance:category1:category2" or "clearance:"

assign seven:michigan     /usr/somefile

users secret:michigan:texas  fred, ethel
users seven:texas            ricky, lucy
```

```
What is the powerset of the set of all categories
What are all possible levels
What levels can be read by a subject at a certain lev
What levels can be written by a subject at a certain
Can a subject at a certain level read or write an obj
What are the direct predecessors of a certain level
What are the direct successors of a certain level

Security Level: seven,(1)

secret       [u'texas']

secret       [u'michigan', u'texas']


[query 6]What levels are the direct
successors of level (seven,(1))

topsecret    [u'michigan']

seven        []
```

(a) (b)

Figure 3.8: Specification and Query. (a) shows the Specification Diagnosis Window. (b) shows the Query Window. (© 2014 ACM. Reprinted by permission.)

modules are designed to assist the authoring of MLS policies and self-evaluation of the understanding of the MLS model. The Specification module has Specification Window and Specification Diagnosis components. The Specification Window component generates a specification of the policy under consideration and is useful when a policy is being created graphically or the imported one is modified. The specification can be used as a guidance for writing correct specifications. The Specification Diagnosis component is used to check the syntax of a specification file loaded in this module. If it is correct, confirmation of correctness will show up as the last line in green along with the original specification content in a pop-up window. Otherwise, information on how to correct the errors will be given under each problematic line (Figure 3.8 (a)). The Exercise module has two components for self-evaluation: Query and Test. The Query component has seven questions (Figure 3.8 (b)) to help the exploration of MLS policies. It provides answers to some frequently asked questions such as what are all sets of categories, what are the possible security levels and whether a specific subject has read or write permission to an object. The Test component provides a way to evaluate the understanding of clearance, category, relationships and permissions through 13 questions on policies in different scales. Users have to choose an answer to proceed to the next question. This can be used for in-class exercises or quizzes. Instructors will receive a student's answer, a grade on each question and overall grade via email. This component currently has an example set of questions covering the core aspects of the BLP model. Instructors may populate the test with their own questions by modifying an input text file.

3.2.3 Evaluation

The effectiveness of MLSvisual was tested in an evaluation conducted in a senior-level computer security class. The course introduces computer security topics of secure coding, access control, cryptography, key managements, etc. Access control includes the UNIX permissions, the DTE, the Bell-Lapadula, and the RBAC models. Students were given extra credit for the participation of the evaluation. They were asked to answer a series of assignment questions.

The effectiveness of MLSvisual was tested in an evaluation conducted in a senior-level computer security class. The course introduces computer security topics of secure coding, access control, cryptography, key managements, etc. Access control includes the UNIX permissions, the DTE, the Bell-Lapadula, and the RBAC models.

Students were given paper and pencil exercises on the BLP model as part of the regular course homework. For this first use of MLSvisual, students were additionally given an extra credit assignment that required use of the tool. The problem was to evaluate a simple policy via a series of questions and then complete a test using the `Test` module. After the students had submitted their solutions to the extra credit assignment problem, the instructor distributed a survey to the class. Completion of the survey was voluntary.

The MLSvisual evaluation consists of two components, 17 rating questions (Table 3.1) and 9 write-in comments. The first 14 questions (Q1-Q14) study the effects of MLSvisual. The choices are: 1:strongly disagree, 2:disagree, 3:neutral, 4:agree, and 5:strongly agree. The Q15, Q16 and Q17 study the time participants spent on the tool. The choices for Q15 are 1:less than 5 mins, 2:5-10 mins, 3:10-15 mins, 4:15-30 mins and 5:more than 30 mins. The choices for Q16 are 1:once, 2:1-3 times, 3:3-5

Table 3.1
Survey Questions (© 2014 ACM. Reprinted by permission.)

Number	Question
Q1	MLSvisual helped better understand BLP model
Q2	MLSvisual was helpful for my self-study
Q3	General graph's analysis mode illustrated the relationship between different SLs clearly
Q4	General graph's edit mode allowed easy creation/modification to policies
Q5	Object graph depicted the files' SLs directly
Q6	Whole graph helped better understand of policies
Q7	Representation and layout eased use of the tool
Q8	Colors helped understand BLP's information flow
Q9	The tool clearly depicted read/write among SLs
Q10	The tool helped realize BLP's limitations
Q11	The tool helped learn Principles of Tranquility
Q12	Feel prepared to policy design after using the tool
Q13	The tool helped understand what was't understood
Q14	RBACvisual enhanced the course
Q15	How long did it take you to understand the BLP model by using the software
Q16	How many times did you use the software
Q17	How long did you use this software in total

times, 4:5-10 times and 5:more than 10 times. The choices for Q17 are 1:less than 5 mins, 2:5-15 mins, 3:15-30 mins, 4:30-60 mins and 5:more than 1 hour. We collected 22 valid forms. The distribution of majors is as follows: 10 in Computer Science, 6 in Computer Engineering, 3 in Computer Systems Science, 1 in Software Engineering, and 2 undeclared.

3.2.3.1 General Discussion

Table 3.2 shows the mean and standard deviation of each question. Feedback from participants was positive with an overall mean of 3.77 and standard deviation of 0.73. Q3 and Q8 received the highest scores of 4.2 and 4.3 with standard deviation 0.8 and 0.6, respectively. This indicates that the `General graph` showed the relationship among security levels clearly and that the use of colors helped students understand the BLP model. Q5 and Q11 received the lowest score 3.0. Q5 investigates whether the security levels of objects are straightforward in the `Object graph`. The low score may be because the `Object graph` and `General graph` were supposed to be used together. However, even if the security level assignment to the objects is visually presented, students probably treated them as separate and independent components, and hence Q5 received a neutral rating. Q11 received 3.0 because there is no direct visual presentation of this principle. Students have to edit a policy in several iterations to get hands-on experience of whether the strong or weak tranquility principle should be preserved. The `Edit mode` is designed for this purpose. The other questions received scores around 4.0. Hence, the general response to the tool was positive and participants considered that the tool helped them understand the concepts and enhanced the course.

Of the three usage questions (Q15-Q17), Q17 had an average of 3.6, which indicated that students used the tool for 15 to 30 minutes. The average of Q15 was 2.9 which means that it took around 10-15 minutes for students to understand the BLP model using MLSvisual. The average of Q16 was 1.5 showing that students used the tool once or twice. Table 3.3 has the distribution of answers to these three questions. Q15 had 9%, 23% and 41% of students select Choice 1, Choice 2 and Choice 3, respectively. Thus, 73% of all students required less than 15 minutes to understand the BLP model. Since no student selected Choice 5, all of them understood the BLP model within 30 minutes. The answer distribution of Q16 indicated that 50% of all students used it only once while the rest used MLSvisual twice. For Q17, 87% of all students selected among Choice 1 to Choice 4, which means that 87% of all students spent less than one hour using the tool.

Table 3.2
Mean (μ) and Standard Deviation (σ) (© 2014 ACM. Reprinted by permission.)

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16	Q17
μ	4.0	3.8	4.2	4.1	3.0	3.7	3.7	4.3	3.8	4.0	3.0	3.6	3.6	3.9	2.9	1.5	3.6
σ	0.6	0.7	0.8	0.7	0.9	0.6	0.7	0.6	0.9	0.7	0.9	0.6	0.8	0.5	0.9	0.5	1.0

Table 3.3
Usage Distribution (© 2014 ACM. Reprinted by permission.)

	Choice1	Choice2	Choice3	Choice4	Choice5
Q15	9%	23%	41%	27%	0
Q16	50%	50%	0	0	0
Q17	5%	9%	23%	50%	13%

We also looked at the correlations between each pair of questions from Q1 to Q14. The ratings of each question are loosely positively related with the highest correlations 0.65 for (Q3, Q10) and 0.64 for (Q7, Q8). The correlation between Q3 and Q10 suggested that those who considered the `Analysis mode` showed the relationship among security levels clearly (Q3) also tended to believe that MLSvisual helped them realize the BLP's limitations (Q10). For Q7 and Q8, those who considered the representation and layout made the use of MLSvisual easy (Q7) also might feel that the color scheme helped them understand the information flow of the BLP model (Q8). There are some other pairs having correlations around 0.55. The correlation between (Q3, Q4) was 0.56, indicating that students who liked the `Analysis mode` of the `General graph` (Q3) also rated the `Edit mode` of the `General graph` (Q4) higher. The correlation 0.55 of (Q6, Q10) suggested that students who rated the `Whole graph` (Q6) higher might find it easier to realize the limitations of the BLP model (Q10). The correlations between (Q1, Q13) and (Q2, Q13) were 0.52 and 0.55, respectively. This suggested that many students who felt that MLSvisual helped them understand what was not understood also tended to consider the tool beneficial to self-study and a better understanding of the BLP model.

3.2.3.2 Statistical Analysis

We used MANOVA and ANOVA to investigate if the use of the tool may affect student ratings. The level of significance is $\alpha = 0.05$. The null hypothesis for this study is: the time spent on understanding the BLP model (Q15), the number of times using this tool (Q16), and the total time spent on this tool (Q17) did not affect the answers to the 14 questions (Q1-Q14). Based on the answers to Q15, we divided students into 3 groups. Group 1 had students who spent less than 10 minutes to understand the model. Group 2 spent 10 to 15 minutes, and group 3 spent more than 15 minutes. The p-value of a MANOVA Wilk's lambda test was 0.525, suggesting that there was no significant difference among these groups. To verify the result, we also used ANOVA to perform individual test against Q15, and found that Q5 vs. Q13 had the smallest p-values 0.051. Since it is still larger than the level of significance, we can not reject the null hypothesis.

Students were divided into two groups according to their responses to Q16. The first group had 11 students who used the tool only once. The second group had the other 11 students who used the tool twice. The p-value of a MANOVA Wilk's lambda test was 0.677, which indicated that the null hypothesis can not be rejected. ANOVA tests against Q16 showed that Q1 and Q12 had the two smallest p-values 0.062 and 0.070, respectively. Since they are still greater than the significance level, the null hypothesis can not be rejected.

For Q17, we divided students into 2 groups. The first group included 8 students who used the tool for less than 30 minutes while the other group of 14 students spent more than 30 minutes. The MANOVA Wilk's lambda test had a p-value of 0.332, and the null hypothesis can not be rejected. ANOVA tests against Q17 showed that the p-value for Q13 (0.0046) was the only one less than the significance level. The null hypothesis was rejected. Therefore, students who spent less than 30 minutes and students who spent more than 30 minutes responded to Q13 differently. This happened because students used the tool after learning the BLP model in class. The parts they did not understand before were some challenging aspects. The different responses showed that many students were able to understand the challenging parts after spending enough time on the tool. Based on the findings, we have sufficient evidence to claim that the time students spent on the tool affects whether they could understand the parts that they did not understand before. But, in general, the use of the tool does not affect student rating when all questions are considered at the same time.

3.2.3.3 Student Comments

The set of 9 write-in questions was designed to gather suggestions from students for future improvement. The aspects we investigated are: whether the graph presentation is helpful, the `Specification diagnosis` module, the `Test module`, the use of colors and user interface, features to add and the software installation issues.

Student feedback was quite positive to the graph presentation. Some students said "*It clearly illustrated the lattice formed by the policy, and helped me see the relationship between levels*", "*The graph was very nice and definitely helped me understand the BLP model better*", "*The graph showed useful information with button to auto-generate*", and "*It worked perfectly as I imagined*". Therefore, we believe that the graph presentation did help students understand the BLP model better.

The comments on the `Specification diagnosis` model were generally positive. Students mentioned that "*It was definitely useful*" and "*It was a nice addition to the visual*". However, some students mentioned that they were not sure whether they had used the module. This is understandable since the extra credit assignment did not include the use of this module.

The `Test` module received positive feedback. Students mentioned that "*I was impressed by how well the software handled examples*" and "*The most populated object graph was nice*". A suggestion "*It would be better if there were answers to the questions at the end of the test*" was also mentioned. Since instructors usually use the module as a quiz, the questions can be answered on demand in class.

All students were satisfied with the use of colors and the user interface. A student suggested that "*Queries should default to a pop-out window*". Most of them did not think about additional features; however, one student suggested that "*Maybe a quick run down on the model and particular specification*". No software installation problem was reported.

Students also provided some general comments for further improvement. They suggested adding tooltip to all buttons, having the larger default window size, and providing a version for 64-bit Linux since some of their systems were not 32-bit compatible and needed some packages installed before use.

3.3 RBACvisual

RBACvisual[2] is another user-level visualization tool designed to facilitate the study and teaching of the role-based access control (RBAC) model, which has been widely used in companies to restrict access to authorized users. The tool provides two graphical abstractions of the underlying specification. Policies can be input and modified graphically or using text-based files. Students can use an embedded Query system to answer commonly asked questions and to test their understanding of a given policy. A Practice subsystem is also provided for instructors to assign quizzes to students;

[2]The material contained in this section was previously published in proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education [76]

the answers can be sent to the instructor via email. We also conducted an evaluation of RBACvisual within a senior-level course on information security. The student feedback was positive and indicated that RBACvisual helped students understand the model and enhanced the course.

3.3.1 Language

We designed the language for RBAC policies based on the elements involved in the model: roles, users and objects, where users are assigned to roles based on their job functions; roles are associated with objects through permissions. Also, roles can be in a hierarchical relation, which is represented by the statements starting with the keyword `inheritance`. Statements with keywords `user` and `object` specifies the users' assignment to roles and roles' permissions to objects. The syntax of the specification is shown below:

1) Role Statement

   Define one or more roles, which are available to other parts of an RBAC specification. Role Statement with keyword `inheritance` can be followed by one role and a list of role names from low to high seniority separated by `<`.

   `Syntax format: inheritance: role1<role2<...`

2) User statement

   Define the assignment of users to roles. User statement starts with keyword `user` followed by a list of users separated by commas. It means that the users in the list are assigned to the role in the statement.

`Syntax format:  user:  role user1, user2...`

3) Object statement

Define the permissions a role has to a list of objects that are delimited by commas. The statement starts with keyword `object`, then followed by the name of a role, a permission set, an optional recursive flag and a list of objects. The permission set `permSet` is a subset of {`r`, `w`, `x`}, where `r, w, x` represent read, write, and execute permissions, respectively.

`Syntax format:  object:  role permSet [-r] object1, object2...`

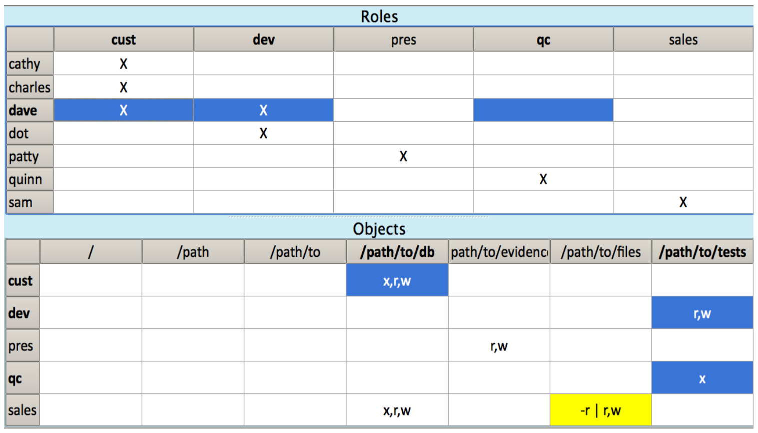
Figure 3.9 has an example of RBAC policy.

```
inheritance:  ADMIN > OSTEACH
inheritance:  ADMIN > SECTEACH
inheritance:  OSTEACH > OS
inheritance:  SECTEACH > SECURITY
user: ADMIN      alan
user: OSTEACH    tina
user: SECTEACH   tina
user: SECURITY   sally, sam
user: OS         oscar, sally
object: ADMIN         r,w,x  -r  /
object: OSTEACH       r,w,x  -r  /cls/os
object: SECTEACH      r,w,x  -r  /cls/security
object: SECURITY      r,x    -r  /tools
object: OS        r     -r     /cls/os/public
object: OS        r,x   -r     /tools
```

Figure 3.9: RBAC policy

3.3.2 Visualization System

RBACvisual also has the specification file (*.rbac) and visualization file (*.rbacvis) as input and output files. These two types of file can be converted to each other using the visualization system. RBACvisual implements the RBAC model in Core and Hierarchical forms [31]. The basic concept of Core RBAC is that users as well as permissions to objects (files and directories) are directly assigned to roles based on their job functions. Therefore, users' membership to roles determines if they have access to objects in the system.



Roles

	cust	dev	pres	qc	sales
cathy	X				
charles	X				
dave	X	X			
dot		X			
patty			X		
quinn				X	
sam					X

Objects

	/	/path	/path/to	/path/to/db	path/to/evidenc	/path/to/files	/path/to/tests
cust				x,r,w			
dev							r,w
pres					r,w		
qc							x
sales				x,r,w		-r | r,w	

Figure 3.10: User Interface with Matrix View (© 2015 ACM. Reprinted by permission.)

RBACvisual has two different views: the Matrix View and the Hierarchy View. Figure 3.10 has an example of the Matrix View. The top matrix is for the user-to-role assignment and the bottom matrix shows the role-to-object permissions.

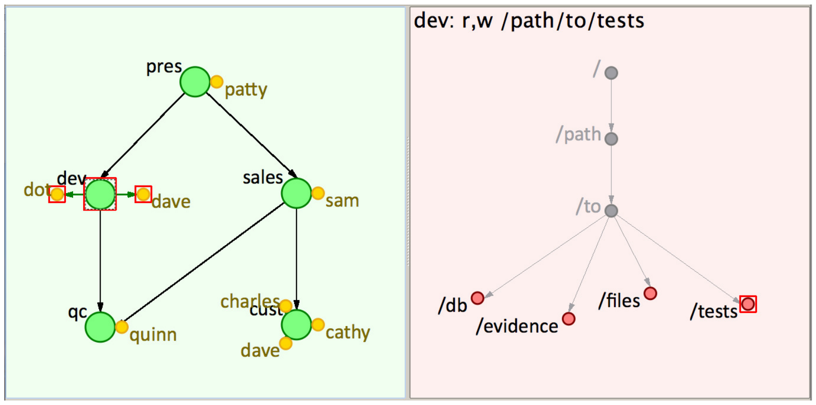
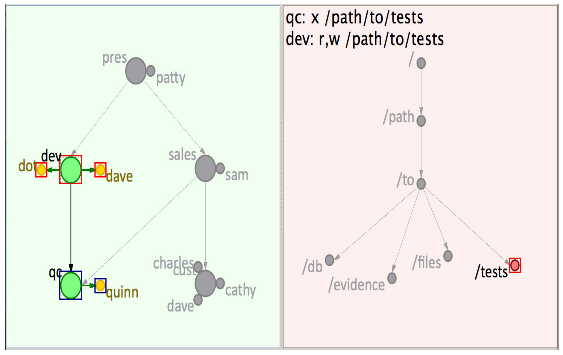


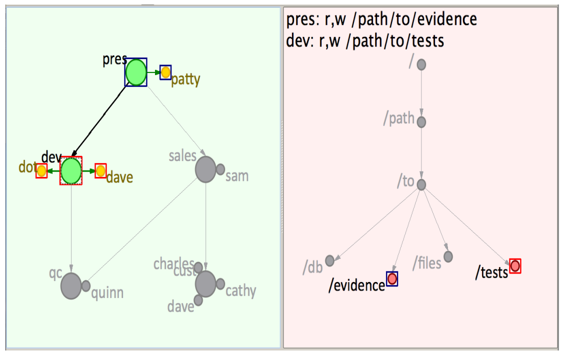
Figure 3.11: Role Node Highlight without Inheritance (© 2015 ACM. Reprinted by permission.)

Figure 3.11 shows the Hierarchy View, which consists of two parts. The Role Hierarchy Section with green background constructs a graph based on the role hierarchy. The Object Hierarchy Section with red background shows the hierarchy of objects in the file system. Green nodes represent roles, red nodes represent objects, and yellow nodes representing users are located around their role nodes. An edge is drawn from node r_1 to node r_2 when node r_1 inherits node r_2 (Edges inferred by transitivity in the role hierarchy are removed to reduce visual clutter.). All inheritance relationships are extracted from the policy and are depicted by an edge. If the inheritance is not specified explicitly in the policy file, the edge line is dashed.

RBACvisual supports highlight schemes with and without role hierarchy. When the hierarchy is included, highlighting shows the users assigned to the role and the objects the role has access to from itself as well as through the inheritance relation. When the hierarchy is not included, highlighting only shows information from the explicit definition of the role itself.


qc: x /path/to/tests
dev: r,w /path/to/tests

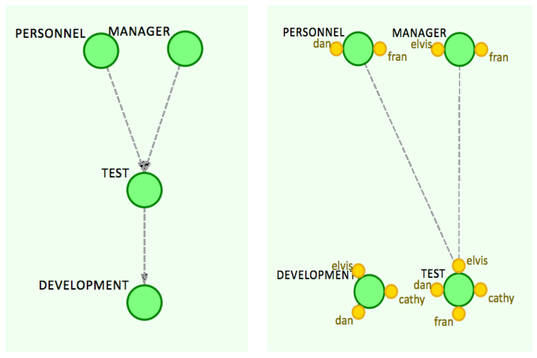
(a)


pres: r,w /path/to/evidence
dev: r,w /path/to/tests

(b)

Figure 3.12: Role Node Highlight. (a) shows the highlight with Children. (b) shows highlight with Parents. (© 2015 ACM. Reprinted by permission.)

When highlighting with role inheritance involved, choices of highlighting the children, parents, or both children and parent nodes of the clicked role node are available. The clicked node will be highlighted in a red frame, and role nodes with the selected inheritance relation will be highlighted in blue frames. User nodes and object nodes will be highlighted in red frames if directly accessible from the clicked role or in blue frames if accessible from blue-framed roles. Figures 3.12 (a) and (b) depict the nodes related to role dev. The highlighting in the left view shows that the child and parent role nodes of dev are qc and pres, respectively. The right view shows that dev and qc both have access to /path/to/tests with different permissions and pres additionally has access to /path/to/evidence. Likewise, when a user node is clicked, the roles it is assigned to and the objects accessible from those roles will be highlighted. When an object node is clicked, the roles and users that have access to the object are highlighted.

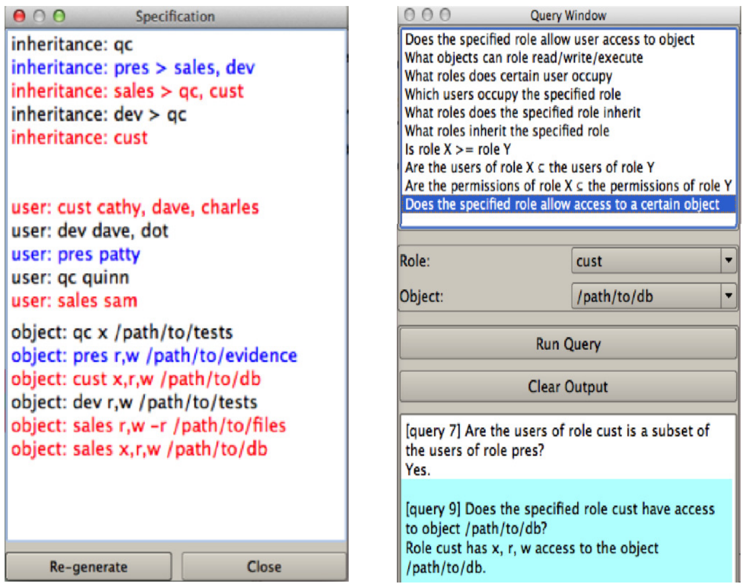


(a) (b)

Figure 3.13: Edit mode. (a) shows the role hierarchy before an edit. (b) shows the role hierarchy after the edit. (© 2015 ACM. Reprinted by permission.)

Both views allow building a policy from scratch and editing the policy graphically. In the Matrix View, the table cell values can be changed. In the Hierarchy View, a context menu (not shown) can be used for editing the properties of each node. The toolbox provides a dialog to modify, add, or delete any element of the user-to-role or permission-to-role assignment. Figure 3.13 shows the role hierarchy of a policy before and after user assignment to roles.

The Specification Window in Figure 3.14 (a) shows the text-based specification of the existing policy, which can be edited via graphical operations on views or textual edit within the window. The textual highlight shows the related nodes within red and blue frames in the highlight mode. The Query Window in Figure 3.14 (b) contains questions commonly asked about an RBAC policy. Parameters for certain questions can be configured on the interface and answers to questions can be found in the bottom field, with the most recent answer being highlighted.



```
inheritance: qc
inheritance: pres > sales, dev
inheritance: sales > qc, cust
inheritance: dev > qc
inheritance: cust


user: cust cathy, dave, charles
user: dev dave, dot
user: pres patty
user: qc quinn
user: sales sam

object: qc x /path/to/tests
object: pres r,w /path/to/evidence
object: cust x,r,w /path/to/db
object: dev r,w /path/to/tests
object: sales r,w -r /path/to/files
object: sales x,r,w /path/to/db
```

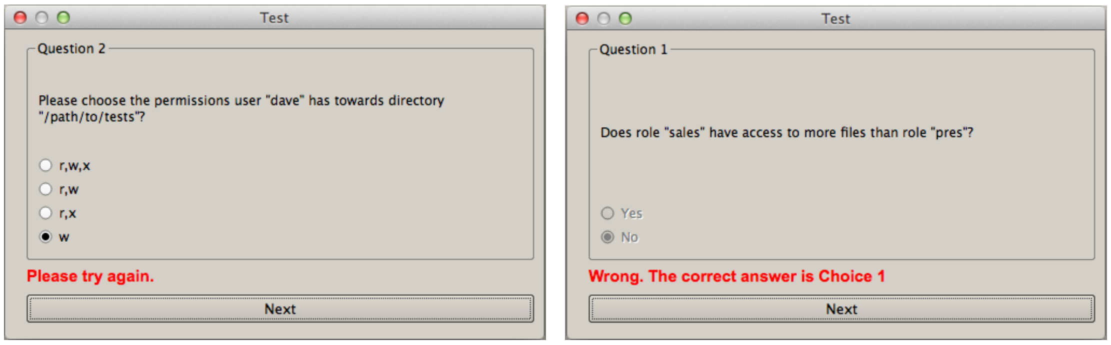
```
Does the specified role allow user access to object
What objects can role read/write/execute
What roles does certain user occupy
Which users occupy the specified role
What roles does the specified role inherit
What roles inherit the specified role
Is role X >= role Y
Are the users of role X ⊆ the users of role Y
Are the permissions of role X ⊆ the permissions of role Y
Does the specified role allow access to a certain object

Role:    cust
Object:  /path/to/db

[query 7] Are the users of role cust is a subset of
the users of role pres?
Yes.

[query 9] Does the specified role cust have access
to object /path/to/db?
Role cust has x, r, w access to the object
/path/to/db.
```

(a) (b)

Figure 3.14: Specification and Exercise Modules. (a) shows the Specification Diagnosis Window. (b) shows the Query Window. (© 2015 ACM. Reprinted by permission.)

RBACvisual allows an instructor to give a series of questions (or a "quiz") to students. Three quiz modes are available which control how a student may progress through the questions. The questions are configurable so that instructors can use their own questions to achieve various teaching goals. All the questions are multiple-choice questions. Instructors specify the quiz mode and the questions that comprise the quiz through a file that adheres to a prescribed format (given in the Instructor Manual).

Instructors can share the question file with their students and a test can be started by importing the question file into the system through a dialog.


Test

Question 2

Please choose the permissions user "dave" has towards directory "/path/to/tests"?

- r,w,x
- r,w
- r,x
- [✓ w]

Please try again.

Next

Test

Question 1

Does role "sales" have access to more files than role "pres"?

- Yes
- [✓ No]

Wrong. The correct answer is Choice 1

Next

(a) (b)

Figure 3.15: Quiz Mode. (a) shows Multiple Trial Quiz Mode with Wrong Answer. (b) shows Self-test Quiz Mode with Wrong Answer. (© 2015 ACM. Reprinted by permission.)

RBACvisual supports three quiz modes. The first mode is Traditional Mode where students' answers will be sent at the end of the quiz. The quiz moves forward after the first response to each question. The second mode is Multiple Trial Mode (Figure 3.15 (a)). In this mode, students are allowed to try multiple times until they get the correct answer to a question. The number of attempts for each question will be stored. The third option is Self-test Mode (Figure 3.15 (b)). Correct answers will be shown to the students after a choice has been confirmed for a question.

3.3.3 Evaluation

The RBACvisual evaluation included two parts: 18 rating questions (Table 3.4) and seven write-in comments. The first 15 rating questions study the effects of RBACvisual. The choices are: 1:strongly disagree, 2:disagree, 3:neutral, 4:agree, and 5:strongly agree. Q16, Q17 and Q18 study the time participants spent on the tool. The choices for Q16 are 1:less than 5 mins, 2:5-10 mins, 3:10-15 mins, 4:15-30 mins and 5:more than 30 mins. The choices for Q17 are 1:once, 2:1-3 times, 3:3-5 times, 4:5-10 times, and 5:more than 10 times. The choices for Q18 are 1:less than 5 mins, 2:5-15 mins, 3:15-30 mins, 4:30-60 mins, and 5:more than 1 hour. This evaluation was conducted in a senior-level Computer Security course. For this first use of RBACvisual, students were given extra credit if they used the tool to solve their assignment questions. A survey was distributed at the end of the semester for students to participate in voluntarily. We collected eight valid forms from students, five of whom major in Computer Science, one in Computer Systems Science, one in Software Engineering, and one in Computer Engineering.

3.3.3.1 General Discussion

Table 3.5 has the means, standard deviations and confidence intervals (at 95% significance level of mean) of rating questions Q1 to Q15. The ratings of questions are

Table 3.4
Rating Questions

Q1	Matrix View helped understand RBAC
Q2	Hierarchy View helped understand RBAC
Q3	Toolbox made it easy to create/edit policy
Q4	Context Menu in Hierarchy View is convenient for policy editing
Q5	Query helped study RBAC policy
Q6	RBACvisual made correct modification on policies easier
Q7	Matrix View was intuitive and clear
Q8	Hierarchy View was intuitive and clear
Q9	Hierarchy View helped understand role inheritance
Q10	Colors used can distinguish different items
Q11	Width of edges was reasonable
Q12	Understood RBAC better after using the tool
Q13	The tool helped find mistakes in my policy
Q14	RBACvisual enhanced the course
Q15	The software was easy to use
Q16	How long did it take you to understand the RBAC model by using the software
Q17	How many times did you use the software
Q18	How long did you use this software in total

no less than 3.88. Their overall mean value is 4.34 with a standard deviation 0.69, suggesting that the feedback to the tool was positive in general. Q6 and Q13 have the lowest mean of 3.88 with standard deviation of 0.99 and 0.64, respectively. Q6 investigates whether the tool makes the correct modification of policies easier. The lower scores it received might be because some modifications did not introduce big changes in visualization and thus some efforts should be taken to examine the changes. Q13 probably shares the same reasoning when changes are applied and it is hard to tell the correctness of a change as it depends on the users' intention, which is hard to detect. The means and confidence intervals of Q7 and Q8 are 4.29 and 4.57, (3.92, 4.65) and (4.18, 4.97), indicating that students generally thought the Matrix View and the Hierarchy View were intuitive and clear. Q1, Q2, Q12 and Q14 received scores no less than 4.13. This suggests that RBACvisual helped students understand the RBAC model better and enhanced the course. Q3, Q4 and Q15 on the easiness of using the tool were rated over 4.25 and thus showed that the tool was easy to use.

Table 3.5

Mean (μ), Standard Deviation (σ) and Confidence Interval (CI^-, CI^+)(© 2015 ACM. Reprinted by permission.)

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15
μ	4.38	4.25	4.38	4.25	4.38	3.88	4.29	4.57	4.86	4.57	4.29	4.13	3.88	4.25	4.63
σ	0.52	0.89	0.74	0.89	0.74	0.99	0.49	0.53	0.38	0.79	0.76	0.64	0.64	0.46	0.52
CI^-	4.02	3.64	3.86	3.64	3.86	3.19	3.92	4.18	4.58	3.99	3.73	3.68	3.43	3.93	4.27
CI^+	4.73	4.86	4.89	4.86	4.89	4.56	4.65	4.97	5.00	5.00	4.85	4.57	4.32	4.57	4.98

The last three questions (Q16 to Q18) are about the usage of the tool. Table 3.6 has the distribution of answers. On Q16, 62.5% of students selected Choice 2 and 37.5% chose Choice 4. This implies that all students were able to understand the RBAC model within 15 minutes. The distribution of Q17 indicates that half of the students used the tool for one to three times and all of them used the tool for less than 5 times. Answers to Q18 suggest that 75% of the students used the tool for less than 30 minutes while there were some students who used the tool for up to one hour.

Table 3.6

Usage Distribution (© 2015 ACM. Reprinted by permission.)

	Choice1	Choice2	Choice3	Choice4	Choice5
Q16	0	62.5%	37.5%	0	0
Q17	12.5%	50%	37.5%	0	0
Q18	12.5%	25%	37.5%	12.5%	12.5%

3.3.3.2 Statistical Analysis

We were interested in knowing the rating correlation of each question pair. To this end, the Spearman rank correlation test was applied to the first 15 questions. We found 16 out of the 105 question pairs had a p-value less than the level of significance $\alpha = 0.05$. This means that nearly 85% of the question pairs did not have a significant monotonic correlation. Moreover, all Spearman ρ's between Q10 and other questions were insignificant, meaning the rating of the use of colors is likely to be independent of the rating of other questions. Figure 3.16 shows the 16 pairs with the value of ρ shown on each edge. It is clear that the ratings of Q2, Q3, Q5, Q6, Q8 and Q11 were very closely inter-related with a Spearman ρ value of at least 0.78. Therefore, the Hierarchy View, Query, easy policy modification, and policy creation/editing were rated similarly in a monotonic way. Q5, Q12, Q13 and Q9 formed a linear chain with $\rho(\text{Q5}, \text{Q12}) = 0.839$, $\rho(\text{Q12}, \text{Q13}) = 0.820$ and $\rho(\text{Q9}, \text{Q13}) = 0.764$. This indicated that if a student rated "Query helped study RBAC policy" (Q5) higher this student would very likely provide higher ratings to questions "RBACvisual helped understand RBAC better" (Q12), "RBACvisual helped find mistakes" (Q13), and "Hierarchy View helped understand role inheritance" (Q9). It is interesting to note that the Spearman ρ between "RBACvisual enhanced the course" (Q14) and "RBACvisual was easy to use" (Q15) is 0.4 with a p-value of 0.374. As a result, we cannot reject the null hypothesis, which means there was no statistically significant monotonic correlation between the rating of Q14 and the rating Q15. On the other hand, the two high Spearman ρ dangling pairs $\rho(\text{Q6, Q14}) = 0.882$ and $\rho(\text{Q3, Q15}) = 0.794$ were perhaps coincidences.

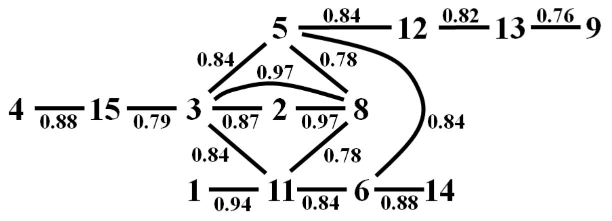


Figure 3.16: Graph of Significant Spearnman Correlation Pairs (© 2015 ACM. Reprinted by permission.)

We also used a Student's t-test to compare the differences among ratings. While the sample size is small, Student's t-test is rather robust and still can be used in this study [25]. We first looked at the "helped" question group (Q1, Q2, Q5, Q9, Q13). Pairwise t-test shows that except for pairs (Q5, Q9) and (Q9, Q13) with p-values 0.03 and 0.00, respectively, all other p-values were larger than 0.1. This suggested that except for (Q5, Q9) and (Q9, Q13), the null hypothesis (that the questions were rated equally) cannot be rejected. The p-value for pair (Q7, Q8) is 0.17, and, hence, students rated the Matrix View and the Hierarchy View equally even though the means were 0.429 and 0.458, respectively. Finally, we looked at three summary questions "Understood RBAC after using the tool" (Q12), "RBACvisual enhanced the course" (Q14), and "The software was easy to use" (Q15). The mean values of Q12, Q14 and Q15 were 4.13, 4.25 and 4.63, respectively, and the p-values for (Q12, Q14), (Q12, Q15) and (Q14, Q15) were 0.60, 0.03 and 0.08, respectively. Therefore, the rating difference between Q12 and Q15 is statistically significant, and students considered ease of use higher than improved understanding of RBAC after using the tool. Since only 13 out of 105 pairwise t-tests were significant and many question pairs were not directly related, the rating differences would be small. Coupled with high ratings of questions, we conclude that the evaluation results were very positive for this sample.

3.3.3.3 Student Comments

The seven write-in questions were designed to gather suggestions from participants for further improvement. The aspects include: representation in visualizations, the effects of in-class demo of the tool, new feature suggestions, and performance and installation of the tool.

The overall feedback to visualization representations was positive. Some students stated "*I enjoy this view when looking at who has permissions quickly. I can click on what I need to know and it will light up anything corresponding to.*", "*This was the best part. The hierarchy showed the role dominance and which users belonged to which roles very clearly.*", and "*The hierarchy view helped me understand what roles are ranked higher and lower than one another.*" Some issues were mentioned: (1) to add header scrolling in the matrix view; and (2) presentation of permissions to objects that fits the visual theme better than the text presentation.

The in-class demo received neutral feedback. For the students who sent evaluation forms back after final exams, it was hard to remember the in-class demo afterwards, and they generally gave a neutral feedback. For the feedback received on time, the feedback was positive. Students mentioned "*I think the most advantage is [that] I am involved and get a helpful feedback quickly.*", and "*I think the most helpful part is being able to click on elements and see the relations between roles, users and objects.*"

Students also provided some general comments for further improvement. They suggested: (1) the Matrix View should have multiple selections that allow comparisons; (2) Keyboard shortcuts should be supported; and (3) the specification should be directly editable in the Specification Window. No installation issues were reported.

In summary, we found that students who rated "Query helped study RBAC policy" tend to give high ratings to "RBACvisual helped understand RBAC better", "RBACvisual helped find mistakes" and "Hierarchy View helped understand role inheritance". We also found that students rated the Matrix View and the Hierarchy View equally. Combined with the high ratings of questions and students comments, we believe that RBACvisual effectively helped students understand and the instructor teach the RBAC model better with intuitive visual representation.

3.4 UNIXvisual

UNIXvisual[3] is the fourth visualization tool in the series designed to facilitate the study and teaching of access control in UNIX. UNIXvisual is aimed both at novice users, who need only to control access to their own files, and students of computer security, who need a deeper and more comprehensive understanding. The system allows students to analyze permission settings in the underlying real file system, as well as in a combination of real and pseudo file systems defined through a specification file. It also allows a student to trace the value and effect of credentials within an executing process. UNIXvisual gives instructors flexibility in the allocation of lecture time by supporting self-study, it lowers the overhead required for teaching access control by running under an ordinary user account, and it enhances learning through the usage of visualization.

3.4.1 Language

The language designed for UNIX policies is based on the mechanism of access decision making in UNIX permissions. The policy uses a root directory to specify the starting point to extract the users, groups and permission to objects from the underlying file

[3] The material contained in this section was previously published in proceedings of the 2016 and 2017 ACM Conference on Innovation and Technology in Computer Science Education [74, 75]

system. The policy can also be used to define a hypothetical file system through User, Group and Object Statements. Figure 3.17 shows an example policy that can be processed by UNIXvisual. The syntax of the statements is listed below:

```
root: ./pseudoroot
user: tony, mike, lucy, mary
user: jim
group: manager      mary, jim
group: tester       mike, lucy
group: developer    tony, lucy
object: drwsrwxr--  tony:developer  -r  /code
object: rwxrwSr--   tony:developer      /code/program1
object: drwxrw-r--  mary:manager    -r  /document
object: drwxrw-r--  mary:manager    -r  /document/projectA
object: drwxrw-r--  jim:manager     -r /document/projectB
object: drwxrwx--x  mike:tester     -r /test
object: drwxrwxr-x  mike:tester     -r /test/projectA
```

Figure 3.17: UNIX policy (© 2017 ACM. Reprinted by permission.)

1) Root Statement

   Define the root directory for the visualization system as the starting point to extract the real file system information. `rootDirectory` must exist in the real file system.

   `Syntax: root: rootDirectory`

2) User Statement

   Define one or multiple pseudo users. The users can later be used in the Group and Object Statements. The statement starts with keyword `user`, followed by a list of users.

   `Syntax: user: user1, user2...`

3) Group Statement

   Define the user members of a group. The statement starts with keyword `group`, followed by a `groupName` and a list of users.

   `Syntax: group: groupName user1, user2...`

4) Object Statement

   Define the access to objects in a similar format with the output of command `ls -l`. The statement starts with keyword `object`. Object's nine permission bits and user and group owners are followed. Then there are an optional recursive flag and the object's path.

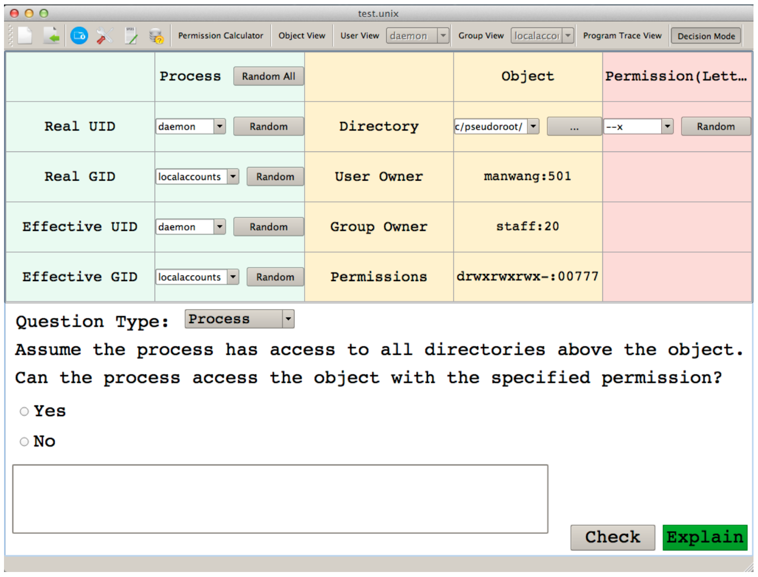
   `Syntax: object: permBits userOwner:groupOwner [-r] objectPath`

3.4.2 Visualization System

The visualization illustrates the process of determining the access a user or group has to objects. In this, UNIXvisual supports four main perspectives. The Decision Mode View illustrates a single decision by the access control system, excluding directory traversal. The Object View explores which users and groups have access to a selected object. The User View and Group View explore the set of objects accessible to a user (through the user bits) or to a group (through the group bits) respectively.

3.4.2.1 Decision Mode

The Decision Mode aims to provide obvious access to an interactive question system in order to encourage students to test their understanding. The interface has two parts (Figure 3.18). The bottom part is the question and answer area where the question statement is presented and answers to the question can be chosen. The top part is the question parameter area where the parameters of the question to ask can be configured by users. This mode provides two types of commonly asked questions: (1) whether a process with certain credentials can access an object with a selected permission and (2) conversion between the letter notation and octal permission notations. Students can answer the question and check their correctness. An animation is also supported to provide a step-by-step explanation to guide students to the solution.



	Process	Random All		Object		Permission(Lett…	
Real UID	daemon	Random	Directory	c/pseudoroot/	...	--x	Random
Real GID	localaccounts	Random	User Owner	manwang:501			
Effective UID	daemon	Random	Group Owner	staff:20			
Effective GID	localaccounts	Random	Permissions	drwxrwxrwx-:00777			

Question Type: Process

Assume the process has access to all directories above the object.
Can the process access the object with the specified permission?

○ Yes

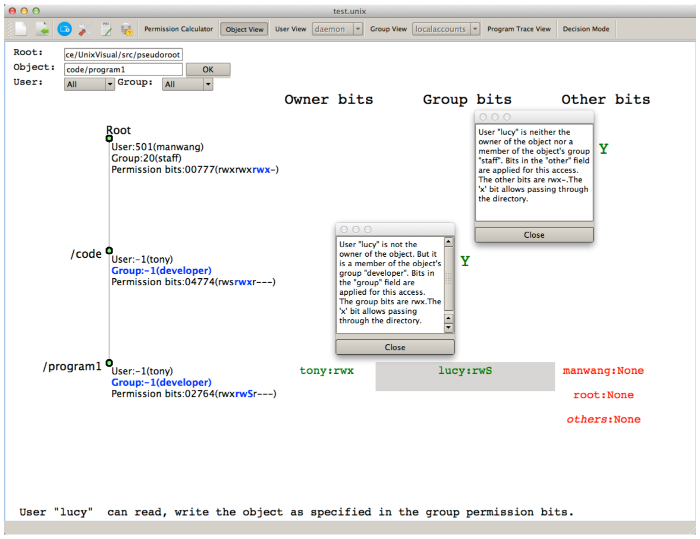
○ No

Figure 3.18: Decision Mode (© 2017 ACM. Reprinted by permission.)

3.4.2.2 Object View

The Object View asks for an object of interest and illustrates the determination of which users have access to the object. Figure 3.19 shows a snapshot of the Object View. The visualization shows the analysis in a matrix form. On the left, paths from the root directory to the target object are represented as nodes with permission information at each directory level. This defines the rows of the matrix. On the right, the permission bit groupings, "Owner bits", "Group bits" and "Other bits" define the columns.

By default, all users are considered for access to the object. Options of focusing on one user or one group are also supported. In the visualization, results of users' access are shown in the last row as color-coded user names. The columns in which a user name appears indicates the bits that will be applied at the object level. Clicking on a user name enables an analysis of that user's access. Color-coded letters of "Y" and "N" are placed in the row that corresponds to object level and the column that corresponds to the group of bits that are applied. In Figure 3.19, the user has selected lucy from the last row to obtain more information on the access lucy has to the object. Clicking on the letters of "Y" and "N" allows another level of detailed explanation of why these bits are applied and why the access is or is not allowed. This triple-layered analysis from color-coded user names to detailed explanation avoids showing the complete explanation all at once and thus encourages students to think about how the permissions work.



Root: ce/UnixVisual/src/pseudoroot
Object: code/program1
User: All Group: All

Root
User:501(manwang)
Group:20(staff)
Permission bits:00777(rwxrwxrwx-)

/code
User:-1(tony)
Group:-1(developer)
Permission bits:04774(rwsrwxr---)

/program1
User:-1(tony)
Group:-1(developer)
Permission bits:02764(rwxrwSr---)

User "lucy" is neither the owner of the object nor a member of the object's group "staff". Bits in the "other" field are applied for this access. The other bits are rwx-.The 'x' bit allows passing through the directory.

User "lucy" is not the owner of the object. But it is a member of the object's group "developer". Bits in the "group" field are applied for this access. The group bits are rwx.The 'x' bit allows passing through the directory.

tony:rwx

lucy:rwS

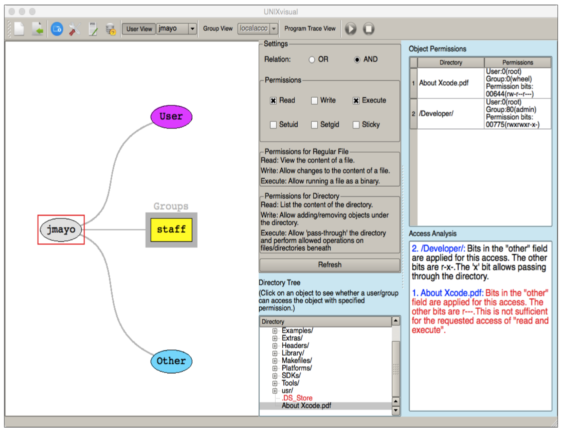
manwang:None
root:None
others:None

```
User "lucy"  can read, write the object as specified in the group permission bits.
```

Figure 3.19: Object View (© 2017 ACM. Reprinted by permission.)

3.4.2.3 User and Group View

The User and Group View illustrates the access allowed by a user or group through the file permission bits to objects under a user-specified directory. An example of the User View is given in Figure 3.20 (a). The visualization can be divided into two parts. The left part contains information about a user (above) or group. A user is represented as a node connected with three nodes to represent the owner, group and other permission bits. A group is represented as a node connected with all its member users (not shown). The right part has four sections. The top left window is the Permission Setting section.



Settings

Relation: OR [✓ AND]

Permissions

[✓ Read] Write [✓ Execute]

Setuid Setgid Sticky

Permissions for Regular File
Read: View the content of a file.
Write: Allow changes to the content of a file.
Execute: Allow running a file as a binary.

Permissions for Directory
Read: List the content of the directory.
Write: Allow adding/removing objects under the directory.
Execute: Allow 'pass-through' the directory and perform allowed operations on files/directories beneath

Refresh

Directory Tree
(Click on an object to see whether a user/group can access the object with specified permission.)

```
Directory
  ⊞ Examples/
  ⊞ Extras/
  ⊞ Headers/
  ⊞ Library/
  ⊞ Makefiles/
  ⊞ Platforms/
  ⊞ SDKs/
  ⊞ Tools/
  ⊞ usr/
    .DS_Store
    About Xcode.pdf
```

Object Permissions

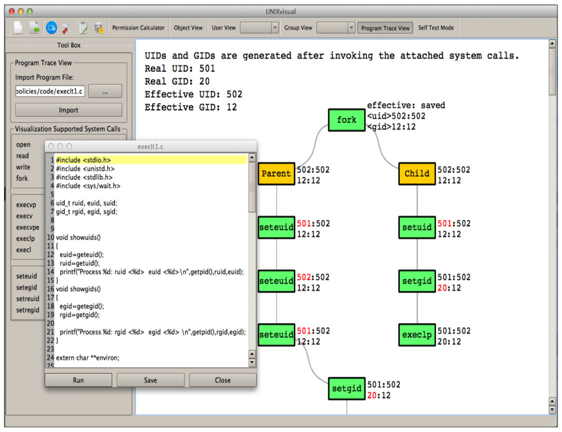
	Directory	Permissions
1	About Xcode.pdf	User:0(root) Group:0(wheel) Permission bits: 00644(rw-r--r---)
2	/Developer/	User:0(root) Group:80(admin) Permission bits: 00775(rwxrwxr-x-)

Access Analysis

2. /Developer/: Bits in the "other" field are applied for this access. The other bits are r-x-.The 'x' bit allows passing through the directory.

1. About Xcode.pdf: Bits in the "other" field are applied for this access. The other bits are r---.This is not sufficient for the requested access of "read and execute".

(a)



Import Program File: policies/code/execlt1.c

```
#include <stdio.h>
#include <unistd.h>
#include <stdlib.h>
#include <sys/wait.h>

uid_t ruid, euid, suid;
gid_t rgid, egid, sgid;


void showuids()
{
  euid=geteuid();
  ruid=getuid();
  printf("Process %d: ruid <%d>  euid <%d>\n",getpid(),ruid,euid);
}
void showgids()
{
  egid=getegid();
  rgid=getgid();

  printf("Process %d: rgid <%d>  egid <%d> \n",getpid(),rgid,egid);
}

extern char **environ;
```

```
UIDs and GIDs are generated after invoking the attached system calls.
Real UID: 501
Real GID: 20
Effective UID: 502
Effective GID: 12
```

```
fork      effective: saved
          <uid>502:502
          <gid>12:12
Parent    502:502
          12:12
Child     502:502
          12:12
seteuid   501:502
          12:12
setuid    501:502
          12:12
seteuid   502:502
          12:12
setgid    501:502
          20:12
seteuid   501:502
          12:12
execlp    501:502
          20:12
setgid    501:502
          20:12
```

(b)

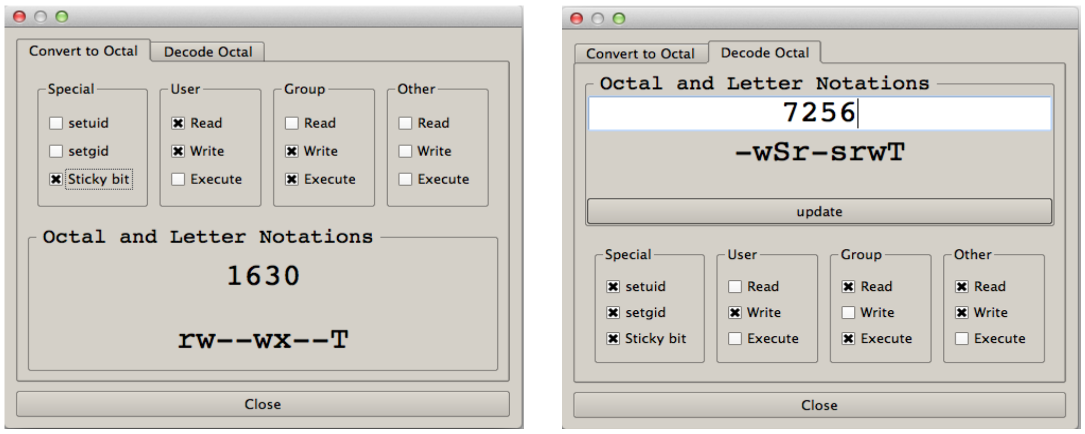
Figure 3.20: User and Group View and Program Trace View. (a) shows User and Group View. (b) shows Program Trace View. (© 2017 ACM. Reprinted by permission.)

In this section, students may choose the type of object access they want to investigate. The bottom left window is the Directory Tree section. It shows the object structure in a standard directory tree hierarchy. Directories are clickable to expand and contract for one directory level. The two windows on the right are the Object Permissions and Access Analysis sections. They are blank initially. Once a user/group is selected, each object in the object hierarchy window is checked against the specified permission in the Permission Setting section. If an object can be accessed by the user with the specified permission, the object remains black. Otherwise, it is shown in red. Clicking on an object in the Directory Tree enables the Object Permissions and Access Analysis windows which shows a detailed analysis. The permission information for the object selected from the Directory Tree and all directories up to the root directory is supplied in the Object Permissions section. An explanation of the access is given in the Access Analysis section. The analysis includes an explanation of which bits were applied and the access decision at the level.

3.4.2.4 Program Trace View

The Program Trace View is designed to help students understand initial assignment of credentials to a process, dynamic modification of credentials, and the effect of these credentials on an access request. This view allows the import of a C program and tracks process credentials across access control-related system calls, like open, fork, setresuid, read, write, etc. Figure 3.20 (b) shows an example of the visualization. After loading a C source code or binary, the initial real and effective user/group IDs are shown in the top left corner. Invoked system calls are depicted sequentially as blocks with effective and saved user/group IDs. The success of a system call is reflected in its block color; green indicates success and red indicates failure. The credentials on the side use red highlighting to indicate changes in credential values after the system call.

3.4.2.5 Permission Calculator



Special: [ ] setuid [ ] setgid [✓ Sticky bit]
User: [✓ Read] [✓ Write] [ ] Execute
Group: [ ] Read [✓ Write] [✓ Execute]
Other: [ ] Read [ ] Write [ ] Execute

Octal and Letter Notations

1630

rw--wx--T

Octal and Letter Notations

7256

-wSr-srwT

Special: [✓ setuid] [✓ setgid] [✓ Sticky bit]
User: [ ] Read [✓ Write] [ ] Execute
Group: [✓ Read] [ ] Write [✓ Execute]
Other: [✓ Read] [✓ Write] [ ] Execute

(a) Letter to Octal Notation (b) Octal to Letter Notation

Figure 3.21: Permission Calculator (© 2017 ACM. Reprinted by permission.)

Octal and letter notations are frequently used to specify UNIX permissions values through the command line. The conversion between these two notations can be tricky for beginners. The Permission Calculator is designed to help students learn different permission notations. Figure 3.21 (a) and (b) show the interface of the letter-to-octal and the octal-to-letter notation conversion. Both interfaces have three ways of expressing permissions: a matrix of checkboxes denoting permission bits, octal notation and letter notation. With the checkbox matrix and both notations side by side, it is easier to interpret the meaning of each bit and how each bit is expressed in different notations.

3.4.2.6 Query and Quiz

UNIXvisual also contains a Query mode and a Quiz mode. The Query mode includes a list of commonly-asked questions on UNIX permissions. Question parameters are configurable through the interface and answers to the questions are presented through guided visualization. This mode provides the convenience of having problems clarified outside of the classroom at any time. The Quiz mode provides an interactive environment for conducting quizzes. Text-based and visualization-based questions can be asked. All the questions are multiple-choice questions and can be configured to accommodate instructors' teaching goals.

The questions that comprise a quiz are written through a text file that adheres to a prescribed format. Students can start the quiz by loading the question file distributed by the instructor. Each question will have to be answered to move to the next one. At the end of the quiz, a dialog will show the location of the student's answer file and the student will be able to send the email to the instructor in order to prevent manual changes.

3.4.3 Evaluation

The evaluation was conducted in a required junior-level concurrent computing course with a total of 55 students. In a 75-minute session, students were asked to take a pre-test on UNIX permissions, followed by a 35-minute UNIX permissions lecture and a 15-minute demo of UNIXvisual. Students were allowed to use UNIXvisual in the following two weeks, and completed a post-test and an evaluation form. The lecture, the demo and the use of UNIXvisual only focus on the basics of UNIX permissions including the letter and octal notations, and the permission to objects without and with directory traversal. The whole process was conducted near the end of a semester on a voluntary basis. Students may not participate due to no attendance policy and were not required to use the software because some of them had learned UNIX permissions in Systems Programming and other courses. Students who completed the pre-test, post-test, and evaluation form were granted extra credits.

We collected 40 valid pre-tests, 44 valid post-tests, and 44 valid evaluation forms. We also collected 51 final exam papers, and recorded grades of the UNIX permissions section. Of the 44 students who submitted the evaluation form, 21 used UNIXvisual, 40 attended the lecture, and 38 submitted the pre-test, post-test and final exam.

3.4.3.1 Test Problems

The questions in the pre-test, post-test and final exam have the same form with the same level of difficulty. There are 10 questions in each test (1 point per question). Questions Q1 and Q2 (Group 1 or G_1) ask about conversion between the octal and letter notations of UNIX permissions. Questions Q3-Q6 (Group 2 or G_2) ask about access requests to an object without directory traversal. Questions Q7-Q10 (Group 3 or G_3) are about the access requests to an object with directory traversal.

Table 3.7
The Means (μ) and Standard Deviations (σ) of the Pre-test, Post-test, and Final Exam Questions (© 2017 ACM. Reprinted by permission.)

Pre-test											
	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Total
μ	0.80	0.78	0.98	0.88	0.88	0.80	0.34	0.88	0.37	0.44	7.05
σ	0.40	0.42	0.16	0.33	0.33	0.41	0.48	0.33	0.49	0.50	1.66
Post-test											
	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Total
μ	0.98	0.95	0.95	0.89	0.91	1.00	1.00	0.98	0.68	0.75	9.09
σ	0.15	0.21	0.21	0.32	0.29	0.00	0.00	0.15	0.47	0.44	1.29
Final Exam											
	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Total
μ	0.92	0.90	0.88	1.00	0.96	0.98	0.69	0.88	0.61	0.78	8.61
σ	0.27	0.30	0.33	0.00	0.20	0.14	0.47	0.33	0.49	0.42	1.60

Table 3.8
The Means (μ) and Standard Deviations (σ) of G_1, G_2 and G_3 in the Pre-test, Post-test and Final Exam (© 2017 ACM. Reprinted by permission.)

	Pre-test			**Post-test**			**Final Exam**		
	G_1	G_2	G_3	G_1	G_2	G_3	G_1	G_2	G_3
μ	0.79	0.88	0.51	0.97	0.94	0.85	0.91	0.96	0.74
σ	0.41	0.32	0.50	0.18	0.24	0.36	0.29	0.21	0.44

Tables 3.7 and 3.8 have the mean and standard deviation of each question and question group in these tests. The correctness of questions in G_1 and G_2 are above 91% in the post-test and the final exam, and G_3 in all three tests has the lowest means and the highest standard deviations. It is reasonable that G_3 received the lowest means as access request questions with directory traversal include more levels of permission checking and thus make the questions more challenging. Figure 3.22 depicts the group comparison of the three tests. It shows that 1) students' overall performance in all groups improved in the post-test and the final exam; and 2) students performed better in G_1 and G_3 in the post-test than in the final exam.

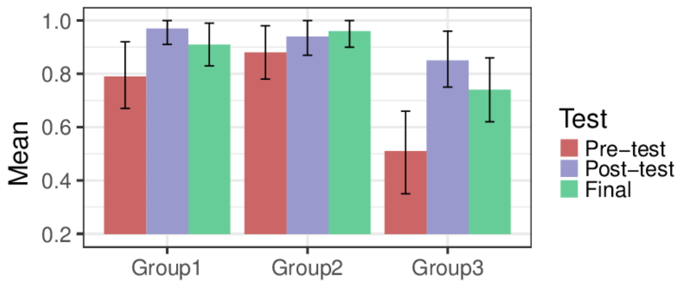


Figure 3.22: The Means with Confidence Intervals of G_1, G_2 and G_3 in the Pre-test, Post-test and Final Exam (© 2017 ACM. Reprinted by permission.)

Table 3.9 has the means and standard deviations of question scores of students who used UNIXvisual and who did not use UNIXvisual in all three tests. Students who used UNIXvisual received higher scores in all question groups in the post-test and the final exam than students who did not use UNIXvisual.

Table 3.9
The Means (μ) and Standard Deviations (σ) of G_1, G_2, G_3 and Total Scores of Students Who Used and Did Not Use UNIXvisual (© 2017 ACM. Reprinted by permission.)

Students Who Used UNIXvisual												
	Pre-test				Post-test				Final			
	G_1	G_2	G_3	Total	G_1	G_2	G_3	Total	G_1	G_2	G_3	Total
μ	0.84	0.86	0.46	6.95	0.98	0.98	0.92	9.52	0.95	0.99	0.88	9.35
σ	0.37	0.35	0.50	1.58	0.15	0.15	0.28	0.81	0.22	0.11	0.33	1.23
Students Who Did Not Use UNIXvisual												
	Pre-test				Post-test				Final			
	G_1	G_2	G_3	Total	G_1	G_2	G_3	Total	G_1	G_2	G_3	Total
μ	0.76	0.90	0.54	7.29	0.96	0.90	0.79	8.70	0.89	0.94	0.65	8.13
σ	0.43	0.30	0.50	1.65	0.21	0.30	0.41	1.49	0.32	0.25	0.48	1.65

3.4.3.2 Test Problems Analysis

In this part, significance tests were applied to find out 1) whether students' performance in the tests improved; and 2) whether UNIXvisual introduced the improvement. The significance tests include Student's t-test, ANOVA (parametric) and Kruskal-Wallis (KW) test (non-parametric), and repeated measures ANOVA (parametric) and Friedman test (non-parametric). We mainly used parametric methods with the non-parametric methods as backups. All significance tests were conducted at 95% significance level.

To evaluate students' performance throughout the tests, we first compared the pre-test and post-test using Student's t-test and KW test. Only Q1, Q2, Q6, Q7, Q9, Q10 and the total score had p-value less than 0.05. With their increased means from the pre-test to the post-test (Table 3.7), this indicates that the students' performance on notation conversion, access requests with directory traversal, and the total score had significantly improved. We also applied Student's t-test and KW test to compare the pre-test and the final exam. The results show that students performed differently in only Q4, Q6, Q7, Q9, Q10 and the total score. As the means of these questions increased (Table 3.7), the performance significantly improved on questions of access request without and with directory traversal and the total score in the final exam. The scores of the post-test and the final exam were also compared using the same method. The results indicate significantly improved performance in Q4 and declined performance in Q7, which means that the performance in other questions and the total score did not differ significantly. Therefore, for the declined performance in G_1 and G_3 from the post-test to the final exam in Figure 3.22, we know that the performance decline in G_1 is insignificant, and that Q7 is the only question that showed a decline in performance in G_3. Note that G_3 has the most challenging questions in the tests. Since there was no homework or project on UNIX permissions between these two tests, the decline in performance in Q7 was likely caused by students' reduced familiarity with the material over time due to a lack of practice.

To investigate the reason for the improvement throughout the tests, we looked into the students who submitted all three tests. As they participated in the UNIX permissions lecture and the UNIXvisual demo, and the use of UNIXvisual, this student group formed an important sample to assess the effect of the lecture with demo and the use of UNIXvsual on the scores of the tests. The repeated measures ANOVA and Friedman test were used, and the p-values of Q1, Q5-Q10, and the total score are less than 0.05. As the means of the total score of the post-test and the final exam are higher than that of the pre-test (Table 3.7), the lecture with demo and the use of UNIXvsual helped students perform better in the post-test and the final exam.

We further examined whether the use of UNIXvisual helped the improvement in the post-test and the final exam. We applied Students' t-test and KW test to the post-test question group scores of students who used UNIXvisual (21 students) and students who did not use the tool (23 students). The results show that G_2, G_3 and the total score had a p-value less than 0.05. With their means in Table 3.9, the performance of students who used UNIXvisual is significantly better than those who did not use the tool. We also divided students who took the final exam into a group of 20 students who used UNIXvisual and a group of 31 students who did not use the tool, and compared their performance using the same tests. G_3 and the total score had a p-value less than 0.05. Given their means in Table 3.9, students who used UNIXvisual performed significantly better than those who did not use the tool in G_3 and the total score. Lastly, we evaluated the background of students who used UNIXvisual and who did not use the tool by comparing their pre-test scores. The t-test and KW test show that the p-values for all question groups and the total score are greater than 0.05. Therefore, these two groups of students had similar background.

So far we have seen that UNIXvisual helped students improve significantly from the pre-test to the post-test, and that the improved performance continued in the final exam. Students who used UNIXvisual and those who did not use the tool had a similar UNIX permissions background. But students who used UNIXvisual made significant improvement and received higher scores in all question groups in the post-test and the final exam than students who did not use the tool. More specifically, the use of UNIXvisual significantly increased the scores of G_3 in the post-test and the final exam. This suggests that UNIXvisual is very effective in helping students understand the access to objects with directory traversal, which forms the most difficult questions in the tests.

3.4.3.3 Evaluation Form

We used a set of questions to collect information on students' perception of the effectiveness of the tool. We also gathered information on the time spent on using the tool and the students' major. The first 12 rating questions study the effectiveness of UNIXvisual. Q1 and Q2 examine the overall effectiveness; Q3 and Q4 relate to the two views that show object permissions without and with directory traversal; Q5 and Q6 are about the views that interpret permissions from the perspective of a user or a group; Q7 and Q8 examine the Permission Calculator; Q9 is about the Query; Q10, Q11 and Q12 are about the interface design. The choices are: 1:strongly disagree, 2:disagree, 3:neutral, 4:agree, and 5:strongly agree. Q13 and Q14 study the time participants spent on the tool. The choices for Q13 are 1:once, 2:twice, 3:3-4 times, 4:5-10 times, and 5:more than 10 times. The choices for Q14 are 1:less than 5 mins, 2:5-14 mins, 3:15-29 mins, 4:30-60 mins, and 5:more than 1 hour. The questions are as follows:

Table 3.10
UNIXvisual Rating and Usage Questions (© 2017 ACM. Reprinted by permission.)

Rating Questions	
Q1	UNIXvisual helped to better understand UNIX permissions
Q2	UNIXvisual enhanced UNIX permissions course coverage
Q3	Decision View helped to understand which users have access to a certain object and why
Q4	Object View helped to understand which users have access to a certain object and why
Q5	User View helped to understand how decisions are made to the access request from a particular user
Q6	Group View helped to understand how decisions are made to the access request from a particular group
Q7	Permission Calculator was helpful for understanding the meaning of each bit in UNIX permissions
Q8	Permission Calculator was helpful for understanding how to specify permissions for an object
Q9	Query was helpful for understanding UNIX permissions
Q10	The use of colors in the visualization is effective
Q11	The size of items is reasonable and clear
Q12	The layout of items is reasonable and clear
Usage Questions	
Q13	How many times did you use UNIXvisual
Q14	How long did you use UNIXvisual in total

Table 3.11
The Means (μ), and Standard Deviations (σ) of UNIXvisual Evaluation Questions (© 2017 ACM. Reprinted by permission.)

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14
μ	3.81	4.05	4.00	3.81	3.85	4.10	4.43	4.29	3.89	3.81	3.71	3.29	1.95	2.43
σ	0.60	0.50	0.58	0.91	1.09	0.72	0.81	0.85	0.60	0.75	0.72	0.90	0.74	0.81

Figure 3.23: The Means with Confidence Intervals of UNIXvisual Rating and Usage Questions (© 2017 ACM. Reprinted by permission.)

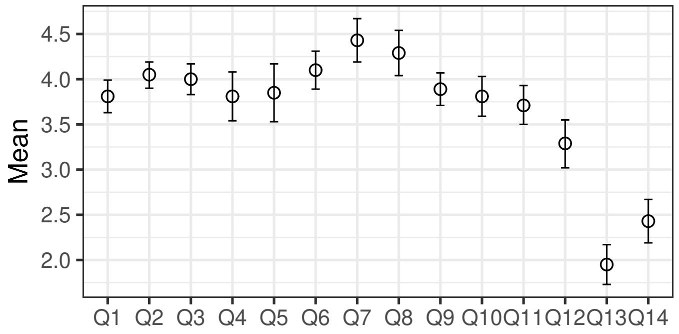


Table 3.11 and Figure 3.23 have the means and standard deviations, and the means with confidence intervals of UNIXvisual rating and usage questions, respectively. All questions except Q12 received a mean greater than 3.7. Students generally believed that UNIXvisual helped the understanding of UNIX permissions and enhanced the course coverage. Permission Calculator received the highest rating (Q7, Q8). The layout received the lowest score (Q12). The reason may be, as mentioned in a student comment, due to the Object View not being scaled properly. Some text overlap was reported. The means of Q13 and Q14 are 1.95 and 2.43, indicating that students generally used UNIXvisual twice for 10 minutes in total. We used the middle point of a range for estimation.

3.4.3.4 Evaluation Form – Student Comments

The four write-in questions were used to collect information of the participants' major, their thoughts on the most and least useful features of the tool, features to add, and problems with installation.

Of the 21 students who used UNIXvisual, 15 students considered the Permission Calculator as the most useful feature. One student wrote that *"the permission calculator can be quite useful to ensure you know how the permissions will look for some file"*. Four students favored the User/Group View. Two other stated the overall features of *"being able to actually check access to a file and check different scenarios"* and *"the instant feedback of whether something works or not with quick explanation"* as the most useful. As for the least useful feature, 16 students did not state any, and three students answered the Permission Calculator. One student mentioned that *"I personally am familiar with permissions, so the calculator was not as helpful"*. Therefore, while 71% of the students considered Permission Calculator as the most useful feature, it is also considered as the least useful one due to the familiarity to the notation conversion on those students' part. Another student considered the Object View the least useful as the view did not scale properly and texts had some overlap.

The students did not encounter any installation problem. When asked to suggest features to add, students were content with the available features. They wrote *"I* *think it is very well designed. Nothing needs to be added"*, and *"the software was very friendly at aiding further learning and understanding of UNIX permissions the way it currently is"*. There are also some comments for further improvements. Students suggested to add *"something to detect if your files are visible by anyone else"*, and *"having a video tutorial on how to use the software"*.

3.5 Design Framework of the Visualization Systems

In practice, users cannot be allowed to modify mandatory policy on a system in general use. A special environment may be created, for example in a virtual machine, but this can require significant system administration overhead to create and maintain. Additionally, the size and syntax of policy specifications for real systems: (1) may unnecessarily obscure the underlying principles that are being taught and (2) make it difficult to design and grade meaningful assignments. These challenges seriously affect the use of access control.

3.5.1 Design Framework

The above four visualization systems were designed under a unified design framework to mitigate these challenges and thus improve the learning and teaching of four commonly-used access control models both for in-class lectures as well as self-learning outside of the classroom. The design framework focuses on the following aspects:

1. **Provide an environment to practice policies.** Similar to many other computer science subjects, the best way to learn access control is through practical experiments. However, due to security reasons and heavy maintenance, the learners do not have the luxury of practicing their own policies on real or virtual machines. Learning access control would be much easier if there is an environment where learners can write, modify and also load policies to examine the effect of security configurations without concerns of technical details and risks of undermining the security of the underlying systems.

2. **Provide intuitive languages for policy writing.** Mechanisms for setting an access control policy are complex and detailed. For example, a single language is used to specify an SELinux policy [63]. This language intermingles the Type Enforcement, Multilevel Security, and Role-based Access Control models [82]. It is even worse that there is no language for specifying a policy under the UNIX access control model. We believe this unnecessarily complicates the learning of

the basic abstractions of the models. These difficulties in learning and writing policies using existing policy languages present a great distraction from the core characteristics of a model itself. To address this issue, we develop a language for each model that is in the form of a combination of natural language and programming language. Hence, our languages could present the intuitiveness of natural languages to make the core components of a model straightforward to users, and the accurateness of programming languages so that each component is specified and interpreted without ambiguity.

3. **Provide visualization to improve learning.** In the field of access control, the essential concepts behind a model and their relationships within a policy can be difficult to interpret. Visualization, as a means of itemizing the concepts into graphical entities and connections, could extract the meaning of policies and present only the important model characteristics to users. Its dynamic nature also allows the presentation of visual and textual hints to users that could act as an experienced professional to provide guidance at each operation. Leveraging visualization for pedagogical purpose seems to be an appealing strategy.

4. **Help students identify and correct misunderstandings.** Given the difficulty of testing an access control policy, it is common that a policy with error is developed without its correctness being questioned. It is also common that system administrators set access controls without testing their effects. Therefore, there is a need to have pedagogical tools that show the result of a policy, so that

users can identify and correct their misunderstandings, and get into the habit of analyzing developed policies before installment. Our framework implements this feature through policy visualization and query systems.

5. **Reach a large segment of learners.** The number and the duration of lectures available for access control, in an already crowded undergraduate security curriculum, are often limited. Further, it is still common for important security material to be learned by employees already in the workforce. This indicates a need for software that supports deep, interactive self-study. Our framework supports this aspect by distributing hand-selected policies to illustrate important model characteristics. We use a tool for each model, to support learners that want to focus on one particular model. The other framework characteristics, like policy visualization and query systems, also support self-study. Moreover, students have varying backgrounds and can comprehend the same materials to different degrees. They also have their own learning styles and preferences. Some may learn best through graphical representations, while others may learn better working on examples and exercises. To address these issues, our framework attempts to support these different learning modes by providing a graphical representation of access control models showing the impact of each rule dynamically, supporting the import of custom policies, and adding a quiz system for conducting self-evaluation and instructor quizzes.

3.5.2 Implementation

Our software design framework for teaching and learning access control is composed of four parts: a model-based specification language, a visualization system, a query system, and a quiz system. The visualization introduces the basic concepts and demonstrates concrete examples. The specification languages are distinct for each access control model but share a similar syntax for the ease of use. Simple policies can be written and visualized in class. Complex policies can be built graphically step by step and effect of each modification is immediately brought to notice. The query system includes a list of commonly asked question. When a question is asked, the solution is provided and animation is utilized for step-by-step explanations. It serves a dual purpose of policy testing and self-test on the understanding of the subject. The quiz system takes two forms. There are Student self-test and Instructor written test for students' and instructors' use. We designed the framework in a way that the software built under it can increase the users' understanding of the target subject, is convenient to use at any time by people with different learning styles, and easy to use so that users only have to focus on teaching/learning the target subject.

Chapter 4

Access Control Policy Authoring and Analysis System

4.1 Overview

The previous chapter introduced visualization systems that were developed for creating and editing access control policies. The systems were proven effective in facilitating the teaching and learning of the access control models. This attains our first goal of providing an easy approach for learning access control models.

For the second goal of efficient access control policy management, we developed a user-level visualization system for policy authoring and analysis. The system of policy authoring and analysis is to achieve the following properties:

1. **Easy access control policy authoring.** The system is designed to provide new learners with an environment to practice access control policy design and evaluation without affecting the underlying operating system, and to provide security professionals with a tool to manage policies effectively and coherently for different security goals. A simplified model language, akin to common English expression of access requirements within an organization, such as "Tom can read and write to directory /Software_development/Project_zero", was designed for this purpose. At the implementation level, the system supports the RBAC model. This model is incorporated because it has the flexibility of being mandatory as well as discretionary access control, and is a model frequently taught and used in many institutions and corporations.

2. **Textual and graphical representations of policies and analysis feedback.** In addition to the traditional textual representation, it is important for a tool to be capable of presenting the policy properties through graphs as human eyes naturally capture and organize information better through graphs than through texts [5]. The graphical representation also has the advantage of displaying an abundance of information and accommodating human cognitive capability with the application of appropriate visualization strategies [67]. For example, the access of a user to an object and all objects on the path starting

from the root directory to the object of interest could be displayed simultaneously. Users will be able to see the stopping point of access when there is no access to the object of interest. Or users can see the penetration of access going from the root to that object when access is possible. Therefore, a graphical representation is able to show an abundance of information in a succinct way, and reveals information that otherwise would be hidden. We use graphical representation to support large-scale policies which are challenging to read and interpret collectively in practice, show the process of an event through step-to-step guidance, and help notice problems.

3. **Interactive access queries.** Checking whether a request is allowed or disallowed after a policy is composed has the same importance as testing after a program is written. It should be an everyday practice, yet has been overlooked. Our system values testing and supports it through access queries. The queries include types such as *which objects are accessible by a certain user*, *whether a user has access to a certain object*, *which users can access a certain object*, and so forth. The answers to each query are represented in both texts and graphs.

Figure 4.1 depicts the main structure of the system with the ellipses representing data models and the rectangles representing process modules.

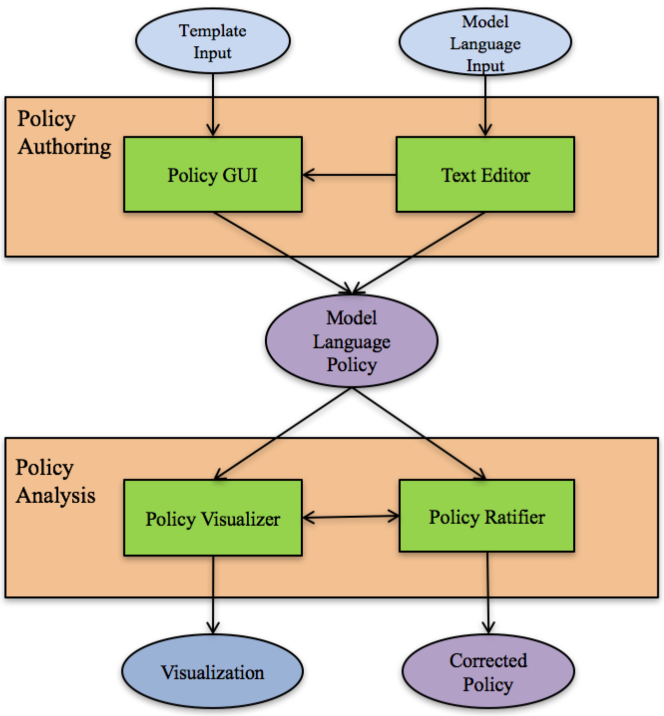


Figure 4.1: System Structure

The system consists of four major parts. The first part is a model language for the specification of policy rules. To minimize the learning overhead, the language simplifies each rule to the assignment of the user, object, action components, which aligns with the format of common access control requirements within organizations.

The second part is the policy authoring component that supports both template-based GUI entry as well as text editor entry. The GUI allows direct entry of policy components for setting up rules with detection of redundant rules in real time. As traditional policy authoring requires knowledge of the model language, it can be difficult for beginners. This component was designed for easy configuration of permissions through visualization and common computer operations without relying on users' access control expertise; the GUI template allows users to set access rules through the least amount of policy elements (user, object and actions), so that new users can start policy writing practice easily. Still, our system allows users who are familiar with the model language to write policies in regular text editors.

The third part is the policy ratifier component. The policy ratifier takes a policy, translates the policy into a different format recognizable to a third-party model checker, and allows access queries in textual forms.

The last part is the policy analyzer component. The policy analyzer takes a policy file and interprets the rules into graphical entities. Users may interact with the visualization to explore information of user's access to objects, how a user gets the access to certain objects, the roles associated with a certain user, and permissions assigned to different roles. The following sections introduce the model language for policy specification, the policy authoring component, the policy ratifier component, and the policy analyzer component in order.

4.2 Model Language

A policy is a collection of specifications that define the values and relationship of key elements, which include user, object and actions/permissions in this case. The users are the user accounts existing in a file system. The objects are the files or directories within the same system, which are the source of information needed for users to accomplish different tasks. The permissions are a combination of read, write, and execute. They represent the kind of actions a user is allowed to conduct on certain objects. We use ⟨User⟩, ⟨PATH⟩, and ⟨PERM⟩ for the representation of a user, an object and a set of permissions, respectively. In a policy, we allow two sources of file system information; one directly defined through user, object and permissions, and one from a preprocessed file containing directory hierarchy extracted from a real system. Thus, our tool allows both individual and merged input of a policy-defined file system and a real file system.

In our model language, each line of specification starts with a keyword. If the keyword indicates the definition of an element, it is followed by one or more values in the type of that element. Otherwise, the rule defines the relationship between elements and the relationship expression follows the keyword. The following defines the use of keywords:

- `oscrawlfile` defines the path pointing to the preprocessed file that contains the directory tree within an operating system. This piece of information can be obtained by our python script "crawlDirectory.py" included in the "policies" folder. The file has the hierarchy of all objects under a root directory specified when running the script. The crawled directory tree will be merged with the policy-defined file system based on the containment of directories; the tool will only use the policy-defined file system if no path is given for this keyword.

- `root` specifies the starting directory for the object hierarchical tree. For both crawled directory tree and the policy defined file system, only the root directory and its descendants will be read into the tool.

- `user` contains the list of all users.

- `object` contains the list of objects. The language allows the specification of disconnected directories, and the objects between them are generated with the hierarchy being automatically computed. For example, if / and /Software_development/Project_zero/source_code are the only directories specified in a policy, directories of /Software_development and /Software_development/Project_zero will be automatically added and their default permissions are set to none. Users can modified the permissions either by explicitly specifying them in the policy, or through the visualization component.

- `rule` specifies the content of a rule. A rule is composed of a user, a set of permissions, an optional recursive sign for permissions and a list of objects.

The grammar for specifying policies uses a format of User-Object-Action (UOA), and is defined as follows:

# UOA Specification

$$\texttt{UOA} \rightarrow \texttt{Stmt} \mid (\texttt{Stmt}\ \texttt{UOA})$$

$$\texttt{Stmt} \rightarrow \texttt{HeadStmt} \mid \texttt{UserStmt} \mid \texttt{ObjStmt} \mid \texttt{RuleStmt}$$

# Statements

$$\texttt{HeadStmt} \rightarrow (\texttt{OSCrawlStmt} \mid \varepsilon)\ \texttt{RootStmt}$$

$$\texttt{OSCrawlStmt} \rightarrow \text{``oscrawlfile:''}\ \texttt{<PATH>}$$

$$\texttt{RootStmt} \rightarrow \text{``root:''}\ \texttt{<PATH>}$$

$$\texttt{UserStmt} \rightarrow \text{``user:''}\ \texttt{UserList}$$

$$\texttt{ObjStmt} \rightarrow \text{``object:''}\ \texttt{ObjList}$$

$$\texttt{RuleStmt} \rightarrow \text{``rule:''}\ \texttt{<USER>}\ \texttt{PermList}\ \texttt{Recursive}\ \texttt{ObjList}$$

# Tokens and lists

$$\texttt{Recursive} \rightarrow \text{``-r''} \mid \varepsilon$$

$$\texttt{UserList} \rightarrow \texttt{<USER>} \mid (\texttt{<USER>}\ \text{``,''}\ \texttt{UserList})$$

$$\texttt{ObjList} \rightarrow \texttt{<PATH>} \mid (\texttt{<PATH>}\ \text{``,''}\ \texttt{ObjList})$$

$$\texttt{PermList} \rightarrow \texttt{<PERM>} \mid (\texttt{<PERM>}\ \text{``,''}\ \texttt{PermList})$$

Our prototype for the policy language also supports a comment sign “#”. Figure 4.2 has an example policy.

```
oscrawlfile: /User/Documents/classesCrawl.txt
root:   /
user:   alan, tina, sally, sam, oscar
object: /, /tools, /classes/os, /classes/security,
        /classes/security/public, /classes/os/public,
        /home, /classes/security/private, /classes/os/private

rule: alan  r,x     -r  /tools
rule: alan  r,w,x   -r  /classes/os, /classes/security

rule: tina  r,x     -r  /tools
rule: tina  r,w,x       /classes/os, /classes/security
rule: tina  r       -r  /classes/security/public, /classes/os/public

rule: sally r,x     -r  /tools
rule: sally r,w     -r  /classes/security/public, /classes/os/public

rule: oscar r,x     -r  /tools
rule: oscar r       -r  /classes/os/public

rule: sam   r,x     -r  /tools
rule: sam   r       -r  /classes/security/public
```

Figure 4.2: Policy in UOA Model Language

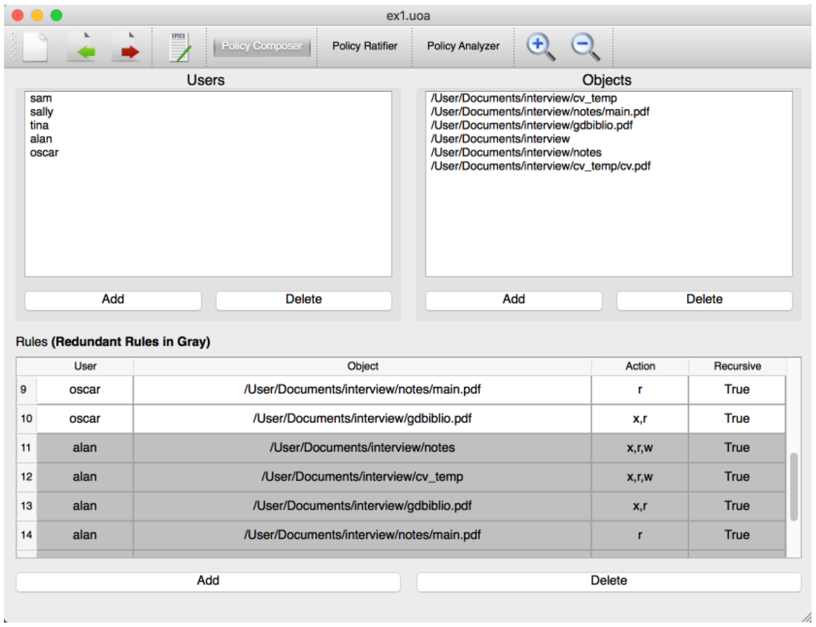
The policy constructs a file system from two parts. One part is a directory tree specified in the file /User/Documents/classesCrawl.txt. The other part is another directory tree generated from the object list following the key word object. The two parts are truncated at the specified root directory / with all its descendants further merged based on directory containment. Following the file system definition, the access to objects are defined through rules. For example, “rule: tina r,w /classes/os, /classes/security” allows user tina the read and write permissions to the directories /classes/os and /classes/security. If tina can further have the same access to all objects contained in these two directories, then the recursive sign “-r” could be placed before the directories.

4.3 Policy Authoring

Composing a policy usually requires knowledge of the policy language in addition to the underlying access control models. The traditional way of writing policy files in text editors can be convenient for professionals and experts. But for beginners, the process of learning the models and their policy languages can be time-consuming. We designed the authoring component to simplify the process of writing policies for users, particularly new learners.

The authoring component supports two types of inputs. The first type is text entry in regular text editors using our model language. The policy file is saved with the extension of “uoa”, which allows it to be imported into the system. This could be convenient for users who are already familiar with the language.

The second type is a template-based policy entry. This type allows direct permission setting through the graphical interface. It allows entry of values for the user and object components and for access rules to be set through a combination of user, object, action and the recursive sign. The user is a user account. The object is a file or directory. The action is a set of r, w, and x, which stand for read, write, and execute accesses. They carry the same permissions as in the UNIX permissions. Lastly, the recursive sign is a boolean value, where false means the action to the object only applies to the object itself while true means the action also applies to all objects beneath the specified object. Figure 4.3 shows the template interface. In addition to writing a policy, this component also detects redundant rules and highlights them in grey. In this example, alan has recursive r, w, x access to /Users/Documents/interview, thus any rule granting access to objects contained in the directory is a redundant rule (in grey) and can be removed by users manually. Through these methods, users are able to use the values provided to write policies without restrictions from their access control expertise.



Users

sam
sally
tina
alan
oscar

Objects

/User/Documents/interview/cv_temp
/User/Documents/interview/notes/main.pdf
/User/Documents/interview/gdbiblio.pdf
/User/Documents/interview
/User/Documents/interview/notes
/User/Documents/interview/cv_temp/cv.pdf

Rules **(Redundant Rules in Gray)**

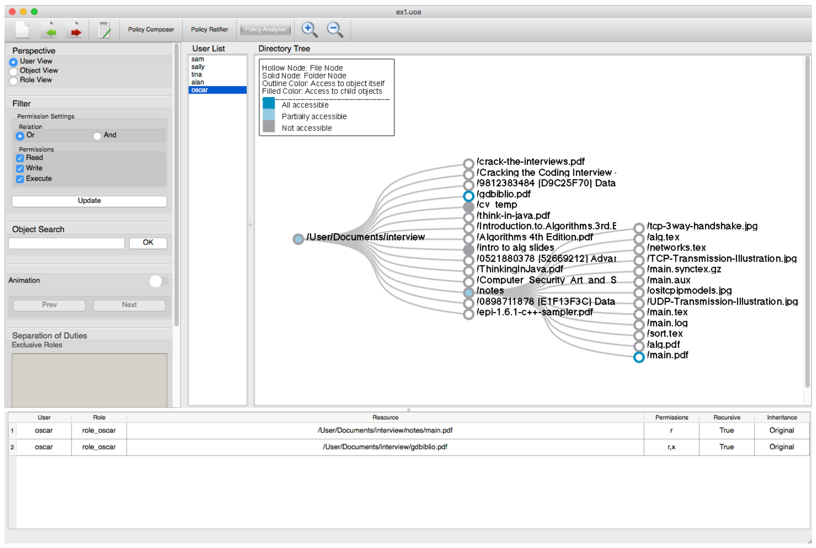
	User	Object	Action	Recursive
9	oscar	/User/Documents/interview/notes/main.pdf	r	True
10	oscar	/User/Documents/interview/gdbiblio.pdf	x,r	True
11	alan	/User/Documents/interview/notes	x,r,w	True
12	alan	/User/Documents/interview/cv_temp	x,r,w	True
13	alan	/User/Documents/interview/gdbiblio.pdf	x,r	True
14	alan	/User/Documents/interview/notes/main.pdf	r	True

Figure 4.3: Policy Composer in ACvisual

4.4 Policy Analysis

The Policy Analyzer component takes a policy composed through the GUI or from an input file written in the model language and generates a visual representation of the rules in the policy. The UOA policy is then converted to an implementation of the RBAC model. An RBAC policy has a list of users, a list of objects and some roles that carry permissions. Each user can be assigned to one or more roles from which she acquires the permissions to objects. As the UOA carries the same meaning of users and objects with the RBAC model, the core conversion then becomes the introduction of roles. Given that roles are different in permissions to objects (role inheritance forms a Hasse Diagram), we create roles based on the distinctive set of permissions from all users. Then the users are assigned to roles with the permissions that are originally assigned to them in the UOA policy; users who have the same set of permissions are assigned to the same role. Role hierarchy is built based on permission set containment; if a role's permission is a proper subset of a second role, then the second role (senior role) inherits the first role (junior role). This approach takes the most straightforward interpretation of user permissions and casts it to a role, which can be coarse in terms of permission division. In the case of a simplified software development team, there are a group of software engineers and a manager. The software engineers can belong to one of the software development, software engineers in tooling, and quality assurance teams. Each team allows their engineers to have read, write and execute permissions to their own data resources. The manager only has the read and execute access. In our system, the manager's role will be a role that does not inherit any role as all developer's roles have write permission that the manager does not have. Thus, the manager's role has permissions to three engineering teams all together. But it is usually a good practice to separate permissions to different team's resources. So there is a possibility that users would prefer creating an individual role with read and execute permissions for each team and let the manager inherit all of them. To encouraging careful role divisions, we allow in the tool ways to make changes to the user, object, role and permissions for custom needs.

The visualization then illustrates user-to-role assignment and role-to-object permissions through graphical entities. Users may interact with the visualization to explore information of user's access to objects, how a user gets the access to certain objects, the roles associated with a certain user, and permissions assigned to different roles. There are three views in this component: the User View, the Object View, and the Role View, within which the User View and the Object View have a simple version and a detail version. The simple version shows the permission query result directly. The detail version shows the process of examination in steps. The Animation Switch toggles the visualization between these versions. All the graphs support zooming operations.



Hollow Node: File Node
Solid Node: Folder Node
Outline Color: Access to object itself
Filled Color: Access to child objects

🟦	All accessible
🟦	Partially accessible
⬜	Not accessible

	User	Role	Resource	Permissions	Recursive	Inheritance
1	oscar	role_oscar	/User/Documents/interview/notes/main.pdf	r	True	Original
2	oscar	role_oscar	/User/Documents/interview/gdbiblio.pdf	r,x	True	Original

Figure 4.4: Policy Analyzer in ACvisual

Figure 4.4 shows the interface of the Policy Analyzer. The tool panel on the left has four sections. The first section Perspective allows the activation of one of the three views. The second section Permission Filter Setting specifies the permission the query checks for. The third section Object Search asks for an object path, and the object will be placed as the root of the Object Tree. This filters the unwanted objects out to keep the Object Tree less cluttered. The last section configures the exclusive roles for the purpose of Separation of Duties. On the right side, there are two parts: the User List and the Visualization Canvas. The User List shows the list of all available users. The Visualization Canvas displays Role View, User View, or Object View, which are introduced below in order.

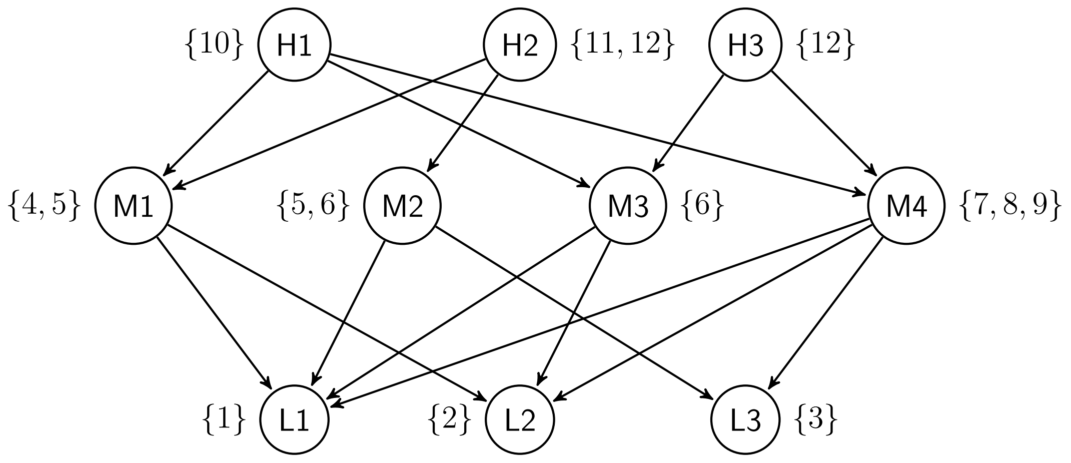
4.4.1 Role View

Our system supports the Core and Hierarchical RBAC models (Section 1.1.4). Roles carry the user permissions through the user-to-role assignment together with the role-to-object permissions. The relationship between roles is also hierarchical where a senior role can acquire a junior role's permission by inheriting the latter.

Table 4.1
Role Configuration of A File System

Role Name	Direct Privileges	Effective Privileges
H_1	{10}	{1,2,3,4,5,6,7,8,9,10}
H_2	{11,12}	{1,2,3,4,5,6,11,12}
H_3	{12}	{1,2,3,6,7,8,9,12}
M_1	{4,5}	{1,2,4,5}
M_2	{5,6}	{1,3,5,6}
M_3	{6}	{1,2,6}
M_4	{7,8,9}	{1,2,3,7,8,9}
L_1	{1}	{1}
L_2	{2}	{2}
L_3	{3}	{3}

Suppose that a system has role configuration as in Table 4.1. The privileges are represented as numbers. Every role has distinctive direct privileges and effective privileges that define their permissions. The direct privileges are the permissions directly assigned to that role, not permissions obtained through inheritance. The effective privileges are the union of the direct privileges of the role itself and the effective privileges of its junior roles. If the roles are depicted as nodes with labels of direct privileges, and the inheritance relationship is drawn from the seniors to the juniors as directed edges, we will be able to draw the configuration in the table as a role graph:



Assume that we have a set of roles $R = \cup_i r_i$ where r_i is a role with effective privileges $P(r_i)$. Among these roles, the set of role inheritances are denoted as $E = \cup_{i,j} e_{ij}$, where e_{ij} denotes that role r_i inherits role r_j. Given that a role r_1 is inherited by another role r_2 if $P(r_1)$ is a subset of $P(r_2)$, we can find the following properties of inheritance relationship from the role graph:

Property 1: Reflexivity

For any role r_i, it inherits privileges from itself.

Proof: For role r_i, its effective privileges $P(r_i)$ is a subset of itself ($P(r_i) \subseteq P(r_i)$). Based on the definition of role inheritance, r_i is inherited by itself.

Property 2: Antisymmetry

For inheritance e_{ij}, it is different from e_{ji} unless $r_i \equiv r_j$.

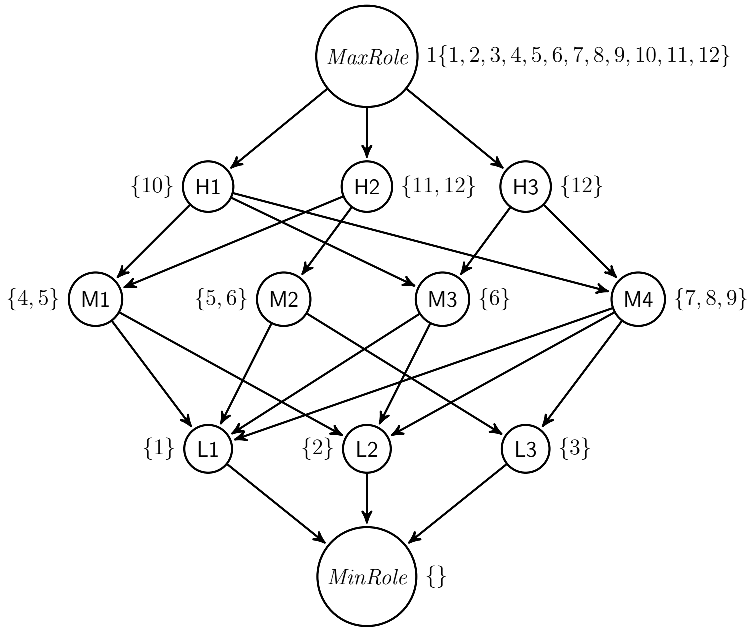
Proof: Suppose we have two roles r_i and r_j that have inheritance relationship e_{ij}. If e_{ji} is the same as e_{ij}, then r_i and r_j satisfy e_{ij} and e_{ji} at the same time. That is, r_j is inherited by r_i (e_{ij}) and r_i is inherited by r_j (e_{ji}). This gives us $P(r_i) \subseteq P(r_j)$ and $P(r_j) \subseteq P(r_i)$, so that we have $P(r_i) = P(r_j)$. As each role has a unique set of privileges, $P(r_i) = P(r_j)$ would not be possible if r_i and r_j are different roles. Therefore, either $r_i \equiv r_j$ to satisfy both inheritance e_{ij} and e_{ji}, or e_{ij} and e_{ji} carry different inheritance relationship for different roles.

Property 3: Transitivity

If r_k inherits r_j, which further inherits r_i, then r_k also inherits r_i.

Proof: With role r_k inheriting r_j and r_j inheriting r_i, we have $P(r_j) \subseteq P(r_k)$ and $P(r_i) \subseteq P(r_j)$. Given the transitivity property of subsets, we further have $P(r_i) \subseteq P(r_k)$. This means that r_k also inherits r_i.

In this graph, edges that can be obtained through reflexivity (introducing self-loops) and transitivity are removed. For example, if role `H1` inherits role `L1`, the edge from `H1` to `L1` will not be drawn as role `H1` already inherits role `M1`, which also inherits `L1` and has effective privileges that contain privilege 1 from `L1`. This generates the Hasse diagram that effectively reduces the clutter in the graph while the same amount of information is preserved. As it is possible to have two roles without an inheritance relationship (*e.g.*, roles at the same level), the inheritance relationship represents a partially ordered set. If we construct a *MaxRole* that has the permission of the union of permissions from all available roles, and a *minRole* that has no permissions, then any pair of roles in the system will have a unique supremum and a unique infimum. This makes the role hierarchy a lattice. Since a role with no permission is unnecessary in real systems, the *minRole* will not be included in our system.

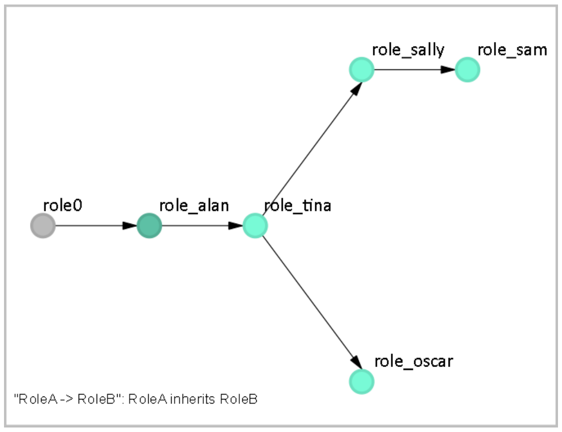


There are five aspects of fundamental importance for the role hierarchy visualization: 1) inheritance relationship between roles; 2) user-to-role assignment; 3) user privileges from direct role assignment and role inheritance; 4) whether there are cycles in the role inheritance graph; 5) whether any user's role assignment violates the Separation of Duties (see Section 4.4.1.2). In Role View, the User List shows all users and the role hierarchy is displayed through a graph. Each role is depicted as a node, and the inheritance between roles is depicted as a directed edge pointing from the senior role to the junior role. This view allows the exploration of roles assigned to individual users and the inheritance between roles. The roles assigned to a user can be displayed by selecting that user. The directly assigned roles and inherited roles will be respectively in orange and light orange while non-relevant roles being grayed out. The permissions of each role can be examined by selecting a role node from the role hierarchy. The associated rules will be displayed in a table at the bottom; the roles inherited by the selected role will also be in light green. role_0 is constructed as the *maxRole* which has recursive read, write and execute permissions to all objects in the system. If there is an actual role with the permissions of the *maxRole*, then that role will be in the place of role_0.

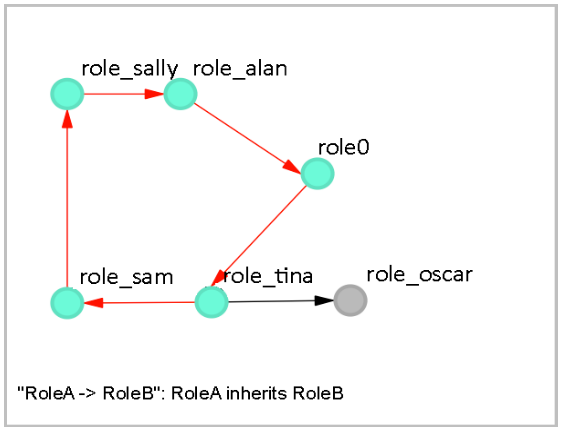
Table 4.2
Example Role Permissions

Role	Permissions	Objects	Recursive
role_sam	r,x	/tools	True
	r	/classes/security/public	True
role_oscar	r,x	/tools	True
	r	/classes/os/public	True
role_sally	r,x	/tools	True
	r,w	/classes/security/public	True
role_tina	r,x	/tools	True
	r,w,x	/classes/os, /classes/security	False
	r	/classes/security/public, /classes/os/public	True
role_alan	r,x	/tools	True
	r,w,x	/classes/os, /classes/security	True

In Table 4.2, role_alan has the same permission as role_tina for /tools, and has all possible permissions to /classes/os and /classes/security and objects included in them. The permissions role_tina has to /classes/os and /classes/security and objects beneath them is a subset of those of role_tina. Therefore, role_alan inherits role_tina. The inheritance relationship between other roles are based on this same permission containment. Figure 4.5 (a) shows the role inheritance hierarchy of this example. In addition to building the correct role inheritance, problems such as cyclic role inheritance and violation of Separation of Duties can occur and produce problematic policies in practice.



(a)



(b)

Figure 4.5: Role View. (a) shows role inheritance hierarchy. (b) shows cyclic role inheritance.

4.4.1.1 Cyclic Role Inheritance

Cyclic role inheritance happens when a junior role inherits its ancestor role or has no less permission than one of its ancestors. It is a situation people should be made aware of during the setup of an access control policy, as it allows junior roles to have the same privileges as its senior roles. This can be an indication of the permission of the junior is mistakenly set up or multiple roles sharing the same permissions which introduces redundant roles. In the above example (Figure 4.5 (a)), suppose recursive read, write and execute to "/" is accidentally added to role_sam, then role_sam has the same permissions as role_0. Since role_0 already has this permission before role_sam does, then role_sam will be inheriting role_0, which introduces a cycle between role_sam and role_0. All the roles in between are also affected and share the same permissions. Our role hierarchy computes this relation and highlights the cycle in red (Figure 4.5 (b)) as a notification of the existence of a cyclic inheritance. With this piece of information, users will be aware of the situation and can make adjustments accordingly.

4.4.1.2 Separation of Duties

Separation of Duties is the concept of having more than one person involved to complete a task. It is to avoid granting excessive privileges to one person. In the case of distributing medicine to a patient, a prescription is needed from a doctor and taken to a pharmacist. Here, a doctor can provide a prescription, but does not have the right to dispense the medication to the patient by laws and regulations. Therefore, one is only allowed to either be the doctor or the pharmacist. Otherwise, it is possible to let the patient unknowingly abuse drugs.

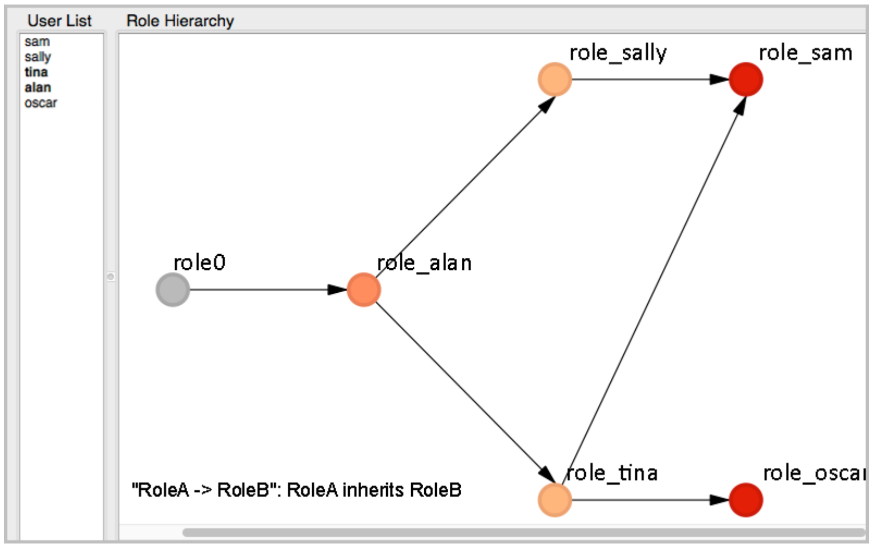


Figure 4.6: Role Inheritance Graph in Role View

We allow users to specify exclusive roles within SOD relationship in groups; each line in the interface indicates a group, and roles of a group, which are delimited by commas, should not be assigned to the same user. The role hierarchy shows the roles within different groups in different colors. In Figure 4.6, role_sam and role_oscar are in an SOD relationship, indicated in the graph as brick red color. Users tina and alan that violate this SOD setup are shown in bold font under the User List.

The detection of users who violate SOD relationships is achieved through the trace of role inheritance. Since users obtain permissions through roles, finding the roles that violate the SODs is the key step. For the roles within an SOD relationship, their nearest common ancestor is the lowest level of role that have permissions from the exclusive roles of the SOD specification and should be first located. Then all the roles that inherit this ancestor role (including the ancestor itself) must have violated the SOD relationship. Based on the user-to-role assignment matrix, the users who violate the SODs are the users assigned to the set of ancestor roles. From the graph, the exclusive roles in a SOD specification are role_sam and role_oscar. The nearest ancestor is role_tina. Both role_tina and role_alan inherit role_tina. Therefore, users tina and alan have permissions from role_sam and role_oscar at the same time, and thus violate the SOD relationship between these two roles.

4.4.2 User View

The User View examines the privileges of a user and is composed of the Role Hierarchy and the Object Tree. A simple version and a detail version are supported for different levels of details in answering access queries. The simple version shows the answer of a query directly; the detail version shows the process of access determination in steps. The Object Tree, the Role Hierarchy, and the access query are described below.

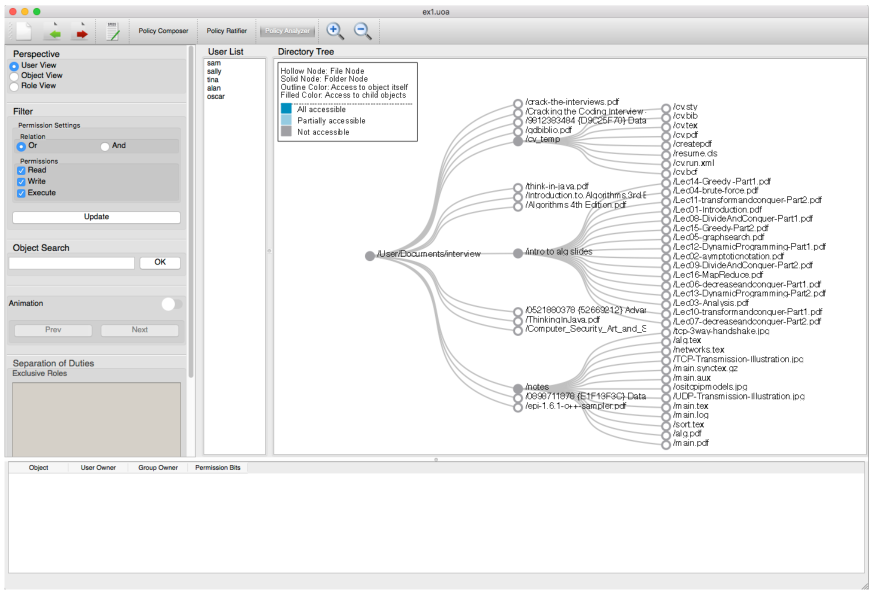


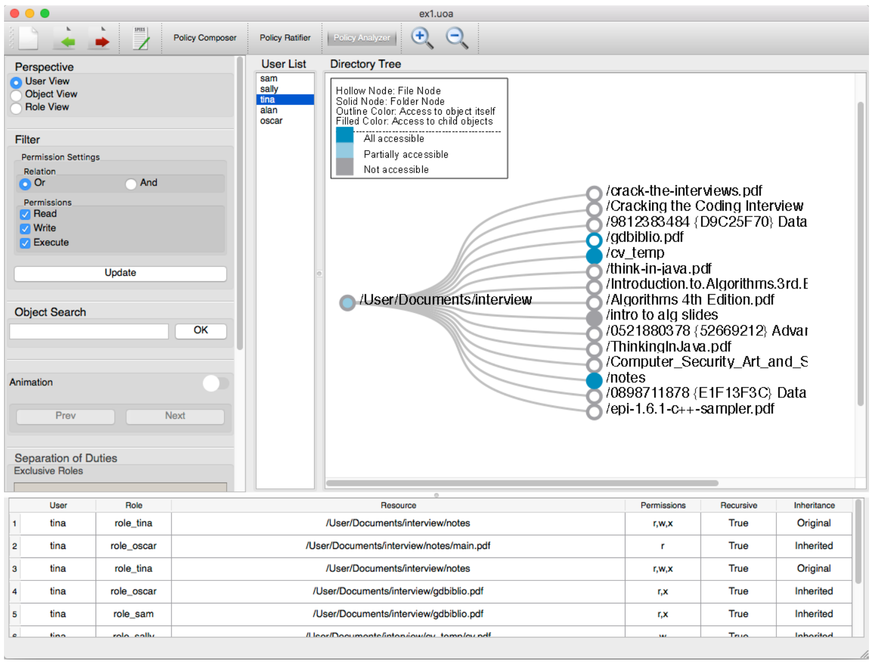
Figure 4.7: User View

4.4.2.1 Object Tree

The traditional directory tree allows the exploration of objects starting from a root directory through mouse clicks. Nodes can be expanded level by level and collapsed all at once. The advantage of this method is that it has been widely used and is very familiar to average users. However, it has drawbacks in three aspects: 1) The traversal must always start from the root node; 2) It is hard to locate a node if too many nodes are expanded; 3) It is hard to trace a node's path from the root. To counter these problems, we decided to use a SpaceTree [50] prototype with additional functionalities for better navigation and reduced clutter.

In the Object Tree, the directory tree is depicted as a horizontally oriented tree with nodes representing objects and edges representing directory containment. Files are depicted as hollow nodes and directories are depicted as solid nodes. The tree is initiated to be rooted at the specified root in the policy. Left mouse clicking a node would highlight the path from the root node to that node. Left mouse clicking a directory node plus Shift button would enable toggling between expanding and collapsing that node. Instead of showing all objects at once, the level of our Object Tree is limited to a maximum of three levels. Suppose that the root is at level one, if a node at level three is expanded, the whole tree would move down one level and be rooted at the ancestor of the expanded node at level two. The siblings of the new root and the nodes originally in level one are then hidden. When collapsing a directory node, the whole tree would make the opposite movement to that of the expansion. That is, the parent of root becomes the new root, and the siblings of the old root are shown. We design the Object Tree exploration in this manner as the closely related information, such as the nodes on the path from the root to the operated node, and the sibling of the operated node, is what carries the information that would affect further explorations when the expanding and collapsing of a node.

When the user of interest is chosen from the User List, the object nodes are colored by its accessibility of a user. The outline color of a node indicates the user's accessibility to the object itself; the filling color of a node indicates the user's accessibility to the children of the clicked object. The intensity of blue color further indicates the level of accessibility: blue means accessible to the node itself or to all child nodes; light blue means that only part of the child nodes is accessible; and gray means not accessible to the node itself or to any child node. Figure 4.8 shows an example object tree when user tina is selected. From the Object Tree, we know that only part of the objects beneath /User/Documents/interview is accessible. We also know that all objects in /User/Documents/interview/notes are accessible, and all objects in /User/Documents/interview/intro to alg slides are not accessible. Even though showing objects within these directories is allowed by the maximum level constraint, the directories are initiated as collapsed. We collapse the nodes with all accessible and all inaccessible children as the accessibility of children is already visible through the filling color of their parent nodes. Through hiding the children, the clutter of the graph is reduced and the scalability of visualization improves. If a user is interested in the detailed permissions of those child objects, manual expansion can still help.


ex1.uoa
Policy Composer
Policy Ratifier
Policy Analyzer
User List
sam
sally
tina
alan
oscar
Directory Tree
Hollow Node: File Node
Solid Node: Folder Node
Outline Color: Access to object itself
Filled Color: Access to child objects
All accessible
Partially accessible
Not accessible
/User/Documents/interview
/crack-the-interviews.pdf
/Cracking the Coding Interview
/9812383484 {D9C25F70} Data
/gdbiblio.pdf
/cv_temp
/think-in-java.pdf
/Introduction.to.Algorithms.3rd.E
/Algorithms 4th Edition.pdf
/intro to alg slides
/0521880378 {52669212} Advar
/ThinkingInJava.pdf
/Computer_Security_Art_and_S
/notes
/0898711878 {E1F13F3C} Data
/epi-1.6.1-c++-sampler.pdf
Object Search
OK
Animation
Prev
Next
Separation of Duties
Exclusive Roles
Update

Perspective: [✓ User View] Object View Role View

Filter — Permission Settings — Relation: [✓ Or] And

Permissions: [✓ Read] [✓ Write] [✓ Execute]

	User	Role	Resource	Permissions	Recursive	Inheritance
1	tina	role_tina	/User/Documents/interview/notes	r,w,x	True	Original
2	tina	role_oscar	/User/Documents/interview/notes/main.pdf	r	True	Inherited
3	tina	role_tina	/User/Documents/interview/notes	r,w,x	True	Original
4	tina	role_oscar	/User/Documents/interview/gdbiblio.pdf	r,x	True	Inherited
5	tina	role_sam	/User/Documents/interview/gdbiblio.pdf	r,x	True	Inherited
6	tina	role_sally	/User/Documents/interview/cv_temp/cv.pdf	w	True	Inherited

Figure 4.8: Accessibility of User tina

4.4.2.2 Role Hierarchy

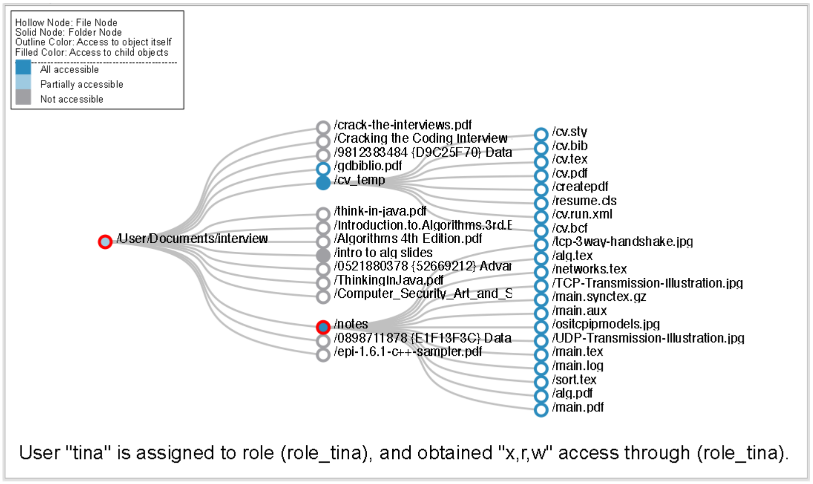
For the examination of a user's access to objects, the permissions of each role explain why some accesses are granted. The Role Hierarchy is constructed to show user-to-role assignment, role-to-object permissions, and roles' inheritance relationship. It is the same graph as in the Role View. Role View allows a detailed exploration from a role's perspective. The Role Hierarchy here act as the media to show detailed permissions of a user through directly assigned roles to the inherited roles based on role hierarchy.

4.4.2.3 Access Query

The User View allows access queries from the perspective of a user. The Animation Switch toggles the visualization between simple version and detail version.

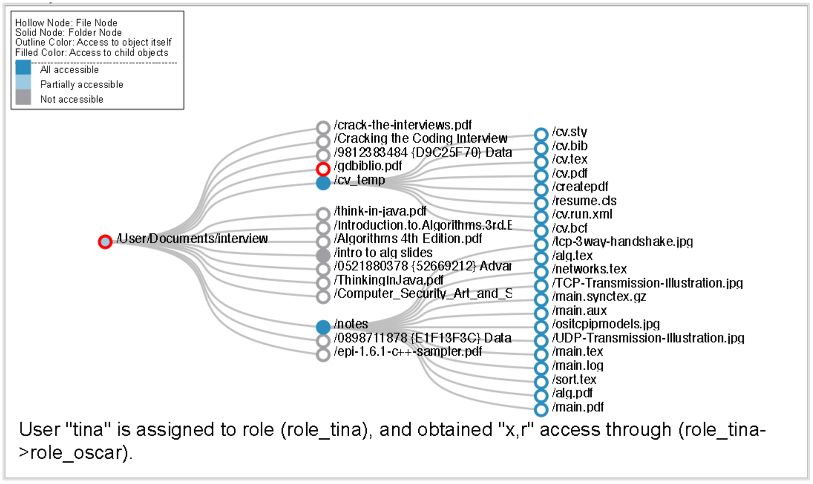
The simple version starts with a selection of a user from the User List. Figure 4.8 shows the accessibility of user tina with the related rules listed in the bottom table. Each rule has the elements of User, Role, Resources, Permissions, Recursive, and Inheritance. It tells which role is used to access an object, the exact permissions the user has to the object, and whether the permission is acquired through direct role assignment or from inherited roles.

In the Object Tree, users may click on any node for an examination of the permission to that object. There are three possibilities to obtain access to an object. A user can access an object through directly assigned roles, through roles inherited by the directly assigned roles, and through roles directly assigned and roles inherited. Figure 4.9 and Figure 4.10 illustrate that tina's permission to notes is read, write and execute and permission to gdbibilo.pdf is read and execute. It also shows some brief role information that the permissions are obtained through directly assigned role role_tina and inherited role role_oscar, respectively. In the second case, the inheritance paths are (role_tina $\rightarrow$ role_oscar) stated for user's reference.


Hollow Node: File Node
Solid Node: Folder Node
Outline Color: Access to object itself
Filled Color: Access to child objects
All accessible
Partially accessible
Not accessible
/User/Documents/interview
/crack-the-interviews.pdf
/Cracking the Coding Interview
/9812383484 {D9C25F70} Data
/gdbiblio.pdf
/cv_temp
/think-in-java.pdf
/Introduction.to.Algorithms.3rd.E
/Algorithms 4th Edition.pdf
/intro to alg slides
/0521880378 {52669212} Advai
/ThinkingInJava.pdf
/Computer_Security_Art_and_S
/notes
/0898711878 {E1F13F3C} Data
/epi-1.6.1-c++-sampler.pdf
/cv.sty
/cv.bib
/cv.tex
/cv.pdf
/createpdf
/resume.cls
/cv.run.xml
/cv.bcf
/tcp-3way-handshake.jpg
/alg.tex
/networks.tex
/TCP-Transmission-Illustration.jpg
/main.synctex.gz
/main.aux
/ositcpipmodels.jpg
/UDP-Transmission-Illustration.jpg
/main.tex
/main.log
/sort.tex
/alg.pdf
/main.pdf

User "tina" is assigned to role (role_tina), and obtained "x,r,w" access through (role_tina).

Figure 4.9: Direct Accessibility of User tina


Hollow Node: File Node
Solid Node: Folder Node
Outline Color: Access to object itself
Filled Color: Access to child objects
All accessible
Partially accessible
Not accessible
/User/Documents/interview
/crack-the-interviews.pdf
/Cracking the Coding Interview
/9812383484 {D9C25F70} Data
/gdbiblio.pdf
/cv_temp
/think-in-java.pdf
/Introduction.to.Algorithms.3rd.E
/Algorithms 4th Edition.pdf
/intro to alg slides
/0521880378 {52669212} Advai
/ThinkingInJava.pdf
/Computer_Security_Art_and_S
/notes
/0898711878 {E1F13F3C} Data
/epi-1.6.1-c++-sampler.pdf
/cv.sty
/cv.bib
/cv.tex
/cv.pdf
/createpdf
/resume.cls
/cv.run.xml
/cv.bcf
/tcp-3way-handshake.jpg
/alg.tex
/networks.tex
/TCP-Transmission-Illustration.jpg
/main.synctex.gz
/main.aux
/ositcpipmodels.jpg
/UDP-Transmission-Illustration.jpg
/main.tex
/main.log
/sort.tex
/alg.pdf
/main.pdf

User "tina" is assigned to role (role_tina), and obtained "x,r" access through (role_tina->role_oscar).

Figure 4.10: Inherited Accessibility of User tina

The detail version is provided for a step-to-step illustration of how a user accesses an object. Users may use the buttons of Prev and Next under Animation to move the animation backward and forward. The animation shows the directly assigned roles of a user, the permissions acquired from directly assigned roles, the roles inherited by the directly assigned roles, the permissions acquired through inherited roles, and at last the union of all permissions to the object as a summary. Figures 4.11, 4.12, and 4.13 illustrate some steps of animation for user `tina`. First, the directly assigned role is highlighted and its permission is explained. Then the inherited roles and edge of inheritance are highlighted to find additional permissions. The final step shows the relevant roles and permissions altogether to the object in question.

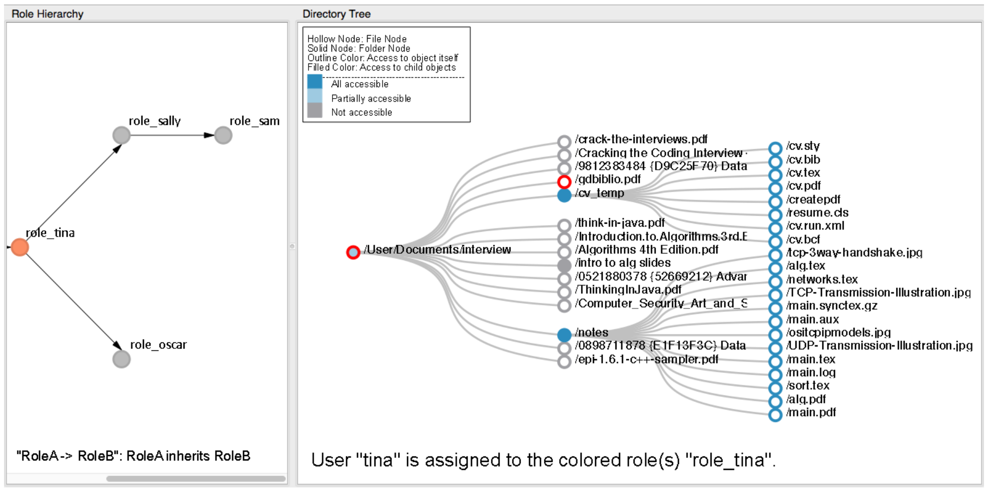

Role Hierarchy
role_sally
role_sam
role_tina
role_oscar
"RoleA-> RoleB": RoleA inherits RoleB
Directory Tree
Hollow Node: File Node
Solid Node: Folder Node
Outline Color: Access to object itself
Filled Color: Access to child objects
All accessible
Partially accessible
Not accessible
/User/Documents/interview
/crack-the-interviews.pdf
/Cracking the Coding Interview -
/9812383484 {D9C25F70} Data
/gdbiblio.pdf
/cv_temp
/think-in-java.pdf
/Introduction.to.Algorithms.3rd.E
/Algorithms 4th Edition.pdf
/intro to alg slides
/0521880378 {52669212} Advar
/ThinkingInJava.pdf
/Computer_Security_Art_and_S
/notes
/0898711878 {E1F13F3C} Data
/epi-1.6.1-c++-sampler.pdf
/cv.sty
/cv.bib
/cv.tex
/cv.pdf
/createpdf
/resume.cls
/cv.run.xml
/cv.bcf
/tcp-3way-handshake.jpg
/alg.tex
/networks.tex
/TCP-Transmission-Illustration.jpg
/main.synctex.gz
/main.aux
/ositcpipmodels.jpg
/UDP-Transmission-Illustration.jpg
/main.tex
/main.log
/sort.tex
/alg.pdf
/main.pdf
User "tina" is assigned to the colored role(s) "role_tina".

Figure 4.11: Directly Assigned Roles of User `tina`

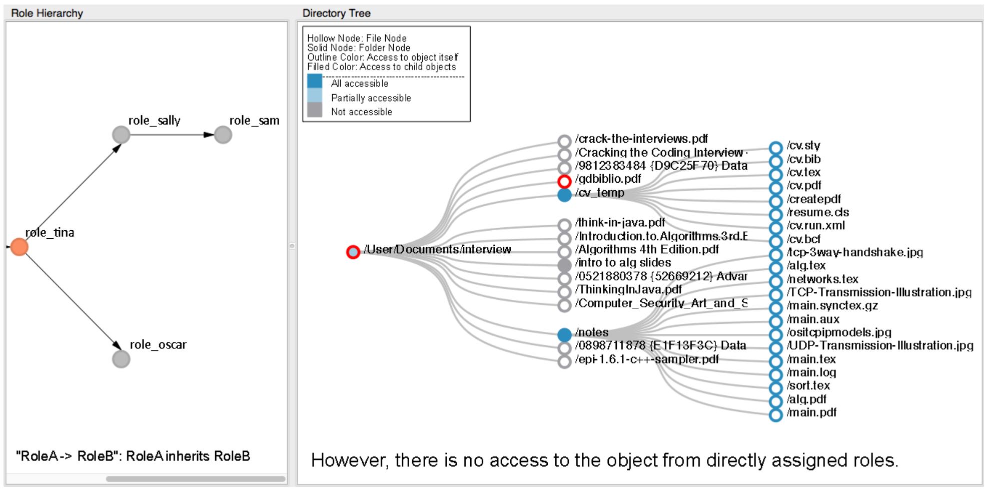
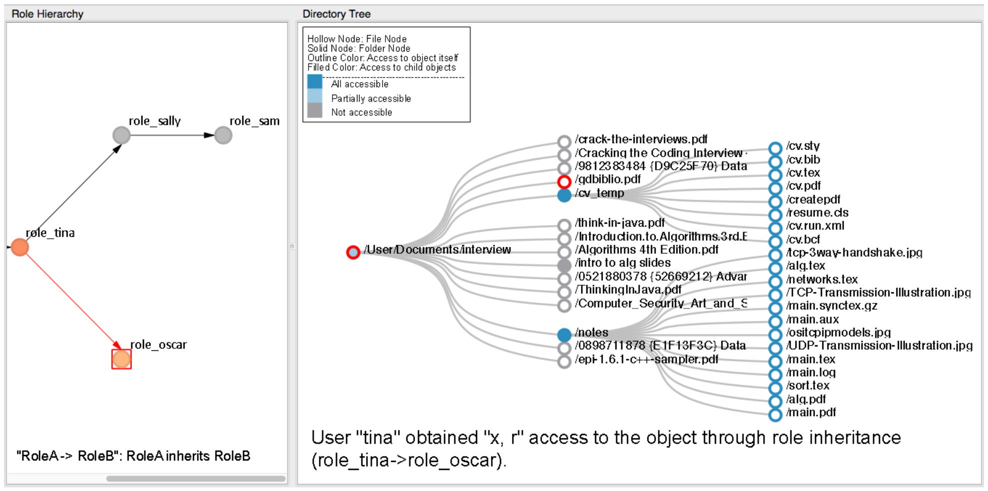
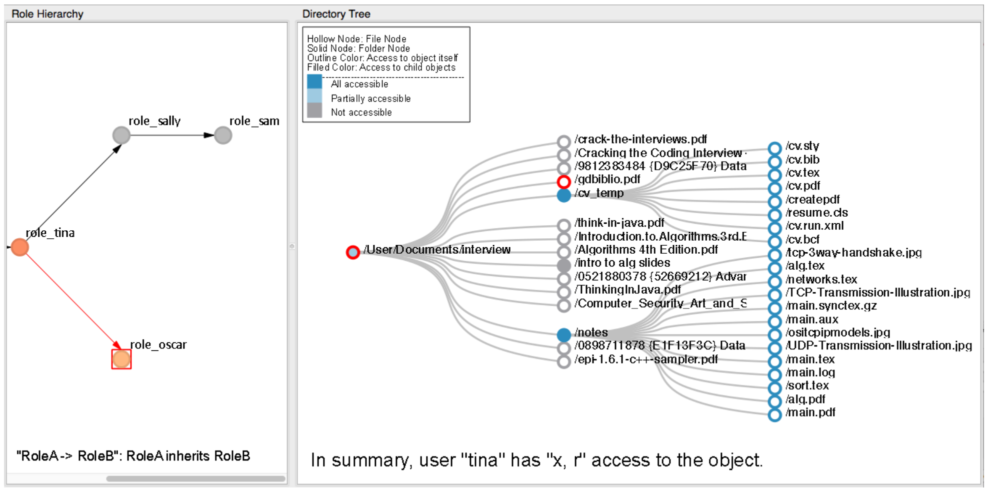

Role Hierarchy
role_sally
role_sam
role_tina
role_oscar
"RoleA-> RoleB": RoleA inherits RoleB
Directory Tree
Hollow Node: File Node
Solid Node: Folder Node
Outline Color: Access to object itself
Filled Color: Access to child objects
All accessible
Partially accessible
Not accessible
/User/Documents/interview
/crack-the-interviews.pdf
/Cracking the Coding Interview -
/9812383484 {D9C25F70} Data
/gdbiblio.pdf
/cv_temp
/think-in-java.pdf
/Introduction.to.Algorithms.3rd.E
/Algorithms 4th Edition.pdf
/intro to alg slides
/0521880378 {52669212} Advar
/ThinkingInJava.pdf
/Computer_Security_Art_and_S
/notes
/0898711878 {E1F13F3C} Data
/epi-1.6.1-c++-sampler.pdf
/cv.sty
/cv.bib
/cv.tex
/cv.pdf
/createpdf
/resume.cls
/cv.run.xml
/cv.bcf
/tcp-3way-handshake.jpg
/alg.tex
/networks.tex
/TCP-Transmission-Illustration.jpg
/main.synctex.gz
/main.aux
/ositcpipmodels.jpg
/UDP-Transmission-Illustration.jpg
/main.tex
/main.log
/sort.tex
/alg.pdf
/main.pdf
However, there is no access to the object from directly assigned roles.

Figure 4.12: Permissions from Directly Assigned Role



User "tina" obtained "x, r" access to the object through role inheritance (role_tina->role_oscar).

Figure 4.13: Inherited Roles of User tina



In summary, user "tina" has "x, r" access to the object.

Figure 4.14: Effective Permission of User tina

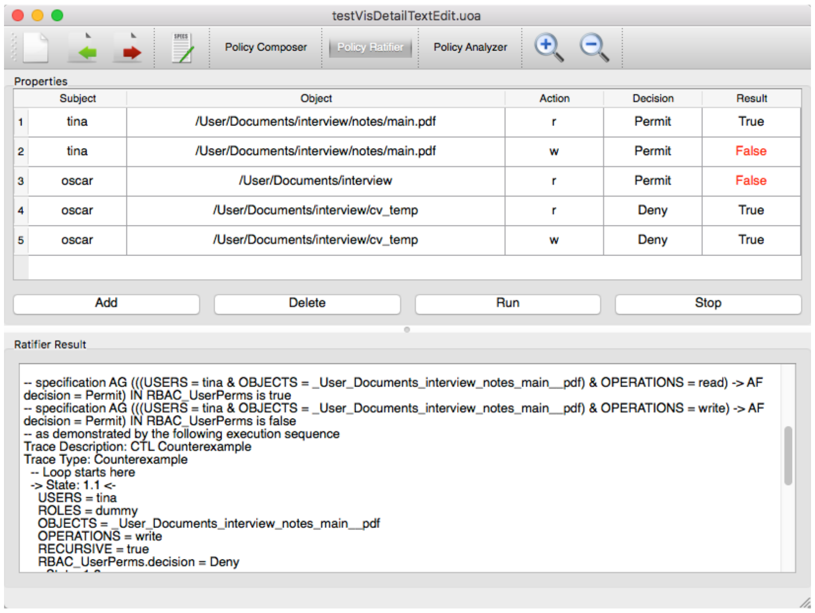
4.4.3 Object View

Another common type of query goes the opposite way to the user-to-object query. The Object View is provided to answer questions of whether an object is accessible to any user and which users have access. The view works the opposite way as the User View. An object must be selected. Then the users in the User List that have access to the object will be shown in bold font. The exact permissions and related rules can be displayed at the selection of a user. The simple version and detail version work in the same manner as those of the User View.

4.5 Policy Ratification

The Policy Ratifier provides a policy property check with the assistance of the NuSMV model checker. NuSMV [28] is a symbolic model checker developed by FBK-IRST, Carnegie Mellon University, and University of Trento. It allows a user to write specifications of synchronous to asynchronous finite state systems and checks in the temporal logic CTL. The input language, similar to that of SMV, states the transition relation of a finite Kripke structure. It provides great flexibility, but also requires great efforts to learn and could introduce the danger of inconsistency for non-expert users. Our ratifier exempts users from the work of learning the SMV language and writing SMV input files for the model checker. Users will only need to add a property consisted of a user, an object, a type of permission, and a decision (Permit/Deny) to check against the running policy in the system.

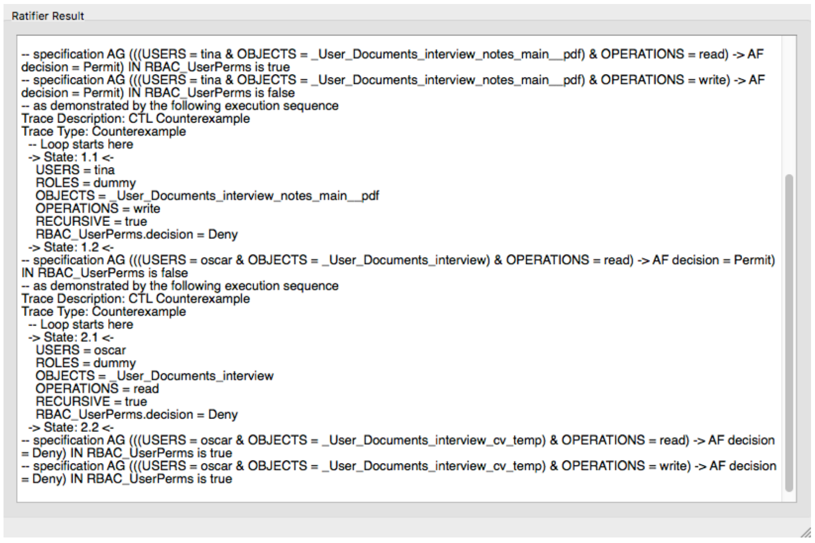
Figure 4.15 shows the interface of the Policy Ratifier. Properties could be added in the top table with an additional result column for the display of result. If the result is false, it will be in red for visibility, and a counterexample with state value to each variable will be given in the bottom window in texts (Figure 4.16).



	Subject	Object	Action	Decision	Result
1	tina	/User/Documents/interview/notes/main.pdf	r	Permit	True
2	tina	/User/Documents/interview/notes/main.pdf	w	Permit	False
3	oscar	/User/Documents/interview	r	Permit	False
4	oscar	/User/Documents/interview/cv_temp	r	Deny	True
5	oscar	/User/Documents/interview/cv_temp	w	Deny	True

```
-- specification AG (((USERS = tina & OBJECTS = _User_Documents_interview_notes_main__pdf) & OPERATIONS = read) -> AF
decision = Permit) IN RBAC_UserPerms is true
-- specification AG (((USERS = tina & OBJECTS = _User_Documents_interview_notes_main__pdf) & OPERATIONS = write) -> AF
decision = Permit) IN RBAC_UserPerms is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
  -- Loop starts here
  -> State: 1.1 <-
    USERS = tina
    ROLES = dummy
    OBJECTS = _User_Documents_interview_notes_main__pdf
    OPERATIONS = write
    RECURSIVE = true
    RBAC_UserPerms.decision = Deny
```

Figure 4.15: Policy Ratifier



```
-- specification AG (((USERS = tina & OBJECTS = _User_Documents_interview_notes_main__pdf) & OPERATIONS = read) -> AF
decision = Permit) IN RBAC_UserPerms is true
-- specification AG (((USERS = tina & OBJECTS = _User_Documents_interview_notes_main__pdf) & OPERATIONS = write) -> AF
decision = Permit) IN RBAC_UserPerms is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
  -- Loop starts here
  -> State: 1.1 <-
    USERS = tina
    ROLES = dummy
    OBJECTS = _User_Documents_interview_notes_main__pdf
    OPERATIONS = write
    RECURSIVE = true
    RBAC_UserPerms.decision = Deny
  -> State: 1.2 <-
-- specification AG (((USERS = oscar & OBJECTS = _User_Documents_interview) & OPERATIONS = read) -> AF decision = Permit)
IN RBAC_UserPerms is false
-- as demonstrated by the following execution sequence
Trace Description: CTL Counterexample
Trace Type: Counterexample
  -- Loop starts here
  -> State: 2.1 <-
    USERS = oscar
    ROLES = dummy
    OBJECTS = _User_Documents_interview
    OPERATIONS = read
    RECURSIVE = true
    RBAC_UserPerms.decision = Deny
  -> State: 2.2 <-
-- specification AG (((USERS = oscar & OBJECTS = _User_Documents_interview_cv_temp) & OPERATIONS = read) -> AF decision
= Deny) IN RBAC_UserPerms is true
-- specification AG (((USERS = oscar & OBJECTS = _User_Documents_interview_cv_temp) & OPERATIONS = write) -> AF decision
= Deny) IN RBAC_UserPerms is true
```

Figure 4.16: NuSMV Result

In our system, there are two types of property checks: the Comprehensive Property Check and the Custom Property Check. The Comprehensive Property Check is triggered by clicking the run button with an empty property table. Appendix A has an example of the Comprehensive Property Check in the SMV language. It automatically generates an SMV file from the RBAC specification output by the authoring component. The following properties are checked:

1. **Cyclic Role Inheritance Check**

   Cyclic role inheritance happens when a role inherits another role that has already inherited it. This will generate a cycle in the role hierarchy, and is in need of inspection since it either indicates a problem in permission assignment or the existence of redundant roles. The cyclic inheritance check is based on two facts: 1) If a role inherits its ancestor role (cyclic inheritance occurs), then this role will have the permissions its ancestor has; and 2) A role must have some permissions that its descendants don't have. Therefore, we find cyclic inheritance by checking whether the permissions of any junior role is no longer a proper subset of those of its senior roles.

2. **Complete Role Permission Check**

   Checks if all role are specified the intended permissions. Properties that list all permissions given to a role are written in this part. If there is a difference, a counterexample would be given.

3. **Complete User Permission Check**

   Checks if all users are assigned the intended permissions. Properties that list all permissions given to users are written in this part. If there is a difference, a counterexample would be given.

The Custom Property Check allows users to check properties of their interest. Properties must be added into the upper table and the check is triggered by clicking run. The following properties are supported.

1. **User Permission Check**

   Checks if a user has the specified permission to certain object. The values of a user, an object, the permissions, and the permission recursiveness should be provided.

2. **User Accessibility Check**

   Checks if a user has the specified permission to any object. This type of property is added by choosing Any for Object.

3. **Object Accessibility Check**

   Checks if an object can be accessed by any user with specified permission. This property is added by choosing Any for User.

4.6 Evaluation

4.6.1 Environment, Procedure and Goals

We evaluated the effectiveness of the tool in three main aspects: 1) whether our tool makes writing access control policies easier, 2) the effectiveness of our tool in helping analyzing access control policies, where the values and relationship between user, role and object can be fully understood, and access queries can be correctly answered , and 3) the general usability.

The evaluation was conducted at the Graphics Lab in the Department of Computer Science at Michigan Technological University. The participants are from different majors; one participant is from Physics, one is in Chemical Engineering, three major in Math, and 15 are in Computer Science, Computer Engineering and related majors. All of the participants except one, who took the Computer Security course before, did not have prior knowledge of the RBAC model.

The participants were first given a brief introduction to access control, asked to complete tasks using the visualization system, and then complete questions about the effectiveness of our tool and their understanding of the RBAC model.

The introduction covered the meaning and application of access control, the concepts of subject, object, and permission, and how RBAC implements access control through roles and role inheritance. Then participants were asked to write a policy based on given access requirements, and answer questions about a second policy. Specific instructions were given on how to use the tool to answer each questions. After the activity was completed, participants were given a set of RBAC technical questions, which were answered without the help of the visualization system. Lastly, the participants completed an evaluation form regarding the use of the tool. The whole process was conducted near the end of a semester on a voluntary basis, and 20 valid set of answers and evaluation forms were collected.

4.6.2 Test Problems

The question set consists of 15 questions about the policy written and policy answered using the tool (tool questions) and eight RBAC technical questions (see Appendix B). Each question counted as 1 point. The tool questions cover each component of the tool and gather information of whether the tool is effective in helping policy writing and analysis. They also allow the participants to obtain user experience of each component for further evaluation feedback.

Table 4.3
The Means (μ), Standard Deviation (σ), and Confidence Intervals (CI^-, CI^+) of Tool Question Groups

	Role View	User View	Object View	Ratifier
μ	0.84	0.84	0.80	1.00
σ	0.36	0.37	0.41	0.00
CI^-	0.68	0.68	0.62	1.00
CI^+	0.99	1.00	0.98	1.00

Table 4.4
The Means (μ), Standard Deviation (σ), and Confidence Intervals (CI^-, CI^+) of the Policy Analysis Tool Questions

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Total
μ	0.90	0.65	0.90	1.00	0.80	1.00	0.90	0.90	0.70	0.70	0.80	1.00	1.00	1.00	0.88
σ	0.31	0.49	0.31	0.00	0.41	0.00	0.31	0.31	0.47	0.47	0.41	0.00	0.00	0.00	0.33
CI^-	0.77	0.44	0.77	1.00	0.62	1.00	0.77	0.77	0.49	0.49	0.62	1.00	1.00	1.00	0.73
CI^+	1.00	0.86	1.00	1.00	0.98	1.00	1.00	1.00	0.91	0.91	0.98	1.00	1.00	1.00	1.00

The tool questions consist of a policy writing tool question and 14 policy analysis tool questions. The policy writing tool question gives a number of access requirements and ask the participants to use the Policy Composer to convert the textual requirements into a usable policy. Our result shows that all participants were able to write a policy fulfilling the given access requirements without prior knowledge of access control.

As for the policy analysis, Tables 4.3 and 4.4 contain the means, the standard deviations and the confidence intervals at 95% significance level of mean of the policy analysis question groups and individual question, respectively. We categorize the questions into groups based on their relevant components: group of Role View has Q1-Q5, group of User View includes Q6-Q10, group of Object View contains Q11 and group of Ratifier includes Q12-Q14. For the tool questions, participants obtained the answers through instructed operations on ACvisual. That is, the tool helps finding answers to the questions without requiring the user's knowledge of access control. The score, hence, is an indication of whether the tool helps in analyzing policies. The overall mean value indicates a correctness of 88%. The means within the Policy Ratifier group (Q12-Q14) show that it was very effective. The lowest mean is for the Object View component, but it still has a high value of 0.8. From the mean values of individual questions in Table 4.4, we can find the factors that need improvement within each group. Q2 and Q5 of the Role View, Q9 and Q10 of the User View received the lowest rates of correctness. This means that finding the roles that inherit a particular role and the permissions to objects at deeper levels of the file system can still be difficult for users. More explanation or better illustration may be needed to help provide the answers. But overall the correctness of 88% shows that the tool provided significant help in analyzing policies.

Table 4.5
The Means (μ), Standard Deviations (σ), and Confidence Intervals (CI^-, CI^+) of RBAC Questions

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Total
μ	1.00	1.00	1.00	1.00	1.00	1.00	0.60	0.70	0.91
σ	0.00	0.00	0.00	0.00	0.00	0.00	0.50	0.47	0.28
CI^-	1.00	1.00	1.00	1.00	1.00	1.00	0.38	0.49	0.79
CI^+	1.00	1.00	1.00	1.00	1.00	1.00	0.82	0.91	1.00

Table 4.5 shows the means, standard deviations and confidence intervals of RBAC technical questions. The questions were designed in a way that could not be incorporated into and solved through ACvisual. As a result, participants had to answer the questions solely based on their understanding of RBAC after using the tool. Their answers to the question then can be used as a measurement of the effectiveness of ACvisual in improving the understanding of RBAC. Based on a user-role-assignment matrix and a role-object-permission matrix, the questions included roles a certain user can occupy (Q1), the permissions a user can have to different objects (Q2-Q6), and inheritance between roles (Q7 and Q8). From Table 4.5, the overall mean shows a 91% of correctness. The first six questions were answered correctly by all participants, while Q7 and Q8 received a mean value of 0.6 and 0.7, respectively. This means that participants could understand the permissions through user, role, and object specification clearly. But the role inheritance still remains the most challenging

aspect of RBAC. The reason why mistakes occurred in this part can be twofold. Participants who are not clear about how the inheritance relation is formed can answer the questions incorrectly. Also, participants who understand the topic can still make careless mistakes as the topic is error-prone by nature. Even though our tool builds the relationship between roles for users without asking their understanding of the topic, it would be better if there is a functionality to help, for instance a step-by-step demonstration of role inheritance construction could be added.

4.6.3 Evaluation Form

Our evaluation form contains a set of rating and usage questions (Table 4.6) and write-in questions to collect information on participants' perception of the effectiveness of ACvisual. The time spent on understanding RBAC and using the tool were also recorded. The first 12 rating questions study the effectiveness of ACvisual. Q1 The choices are: 1:strongly disagree, 2:disagree, 3:neutral, 4:agree, and 5:strongly agree. Q13 and Q14 study the time participants spent on understanding the RBAC model and using the tool. The choices for Q13 are 1: less than 5 mins, 2: 5-10 mins, 3: 11-15 mins, 4: 16-30 mins, and 5: more than 30 mins. The choices for Q14 are 1: less than 5 mins, 2: 5-15 mins, 3: 16-30 mins, 4: 31-60 mins, and 5: more than 60 mins.

Table 4.6
ACvisual Rating and Usage Questions

Rating Questions	
Q1	User-Object-Action policy is easy to write for defined requirements
Q2	Policy Composer made it easy to create and edit a policy
Q3	Policy Ratifier made it easy to check access properties of a policy
Q4	Policy Analyzer was helpful for studying a policy
Q5	User View made accessible objects and roles of a user clear
Q6	Object View makes users and roles that have access to an object clear
Q7	Role View makes it easy to look up permissions and inheritance of roles
Q8	Use of colors in the visualization can easily distinguish different items
Q9	Fonts in the visualization is clear and in proper size
Q10	I feel more comfortable writing access control policies after using ACvisual
Q11	I understood RBAC model after using ACvisual
Q12	The software was easy to use
Usage Questions	
Q13	How long did it take to understand RBAC using ACvisual
Q14	How long did you use ACvisual

4.6.3.1 General Discussion

Table 4.7 and Figure 4.17 have the means, the standard deviations, and the confidence intervals at 95% significance level of rating and usage questions. Rating questions (Q1-Q12) received a positive feedback with an average score of 4.08 and a standard deviation of 0.64. Q6 and Q7 received the highest ratings of 4.40. This means that the Object View showed the users and roles that have access to an object clearly and that the Role View made it easy to look up permissions and inheritance of roles. Q8 and Q9 on the choice of color and font size for visualization received the lowest ratings of 3.67 and 3.30, respectively. Some participants later indicated in the write-in questions that the colors are similar for directly assigned and inherited roles of a user in the Role View. This could be solved by using more distinguishable colors. As for the font size, the problem may lie in the initial display of Object Tree in the User View and Object View. For large scaled file systems, the Object Tree can have many nodes, and the initial display intends to present an overview of the object hierarchy along with access information. Thus, the fonts can be small in the beginning. But with the provided zooming and scrolling, users can make adjustment to the font size. For a better picture of the rating of each aspect mentioned at the beginning, questions were divided into four groups: 1) Policy Writing of Q1, Q2 and Q10 (**G1**), 2) Policy Analysis of Q3-Q7 (**G2**), 3) General Usability of Q8 and Q9 (**G3**), and 4) RBAC Understanding of Q11 (**G4**). From the group means in the last four columns of

Table 4.7, **G1**, **G2** and **G4** received positive feedback with mean values greater than 4.10 while the interface's lowest mean indicating a need of improvement.

Table 4.7
The Means (μ), Standard Deviations (σ), and Confidence Intervals (CI^-, CI^+) of ACvisual Rating Questions and Question Groups

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	G1	G2	G3	G4
μ	4.30	4.20	4.05	4.20	4.25	4.40	4.40	3.67	3.30	4.00	4.00	3.90	2.71	3.62	4.15	4.24	3.65	4.10
σ	0.48	0.42	0.63	0.63	0.63	0.52	0.48	0.79	0.67	0.67	0.74	0.32	1.14	0.79	0.48	0.57	0.70	0.72
CI^-	4.09	4.02	3.77	3.92	3.97	4.17	4.19	3.32	3.00	3.71	3.68	3.76	2.21	3.27	3.94	3.99	3.34	3.79
CI^+	4.51	4.38	4.33	4.48	4.53	4.63	4.61	4.01	3.60	4.29	4.32	4.04	3.20	3.96	4.36	4.49	3.96	4.41

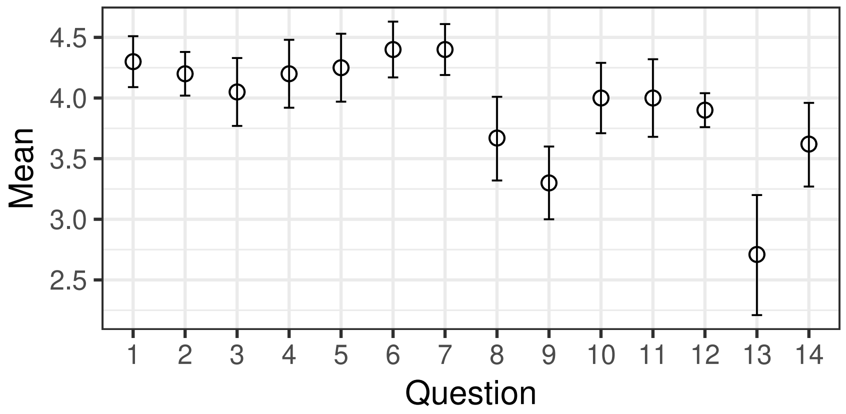


Figure 4.17: The Means with Confidence Intervals of ACvisual Rating and Usage Questions

The last two questions Q13 and Q14 are about the usage of the tool. The means of Q13 and Q14 are 2.71 and 3.62 with standard deviations of 1.14 and 0.79, respectively. Table 4.8 has the distribution of answers. On Q13, 70% of participants selected Choice 1, 2 and 3. This indicates that the majority of the participants understood the RBAC model within 15 minutes. As for Q14, 90% of the participants selected Choice 2 and 3, meaning that their use of our tool is between 30 to 60 minutes. The participants who chose Choice 3 were able to finish the tool questions in less than half of the intended time (60 minutes), which shows that the tool was easy to use. But still, there are a small number of participants who used the tool for more than an hour.

Table 4.8
Usage Distribution

	Choice1	Choice2	Choice3	Choice4	Choice5
Q13	10%	25%	35%	25%	5%
Q14	0	0	35%	55%	10%

4.6.3.2 Statistical Analysis

Spearman rank correlation test was applied to further investigate the rating correlation of each question pair. Out of the 91 pairs of correlations (excluding correlations to oneself) in Figure 4.18, only 6 pairs had a p-value less than the level of significance $\alpha = 0.05$. This means that nearly 94% of the question pairs did not have a significant monotonic correlation. Out of the six correlated pairs, Q5, Q6, Q10, and Q11 have $\rho(Q5, Q6) = 0.87$, $\rho(Q5, Q10) = 0.79$, $\rho(Q5, Q11) = 0.86$, $\rho(Q6, Q11) = 0.85$, and $\rho(Q10, Q11) = 0.82$. This means that participants who rated high in the clear representation of User View (Q5) and Object View (Q6), would be very likely to rate high in questions of “feel more comfortable writing policies after using ACvisual” (Q10) and “understood RBAC after using ACvisual” (Q11). Moreover, Q1 and Q2 have a correlation of 0.76 with a p-value less than 0.05. This shows that participants who found “the User-Object-Action policy easy to write” (Q1) would also provide a high rating to “Policy Composer made it easy to create and edit a specification” (Q2). This is because the Policy Composer uses an interface with key elements from the User-Object Action language. So participants who already know one of them could pick up the other form of presentation easily.

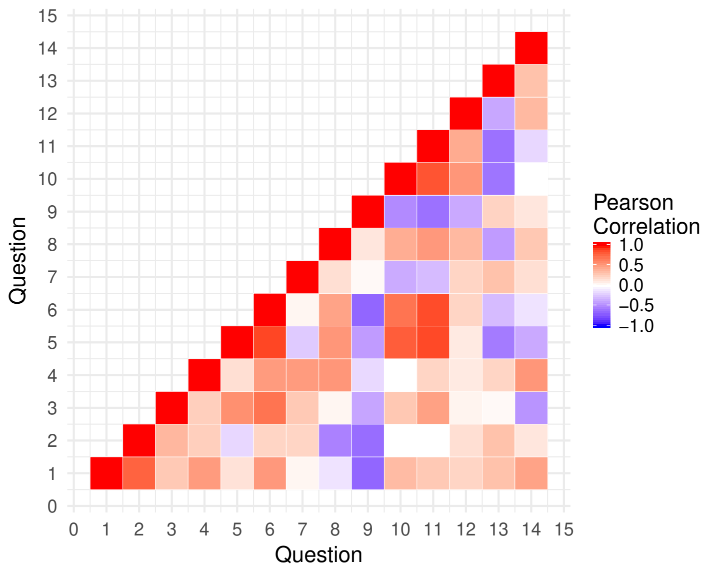


Figure 4.18: Correlation Heatmap of Rating Questions

We also looked into the differences among ratings using the Student’s t-test. Pairwise t-test shows that only Q9 with all questions except Q8 had p-values less than 0.05. This suggests that for all the rest of question pairs, the null hypothesis that the questions were rated equally cannot be rejected. Combined with the mean values, it also shows that the rating of font size is significantly lower than all other questions.

4.6.4 Evaluation Form - Student Comments

The four write-in questions were used to collect information of the participants' major, their thoughts on the most and least useful features of the tool, parts to improve, and problems during installation and using the tool.

Five out of the nine participants considered User View as the most useful feature. They wrote that "*the permission of a user is clear and easy to locate*" and "*the animation is easy to understand of the relationship between users, roles and objects*". Two other participants preferred the Role View. They stated that "*the role hierarchy made the relation between roles clear*" and that "*the graph is easy to interpret and adding/deleting permissions is easy*".

Some suggestions were also given for further improvement. Many participants mentioned that the font used in the Object Tree could be bigger, and that an introduction tutorial could be added to the package. There was no installation problem reported, and one participant encountered an out-of-bound problem when importing a policy.

4.6.5 Conclusion

We measure our goals through three types of feedback: the tool questions, the RBAC technical questions, and the evaluation questions. The first goal, easing policy writing, can only be measured through the subjective evaluation feedback. All participants' being able to write correct policies given textual access requirements and **G1**'s mean of 4.15 shows that users found writing policies through the tool is effective and easy. Q1 and Q2's high ratings also show positive feedback on both User-Object-Action language and the policy template in Policy Composer. Participants also mentioned that they felt more comfortable writing a policy after using the tool, particularly after being able to analyze policies through User View and the Object View.

The second goal, helping in analyzing policies, is measured through the correctness of tool questions and the feedback of evaluation questions. The tool questions were answered through using the tool and did not require prior knowledge of access control. Participants on average were able to answer 88% of the questions correctly. Also, in the evaluation feedbacks, **G2** received the highest rating of 4.24 out of 5 points.

The third goal about the intuitiveness of user interface is also measured through evaluation questions. **G3** received the lowest mean of 3.65. It shows a positive feedback but also indicates room for improvement.

Lastly, we were interested in knowing whether the tool helped in understanding RBAC model. The RBAC technical questions and evaluation questions were used to measure the effectiveness. Participants were able to answer 91% of the questions correctly with no help. The evaluation feedback also shows that they could understand RBAC model after using the tool for 15 minutes.

In summary, We received positive feedback on the effectiveness of ACvisual in making writing access control policies easier, in helping policy analysis, in the user interface, and in helping the understanding of RBAC model. In addition to being components of policy analysis, User View and Object View also helped in making policy writing as well as understanding RBAC easier. However, the user interface still needs some rework: the colors designated to directly assigned and inherited roles should be more distinguishable, and the font size in Object Tree should be improved.

Chapter 5

Flow Visualization Systems

This chapter summarizes the research I had been involved in the field of fluid dynamics education. We start by briefly introduce the background of fluid dynamics and the current state of teaching in this field, and the need of developing FlowVisual.

Fluid mechanics and computational fluid dynamics (CFD) are among the core courses in many engineering majors such as mechanical engineering, aerospace engineering, biomedical engineering, chemical engineering, and civil engineering. In these courses, it is important for students to acquire knowledge of fundamental flow field concepts. Many of those concepts are not straightforward to learn. For instance, it is not easy for beginning-level students to fully understand the differences between various kinds of field-lines and critical points. Commonly, these materials are taught by instructors through explaining concepts and definitions, drawing diagrams and illustrations, and occasionally, playing custom-made animations or video clips. Using intuitive and real flow examples proves to be an excellent way of learning. However, most examples available today are only designed for lecture or demonstration but not for student interaction or self-learning. Developing a pedagogical visualization tool holds the potential to help students better learn these essential flow field concepts through interactive exploration.

FlowVisual is a tool we developed to facilitate the teaching of flow field concepts. It consists of two individual tools, a desktop version for 2D flow field data and a mobile version for 3D flow field data. Both of the tools have been evaluated through a formal user study involving students from mechanical engineering, electrical engineering, and computer science at Michigan Technological University. The desktop version has been used in classroom teaching of CFD course for multiple times and has received positive feedback from students. We also found that the app version helped students with no previous 2D flow field knowledge understand concepts to the similar degree of students who had studied those concepts before.

We have released FlowVisual desktop online and FlowViusal mobile in App Store along with the tutorial so that other instructors and students who are interested in our work can benefit as well, making it truly useful for teaching and learning fluid dynamics and flow visualization. In the following sections, we first introduce the concepts in flow field. Then we will focus on the functionality to the tools, and the evaluations are skipped.

5.1 Terms

This section gives a brief introduction to some important concepts of flow fields that are incorporated into our FlowVisual.

Flow Field A flow field (or vector field) is an assignment of a velocity vector to each point in the domain to represent the movement of the flow. Essentially, it is a mapping

$$F(\mathbf{p};t) = \mathbf{v} \tag{5.1}$$

that assigns a vector $\mathbf{v}$ to each point $\mathbf{p}$ at time t. Mathematically, a flow field could be expressed as a differential equation

$$\frac{d\mathbf{p}}{dt} = \mathbf{v}(\mathbf{p};t). \tag{5.2}$$

Steady and Unsteady Flow When all the time derivatives of a flow field vanish, the flow is considered to be a steady flow. In other words, steady flow refers to the condition where the fluid properties at each point in the system do not change over time. When time does affect the behavior of the flow, we consider the flow as an unsteady flow.

Streamline A streamline is the trajectory that a massless particle follows if released in a steady flow field. It is also known as the curve that is everywhere tangent to the vectors it passes through. Mathematically, a streamline is the solution from $\mathbf{p}_c = ((x_c, y_c, z_c); t_c)$ constrained in an instantaneous vector field of $\mathbf{v}(\mathbf{p}_c; t_c)$ at time t_c, and it can be represented by

$$\mathbf{p}(b) = \mathbf{p}_c + \int_0^b \mathbf{v}(\mathbf{p}(\sigma); t_c) d\sigma. \tag{5.3}$$

Pathline A pathline is the trajectory that an individual particle follows in an unsteady flow field. Given a flow field $\frac{d\mathbf{p}}{dt} = \mathbf{v}(\mathbf{p}; t)$, the solution with initial state $\mathbf{p}_0 = ((x_0, y_0, z_0); t_0)$ is

$$\mathbf{p}(t = b) = \mathbf{p}_0 + \int_0^b \mathbf{v}(\mathbf{p}(\sigma); t_0 + \sigma) d\sigma, \tag{5.4}$$

which is referred to as a pathline starting at position $\mathbf{p}_0$.

Streakline A streakline is the locus of points of all the fluid particles that have passed continuously through a particular spatial point in the past. Given a set of pathlines traced from the same position at different time steps, connecting all points at the same time step forms a streakline.

Timeline A timeline is a line formed by a set of fluid particles that were marked at a previous instant in time, creating a curve that is displaced over time as the particles evolve. Given a set of pathlines traced from the same time step at different positions, connecting all the points at the same time step forms a timeline.

Line Integral Convolution (LIC) A LIC image uses a dense texture to depict a complete overview of a 2D flow field or a slice of a 3D flow field. It works by integrating a random static pattern of black-and-white paint sources with the flow field data to visualize specific part of the flow field. As the flow passes by the sources, each fluid particle picks up some of the source intensity. The result is a random striped texture where points along the same streamline tends to have similar intensities.

Stream Surface A stream surface is a continuous surface that is tangent to the vector at every point it passes through, which can be obtained from streamlines traced from a densely seeded curve.

Critical Point A point $\mathbf{p}$ is called a critical point of $\mathbf{v}(\mathbf{p}; t_c)$ if $\mathbf{v}(\mathbf{p}; t_c) = 0$. Critical points are crucial because they are enclosed by their compact neighborhood with distinct patterns determined by their types.

5.2 FlowVisual for 2D Flow Field

FlowVisual[1] desktop to facilitate the teaching and learning of fundamental concepts in fluid dynamics. Given a 2D flow field, our tool allows users to drop seeds into the field. We depict the paths that the seeds will follow at any point in time so that the users can observe the integral lines. Besides point seeding for single field-lines, we also enable rake seeding so that a group of field-lines can be traced simultaneously for efficiency. Our tool includes streamline visualization for steady flow fields and pathline, timeline, and streakline visualization for unsteady flow fields. Visually comparing streamlines with streaklines, pathlines with timelines, and pathlines with streaklines allow students to easily understand the similarities and differences among these integral lines, which may be difficult to comprehend without visual explanation and interrogation. Besides various field-lines, we also compute and display the line integral convolution texture so that users can intuitively grasp an overview of the underlying flow data. To enable effective feature identification, we detect and analyze critical points of various kinds and highlight them in the visualization via template-based seeding.

Our visualization interface is implemented using QT 4.0 and OpenGL. Figure 5.1 shows a screenshot of the user interface. It includes a rendering window on the left

[1]The material contained in this section was previously published in proceedings of the 2013 American Society for Engineering Education Conference [78]

and control panels on the right. The operation hint is displayed right below the rendering window. The program also supports hover hints for each option in the panels.

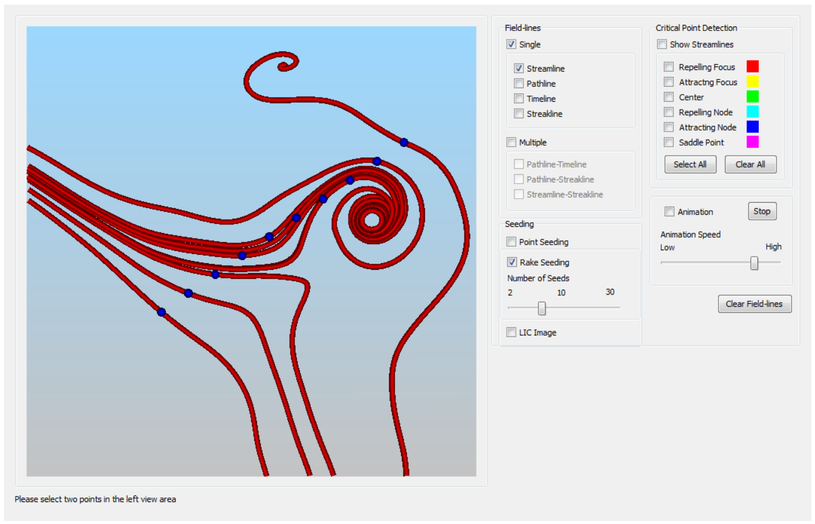

Field-lines
[✓] Single
[✓] Streamline
[ ] Pathline
[ ] Timeline
[ ] Streakline
[ ] Multiple
[ ] Pathline-Timeline
[ ] Pathline-Streakline
[ ] Streamline-Streakline
Seeding
[ ] Point Seeding
[✓] Rake Seeding
Number of Seeds
2 10 30
[ ] LIC Image
Critical Point Detection
[ ] Show Streamlines
[ ] Repelling Focus 🟥
[ ] Attractng Focus 🟨
[ ] Center 🟩
[ ] Repelling Node 🟦
[ ] Attracting Node 🟦
[ ] Saddle Point 🟪
[ ] Animation
Animation Speed
Low High
Please select two points in the left view area

Figure 5.1: The User Interface of FlowVisual Desktop. (© 2013 American Society for Engineering Education. Reprinted by permission.)

5.2.1 Field-line Visualization and Comparison

Field-line Tracing. Users can click on the rendering window to drop seeds for field-line tracing. Computing a series of points following the direction of the vector field captures the entire field-line. More precisely, the tangent line to the path at each point is required to be parallel to the vector at that point. In practice, the points are calculated by bilinearly interpolating vectors using the Runge-Kutta fourth order method.

Line Drawing. Figure 5.2 shows an example streamline with all three forms of visual representation we provide for field-lines: line, tube, and animated arrow. The first two forms show the entire streamline statically, while the last one dynamically conveys a vector direction as well as its magnitude along the streamline in an animated fashion. OpenGL functions were utilized for all the drawing.

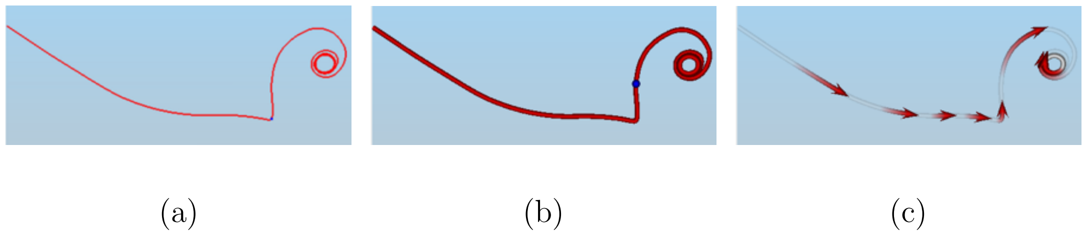
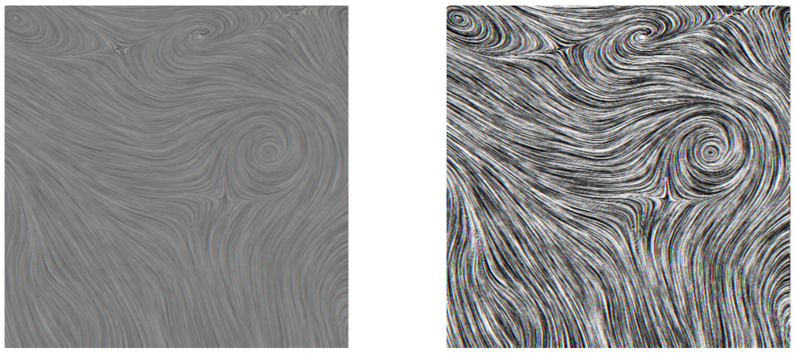


Figure 5.2: Visual Forms of Field-line Representation. (a) line. (b) tube. (c) animated arrow. (© 2013 American Society for Engineering Education. Reprinted by permission.)

LIC Texture. We generate the line integral convolution (LIC) texture to provide users with a background image showing an overview of the entire vector field. The LIC algorithm was introduced by Cabral and Leedom [19], and has been widely used in flow visualization. Figure 5.3 shows two versions of the LIC textures with different contrasts. LIC textures are used for both steady and unsteady flow fields. For unsteady flow fields, LIC images will be updated synchronously along with pathline, timeline, and streakline visualization and animation.



(a) (b)

Figure 5.3: LIC Texture Showing Underlying Flow Data. (a) original LIC. (b) LIC after histogram equalization. (© 2013 American Society for Engineering Education. Reprinted by permission.)

Field-lines. FlowVisual includes the visualization of four different types of field-lines: streamline, pathline, streakline, and timeline (see Figure 5.4). To distinguish different field-lines, each type of line is encoded with distinctive colors: red for streamlines, yellow for pathlines, green for streaklines, and magenta for timelines. Streamlines are traced in one single time slice, while all other field-lines are traced throughout multiple time slices.

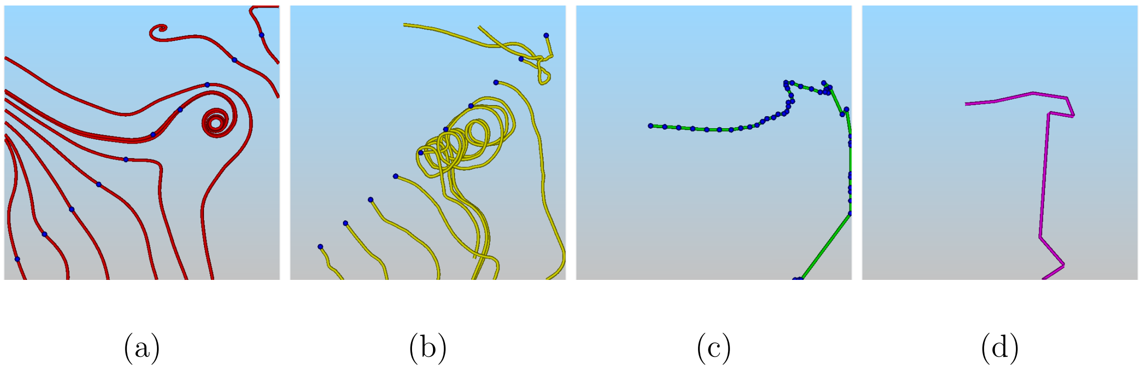


Figure 5.4: Single Field-lines. (a) streamlines. (b) pathlines. (c) streakline. (d) timeline. (© 2013 American Society for Engineering Education. Reprinted by permission.)

Multiple Field-line Comparison. To demonstrate the concepts of field-lines and the relationship among them, we also provide multiple field-line comparison in conjunction with animation. Since timeline and streakline are defined based on pathline, both timeline and pathline comparison and streakline and pathline comparison play a crucial role in helping users understand the formation of timeline and streakline. Example comparisons of pathline with timeline (a)-(c) and pathline with streakline (d)-(f) are drawn over time in Figure 5.5. In addition to the animation of field-lines, we update LIC textures synchronously showing the underlying unsteady flow field for reference. Users can adjust the animation speed or pause/resume the animation for detailed examination.

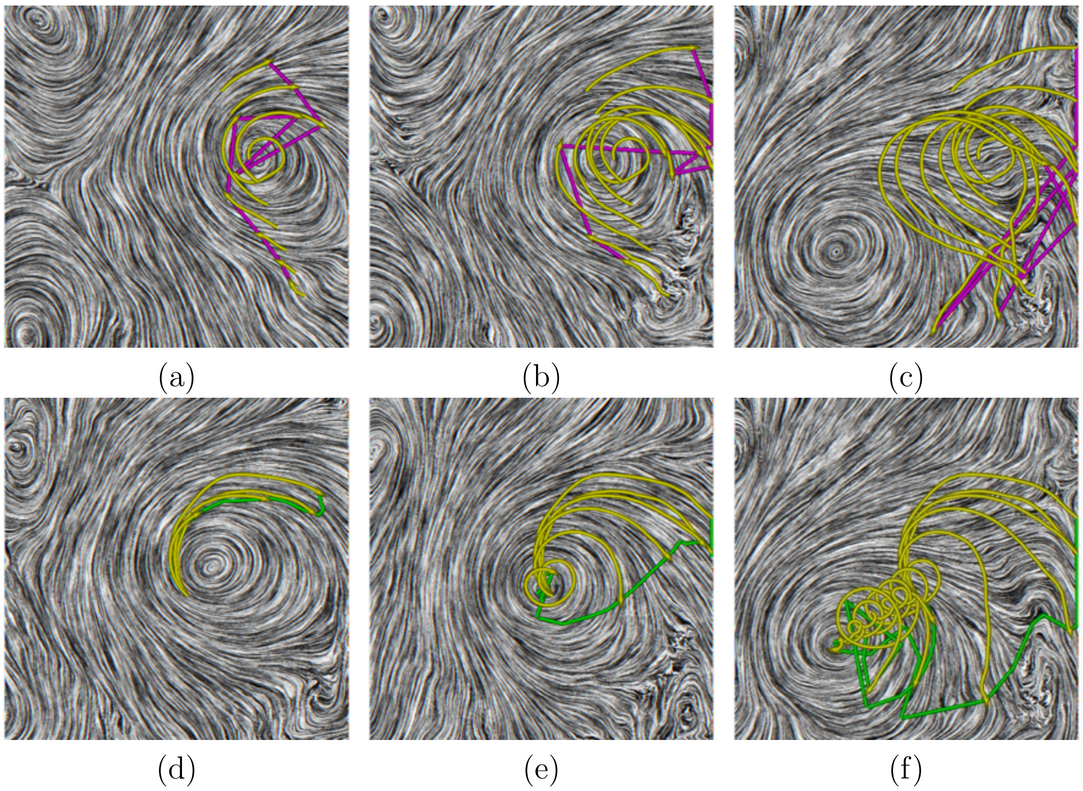


Figure 5.5: Multiple Field-line comparison with LIC texture overlay. (a)-(c) pathline and timeline. (d)-(f) pathline and streakline. (© 2013 American Society for Engineering Education. Reprinted by permission.)

5.2.2 Critical Points

Extracting features from vector fields has been a central research focus for decades. A great deal of work has been conducted to tackle this problem. In FlowVisual, we need to achieve the following two goals: (1) figuring out locations and types of critical points for a given vector field, and (2) designing templates for automatically placing streamlines to effectively highlight different types of critical points.

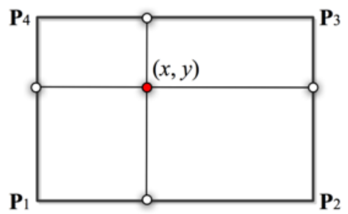


Figure 5.6: Interpolating a critical point. (© 2013 American Society for Engineering Education. Reprinted by permission.)

Critical Point Detection. As mentioned in the work of Helman and Hesselink [36], the vectors at critical points have to be zero. For discrete vector data, we detect critical points through sign checking and vector interpolation based on vectors defined on grid points. Specifically, for each 2D grid cell, we check whether there is at least one change of sign of the vectors at its four corners. If both the x and y vector components have sign change, then we interpolate the position of a critical point within the cell. Otherwise, there is no critical point in the cell. To compute the exact location of a critical point as shown in Figure 5.6, we have the following formula

$$(1-x)(1-y)P_1 + x(1-y)P_2 + xyP_3 + (1-x)yP_4 \tag{5.5}$$

Setting the above formula to zero, we have

$$(1-x)(1-y)P_1 + x(1-y)P_2 + xyP_3 + (1-x)yP_4 = 0 \tag{5.6}$$

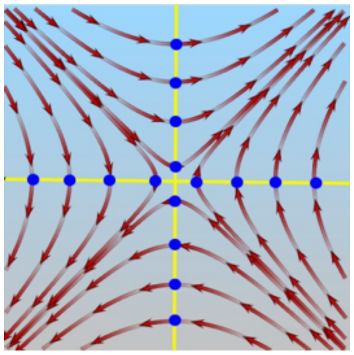
Therefore,

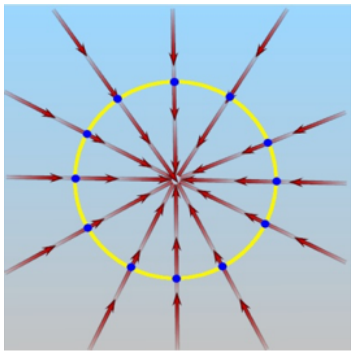
$$y = \frac{-P_1 + (P_1 - P_2)x}{(-P_1 + P_4) + (P_1 - P_2 + P_3 - P_4)x} \tag{5.7}$$

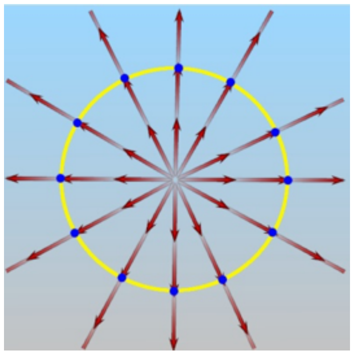
Let $-P_1 = C_1$, $P_1 - P_2 = C_2$, $-P_1 + P_4 = C_3$, and $P_1 - P_2 + P_3 - P_4 = C_4$, then

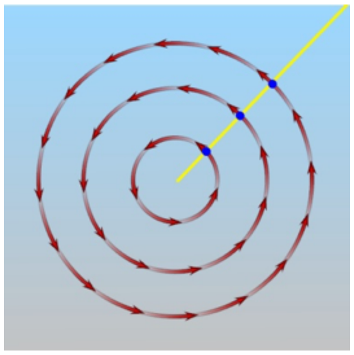
$$y = \frac{C_1 + C_2 x}{C_3 + C_4 x} = \frac{C_2}{C_4} + \frac{C_1 - \frac{C_2 C_3}{C_4}}{C_3 + C_4 x} \tag{5.8}$$

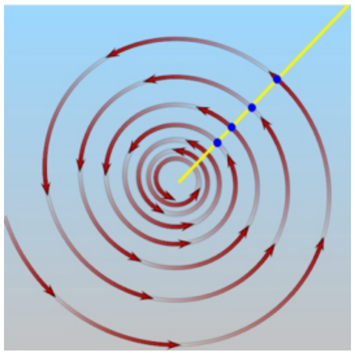
Thus, all the points with vector of (0, y) are on a hyperbola. Symmetrically, all the points with vector of (x, 0) are also on a hyperbola. By definition, the critical points in the cell are at the intersection of these two hyperbolas.


(a)


(b)


(c)


(d)


(e)

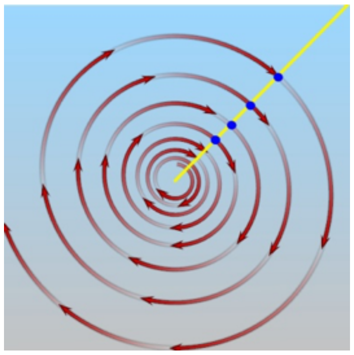

(f)

Figure 5.7: Six Types of Critical Points and their Seeding Templates. (a) saddle. (b) attracting node. (c) repelling node. (d) center. (e) attracting focus. (f) repelling focus. (© 2013 American Society for Engineering Education. Reprinted by permission.)

Critical Point Classification and Visualization. As shown in Figure 5.7, for 2D flow fields, we can classify critical points into six different types. This can be determined by distinguishing different types on the real and imaginary parts of the eigenvalues of the Jacobian matrix in the neighborhood of a critical point [36]. Since imaginary parts demonstrate the circulating flow pattern while real parts represent the repelling or attracting behavior of the flow, a straightforward analysis on both parts would determine the type of critical points. To effectively display critical points, we draw sphere-shape textures at the locations of critical points and add halos of different colors to indicate their types: magenta for saddle, blue for attracting node, cyan for repelling node, green for center, yellow for attracting focus, and red for repelling focus. Figure 5.8 (a) shows such an example.

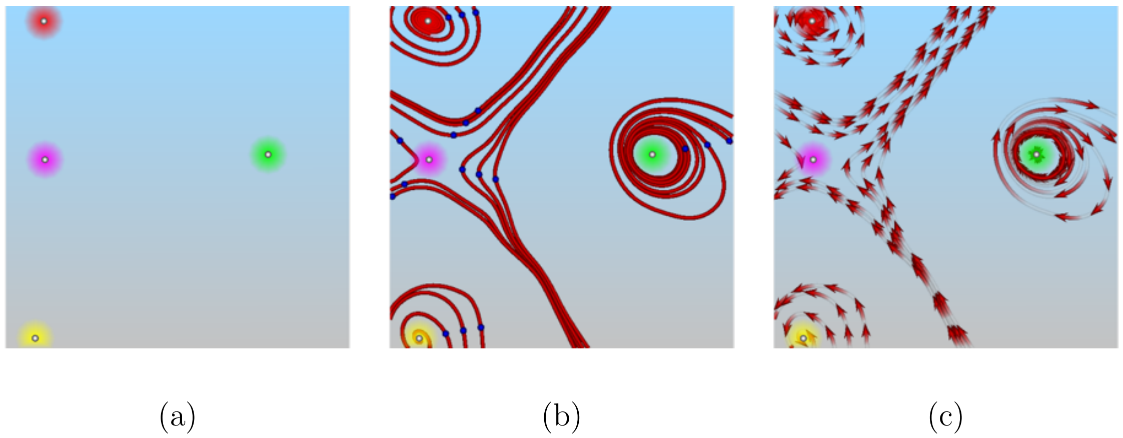


Figure 5.8: Visualization of Critical Points. (a) critical point highlighting. (b) static streamlines around critical points via template-based seeding. (c) dynamic streamlines with animated arrows. (© 2013 American Society for Engineering Education. Reprinted by permission.)

Since streamline patterns around distinct types of critical points are quite different from one another, streamline placement becomes important in order to effectively highlight the characteristics of critical points. We adopt a similar strategy proposed by Verma et al. [70] that applies a different seeding template for each type of critical point, as shown in Figure 5.7. For a saddle point, the template has the seeds distributed along the bisector of the hyperbola. But for an attracting/repelling focus or a center, the pattern forms a circular or spiral shape. Therefore, seeding on circles with an increasing radius is a good strategy. As for an attracting/repelling node, the streamlines are either toward/away from the critical point. The corresponding strategy is to place seeds evenly on a circle in a good distance from the critical point. The visualization of critical points with template-based seeding is shown in Figure 5.8.

5.3 FlowVisual for 3D Flow Field

FlowVisual[2] mobile is an educational app running on iOS devices to illustrate basic flow field concepts in 3D. This app is an extension of the desktop version of FlowVisual for 2D flow fields [78]. It is developed to illustrate the concepts in 3D space as cases in 3D are more common yet more challenging to understand in practice. Besides different kinds of field-lines, we also implemented stream surfaces in this app to enrich the perception of the flow field characteristics in a more continuous fashion. Our key

[2]The material contained in this section was previously published in the proceedings of 2016 Visualization and Data Analysis Conference [77]

deliverable is an app for classroom demonstration and for self-study by students and professionals. Its implementation on iPad makes it highly portable and accessible by anyone who is interested in learning and exploring key flow field concepts. Figure 5.9

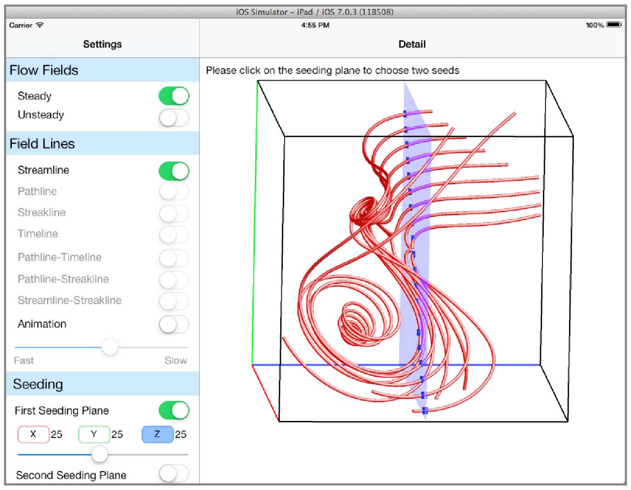


Figure 5.9: The user interface of the FlowVisual app. (© 2016 IS&T. Reprinted by permission.)

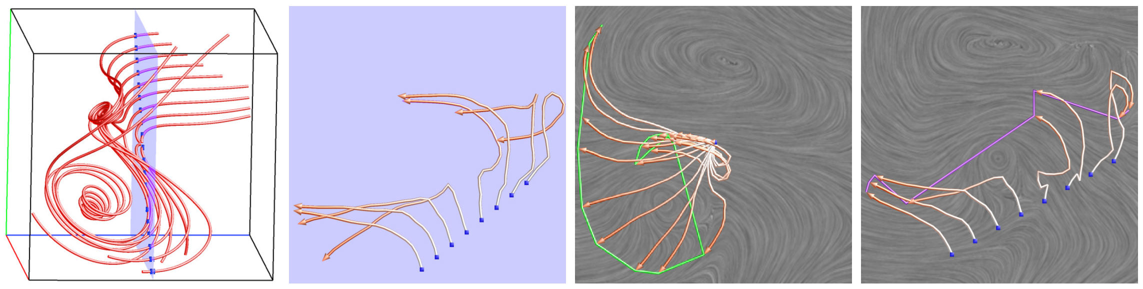
shows the user interface of the FlowVisual app. There are two major parts: a drawing canvas and a function panel. The drawing canvas is where users place seeds and where flow field concepts are visualized. The panel has three sections: field-line section, stream surface section, and critical point section. The field-line section supports visualization of different field-lines to help students understand the definitions and their similarities and differences. It includes the visualization of streamline, pathline, streakline, and timeline, as well as the comparisons of pathline and streakline, pathline and timeline, and streamline and streakline. We support two types of seeding: point seeding and rake seeding. LIC texture is also incorporated to provides an overview of the underlying flow to guide seed placement. The stream surface section supports multiple surface overview and single surface inspection with streamlines and streamline animation. The multiple surface overview provides an overall impression of the flow field by displaying multiple stream surfaces at the same time. The single surface inspection allows one surface to be examined along with streamlines and streamline animation, without occlusion from other surfaces. The critical point section supports the detection and classification of critical points and template-based seeding, which helps reveal the flow pattern around each critical point.

5.3.1 Field-line Visualization and Comparison

Seeding. To trace field-lines, users may specify any point in the domain as the seed. To ease the placement of seeds in the 3D space, we use seeding planes to fix the coordinate of one dimension and allow users to place seeds on the seeding planes. Users can use up to two seeding planes simultaneously. Either of the two planes can be switched between xy, yz or xz planes with adjustable z, x or y coordinates, respectively. Our app supports both point seeding and rake seeding. Point seeding allows only one seed at a time. Rake seeding, on the other hand, allows multiple seeds to be placed. Users are asked to click two end points. Based on the number of seeds (between 2 and 20) specified on the user interface, seeds will be placed evenly in between the chosen two end points.

Line Drawing. To trace the trajectory of a particle within a flow field, we need to solve for positions that the particle passes through using the differential equation representing the flow field. We employ the fourth-order Runge-Kutta method and depict the field-lines in two forms: solid tubes (see Figure 5.9) and animated arrows (see Figure 5.12). The tube form is depicted by simply connecting the points that we trace along the field-line. It shows the entire field-line with different colors representing different types of field-lines. As an extension of the static tube, animated arrows show the formation of filed-lines. In addition to drawing the entire line as the background using a transparent gray color, the animation uses dashed lines with arrowheads to indicate flow directions.

Visualization and Comparison. Our app includes the visualization of four different types of field-lines: streamline, pathline, streakline, and timeline. To distinguish different field-lines, we employ distinct colors: red for streamlines, orange for pathlines, green for streaklines, and purple for timelines. Streamlines are traced in one single time step (i.e., steady field), while all other field-lines are traced over multiple time steps (i.e., unsteady field). Figure 5.10 shows different field-lines depicted by our app.



(a) (b) (c) (d)

Figure 5.10: Field-lines. (a) streamlines. (b) pathlines. (c) pathline-streakline with LIC. (d) pathline-timeline with LIC. (© 2016 IS&T. Reprinted by permission.)

To demonstrate the concepts of field-lines and their relationships, we also provide multiple field-lines comparisons with animation. Timeline and streakline are both defined upon pathlines. Therefore, including pathline-timeline comparison and pathline-streakline comparison by showing the formation of timeline and streakline step by step would help users better understand the relationships between these flow lines.

Additionally, we use the LIC texture as the background to provide an overview of the flow within a plane in the flow field. The algorithm of generating LIC texture adds a random static pattern of black-and-white paint sources to visualize a flow field. As the flow passes by the sources each fluid particle picks up some of the source intensity. The result is a striped texture where points along the same streamline tends to have similar intensities. If the LIC texture is turned on when tracing pathline, streakline, or timeline, the texture will be updated synchronously over time showing the underlying unsteady flow field. Users can adjust the animation speed as desired. imgs/FlowVisual3D 5.10 (c) and (d) show pathline-streakline and pathline-timeline comparisons at selected time steps with LIC textures.

5.3.2 Stream Surfaces

A stream surface is a continuous surface that is everywhere tangent to the vector it passes in a steady flow field. It can be obtained by connecting the set of streamlines traced through every sample point on a seeding curve. In contrast to having numerous discrete streamlines in an area, a surface presents the flow pattern in a more coherent manner.

Seeding and Surface Construction. Selecting the seeding curve for a surface is crucial for surface generation. It influences the resulting surface in two aspects: the effectiveness of surface in characterizing flow features, and the smoothness of the surface.

We choose the seeding curves whose corresponding stream surfaces are able to capture the pattern of critical points. The types and locations of critical points reveal important patterns of a flow field, which are difficult to predict if no surfaces pass through those regions. Once the starting seed is placed, the following seeds on the curve are generated along the binormal vector of the previous seed so that the resulting stream surface can demonstrate the flow direction. For a point **p** on a streamline with velocity vector **v** and normal vector **n**, its binormal vector is the vector at **p** that is orthogonal to the plane containing **v** and **n**.

To generate smooth surfaces, we use a threshold value σ as the largest distance between two consecutive sample points to ensure the seeding curve is densely sampled. Then, we employ the easy integral surface algorithm [46] for surface construction where the maximum distance between two consecutive sample points on the propagation front is within σ. The front propagation is performed by tracing streamlines one step at a time and connecting the neighboring points into quads. Special cases such as divergence and convergence are taken care of to fill the gaps and avoid oversampling.

The surfaces are manually selected for the given five critical points data set. We first randomly generate 3000 lines that follow the binormal direction, and then manually specify segments of these lines as the seeding curves. To capture the flow features, we select those surfaces that pass through the critical regions. On the other hand, to avoid the surface being overly complicated, we do not select those surfaces that diverge and end at more than two critical points.

Surface Coloration. To show the correspondence between stream surfaces and critical points, we color stream surfaces that relate to the same critical points with similar colors (see Figure 5.11 (b) and (c)). As mentioned before, a single surface

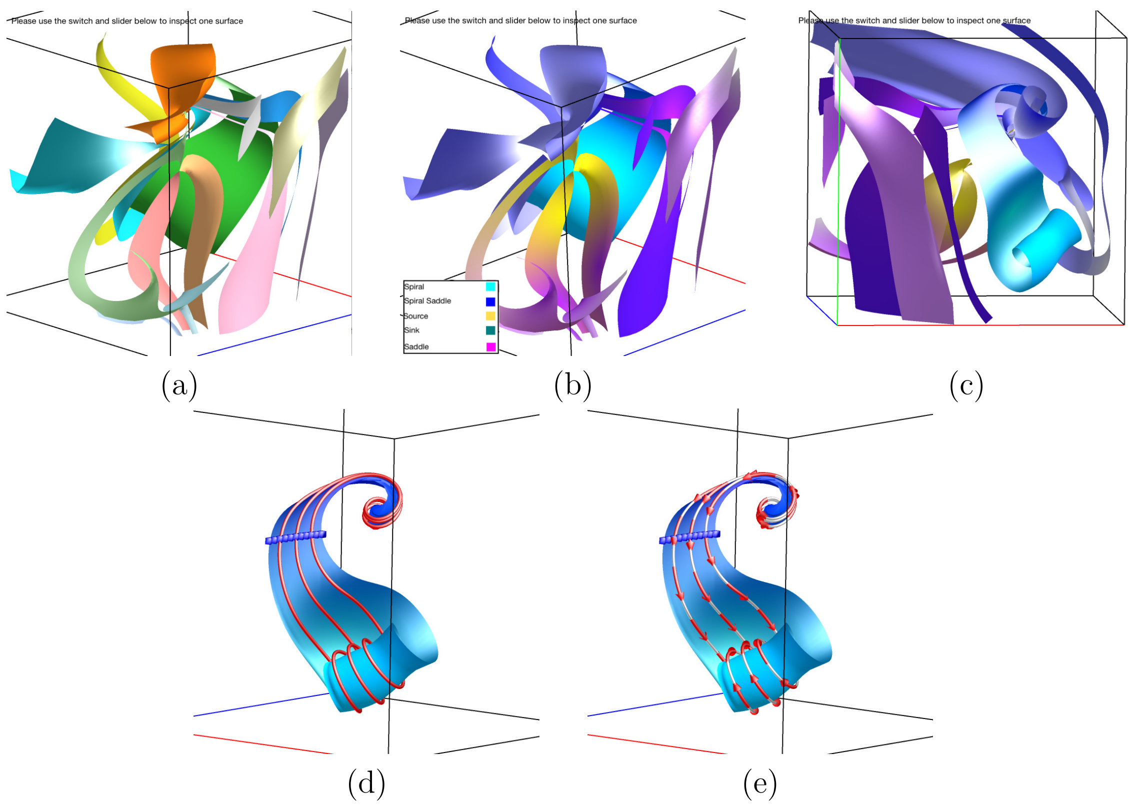

Please use the switch and slider below to inspect one surface
Please use the switch and slider below to inspect one surface
Spiral
Spiral Saddle
Source
Sink
Saddle
Please use the switch and slider below to inspect one surface
(a)
(b)
(c)
(d)
(e)

Figure 5.11: Stream Surfaces. (a) multiple surface overview with unique color for each surface. (b)-(c) multiple surface overview with coloring based on the types of related critical points. (d) single surface with streamlines. (e) single surface with streamline animation. (© 2016 IS&T. Reprinted by permission.)

passes no more than two critical points. Given zero velocity at critical points, the tracing of a surface either terminates at a critical point or on the boundaries of the flow field. This indicates that each of our surfaces connects two critical points or is constructed between a critical point and the volume's boundary. In the former case, the color of a surface vertex is linearly interpolated between the colors of the two critical points. In the later case, the color will gradually fade out as the surface moves far away from the critical point.

Surface Drawing. We precompute a total of fourteen surfaces from the five critical points data set and store them as files to reduce runtime workload and ensure prompt response during interaction. There are two options to examine the surfaces: multiple surface overview, and single surface inspection with streamlines and streamline animation. The first option displays all 14 surfaces to present an overview of the flow field. The second option enables closer inspection of a particular surface with streamlines or streamline animation.

Streamline Drawing. In the single surface inspection mode, we provide two options: streamlines and streamline animation, to help detailed inspection. An example is shown in Figure 5.11 (d) and (e). The displayed surface will have three streamlines evenly distributed on the surface to show the exact pattern of the flow. Streamline animation is also available for showing the speed and direction of the flow.

5.3.3 Critical Points

Extracting features from flow fields has been a topic of active research for decades. A great deal of work has been done to tackle this problem. Given a flow field, we achieve the following two goals in this work: figuring out locations and types of critical points for a given flow field, and designing seeding templates that effectively capture streamline patterns around different types of critical points.

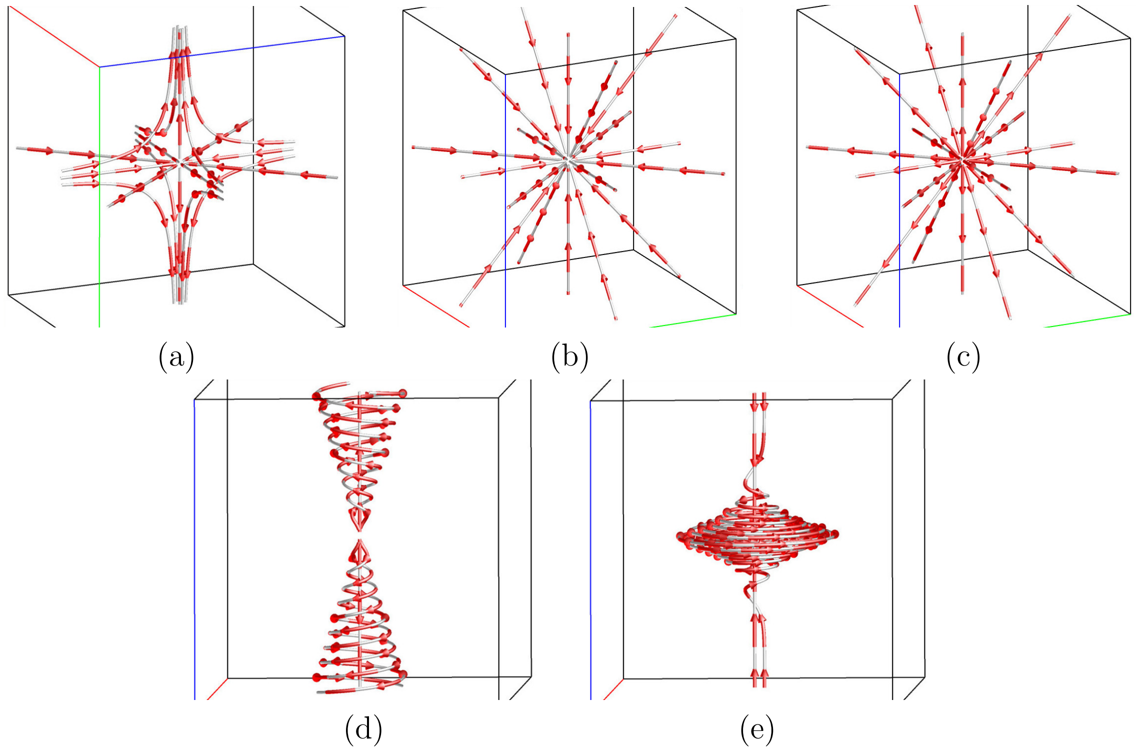


Figure 5.12: Critical points and their seeding templates. (a) saddle. (b) sink. (c) source. (d) spiral. and (e) spiral saddle. (© 2016 IS&T. Reprinted by permission.)

Critical Point Detection. A critical point is a position in a flow field domain where the velocity vanishes. For discrete vector data, we detect critical points through sign checking and vector interpolation. Specifically, for each voxel, we check whether there is at least one change of sign of the vectors at its corners. If the x, y and z vector components all have a sign change, it means that a critical point may exist within the voxel and a further step is taken to obtain the precise location of the critical point.

For a 3D flow field, the detection of critical points is achieved by utilizing the Greene's bisection method [33]. This method divides the flow field into equally-sized cubes and computes their Poincaré index to check the existence of any critical point within the cube. If a cube has a non-zero Poincaré index, the cube will be bisected into subcubes iteratively to find the precise locations of the critical points.

Critical Point Classification. Critical points are classified according to the flow patterns in their neighborhood. Mathematically, the type of critical point is determined by the real and imaginary parts of the eigenvalues of the Jacobian matrix in the neighborhood of the critical point. Since imaginary parts demonstrate the circulating flow pattern while real parts represent the repelling or attracting behavior of the flow, an analysis on both parts would determine the types of critical points [35, 79].

A first-order critical point $\mathbf{p}_0$ where $\mathbf{v}(\mathbf{p}_0 : t) = 0$ can be classified based on its eigenvalues of the Jacobian matrix $\mathbf{J}_\mathbf{v}(\mathbf{p}_0)$ when $det(\mathbf{J}_\mathbf{v}(\mathbf{p}_0)) \neq 0$.

Consider a 3D flow field

$$\mathbf{v}(x,y,z) = \begin{pmatrix} u(x,y,z) \\ v(x,y,z) \\ w(x,y,z) \end{pmatrix},$$

we have its Jacobian matrix as follows

$$\mathbf{J}_\mathbf{v}(x,y,z) = \begin{pmatrix} \frac{\partial u(x,y,z)}{\partial x} & \frac{\partial u(x,y,z)}{\partial y} & \frac{\partial u(x,y,z)}{\partial z} \\ \frac{\partial v(x,y,z)}{\partial x} & \frac{\partial v(x,y,z)}{\partial y} & \frac{\partial v(x,y,z)}{\partial z} \\ \frac{\partial w(x,y,z)}{\partial x} & \frac{\partial w(x,y,z)}{\partial y} & \frac{\partial w(x,y,z)}{\partial z} \end{pmatrix}.$$

Let λ_1, λ_2, λ_3 be the eigenvalues of $\mathbf{J}_\mathbf{v}(\mathbf{p}_0)$. R_1, R_2, R_3 are their real parts, and I_1, I_2, I_3 are their imaginary parts. We order λ_1, λ_2, λ_3 according to the values of their real parts so that $R_1 \leq R_2 \leq R_3$. Based on the sign of their real parts and the presence of imaginary parts, critical points can be grouped into the following types:

`Repelling node:`	$R_{1,2,3} > 0$	$I_{1,2,3} = 0$
`Attracting node:`	$R_{1,2,3} < 0$	$I_{1,2,3} = 0$
`Repelling focus:`	$R_{1,2,3} > 0$	$I_1 = 0, I_{2,3} \neq 0$
`Attracting focus:`	$R_{1,2,3} < 0$	$I_1 = 0, I_{2,3} \neq 0$
`Repelling node saddle:`	$R_1 < 0 < R_2 \leq R_3$	$I_{1,2,3} = 0$
`Attracting node saddle:`	$R_1 \leq R_2 < 0 < R_3$	$I_{1,2,3} = 0$
`Repelling focus saddle:`	$R_1 < 0 < R_2 \leq R_3$	$I_1 = 0, I_{2,3} \neq 0$
`Attracting focus saddle:`	$R_1 < 0 < R_2 \leq R_3$	$I_1 = 0, I_{2,3} \neq 0$
`Center:`	$R_{1,2,3} = 0$	$I_1 = 0, I_{2,3} \neq 0$

For simplicity, we merge all types of critical points into five types in our app as shown in Figure 5.12. Below is our merging strategy:

`Source:`	`Repelling node`	
`Sink:`	`Attracting node`	
`Spiral:`	`Repelling focus`	`Attracting focus`
`Spiral saddle:`	`Repelling node saddle`	`Attracting node saddle`
	`Repelling focus saddle`	`Attracting focus saddle`

This strategy preserves the distinction of sink and source. It also categorizes the types whose patterns present a combination of spiral and saddle shapes. In this way, we avoid using too many different colors to differentiate different types of critical points on the screen. As a result, users will not have to constantly refer back and forth to check the color and the type of a critical point.

Seeding Template. Since streamline patterns around distinct types of critical points are quite different from one another, streamline placement becomes important in order to effectively reveal the characteristics of critical points. We adopt a similar strategy proposed by Ye et al. [83] that applies a different seeding template for each type of critical point. For a saddle, the template has the seeds distributed along the bisector of the hyperbola. But for an attracting/repelling focus or a center, the pattern forms a circular or spiral shape. Therefore, seeding on concentric circles with increasing radii is an appropriate strategy. As for an attracting/repelling node, the flow directions are either toward or away from the critical point. The corresponding strategy is to place seeds evenly on a circle centered at the critical point.

Critical Point Visualization. Our app allows users to turn on/off a type of critical point. Critical points are drawn in the color of their types as indicated on the interface. By clicking on a critical point, its template seeds are placed to trace the corresponding streamlines so that the flow field around the critical point could be perceived. By applying a different template for each type of critical point, we avoid the overwhelming occlusion brought by drawing all streamlines around critical points while maintaining interactive performance. In addition, users may edit the templates through changing the size of the template, the number of layers along the z direction, and the distance between the layers. Figure 5.13 shows examples of visualizing critical points and their corresponding streamlines.

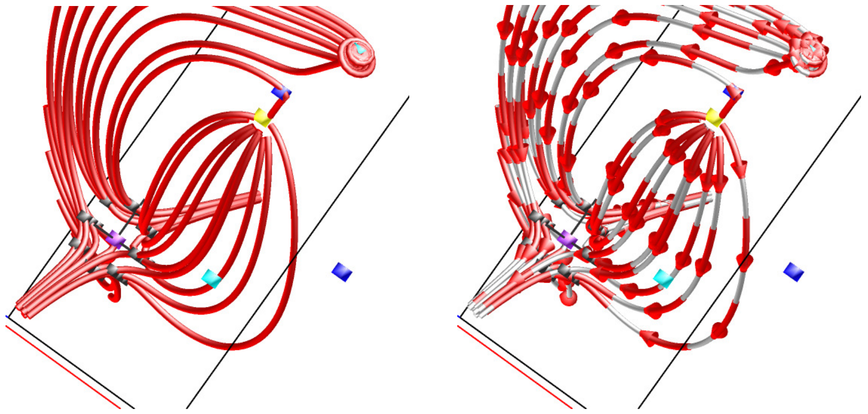

(a) (b)

Figure 5.13: Streamlines around critical points via template-based seeding. (a) static streamlines. (b) dynamic streamlines with animated arrows. (© 2016 IS&T. Reprinted by permission.)

Chapter 6

Results and Discussion

6.1 Conclusion

Access control, as an important aspect of Computer Security, has been taught and used in many institutions, organizations, and companies. It remains a challenging topic partly due to its abstract nature and partly because of the lack of practice environment during the study of the topic. We developed a set of tools that cover the widely-used access control models to help the learning and management. The pedagogical tools are model specific but under a unified design framework. Each tool consists of the same four components: a specification language, a visualization system, a query system, and a quiz system. The specification languages use similar syntax that shows the key components of individual models, but still keeps the specification in the simplest form. The visualization system provides interactive feedback to user's exploration. The query and quiz subsystems then allow users to obtain answers to frequently asked questions and provides a platform to conduct self-quiz on the materials. Instructors can also use the tools for teaching and quizzes, and provide example policies that allow students to interact with real examples before, during and after classes. Once a user is familiar to one of the tools, she may find the other tools easy to pick up. In addition to the pedagogical tools, we also developed a system ACvisual aiming to be a useful tool for both new learners and professional administrators in policy writing and analysis. It provides another more generic language that can be easily mapped from the common access requirement statements. The ratifier and the analyzer components together provide both graphical and textual analysis to policies. Selective as well as comprehensive testing of accesses are supported for quick check of the policies before deployment. All these tools had been evaluated by students at Michigan Tech and received positive feedback on the coverage of functionality and effectiveness in helping learning the materials. Most of the suggested improvements were on the user interface about the font size and use of color. The suggestions from participants were considered and had been incorporated.

For the flow field tools, we have developed PC and portable versions for 2D and 3D flow fields, respectively. The tools provide illustrations of streamline, pathline, streakline, and timeline. They also support critical point detection and classification to help to find the key features that define a flow field. Due to the computation power and storage limitation of mobile devices, we down sampled the 3D flow field data and incorporated an efficient stream surface drawing algorithm that significantly reduced the number of control points. The tools were evaluated by students from different majors and received positive feedback on the coverage and visual depiction of the flow field concepts, the usability, and its enhancement to the courses.

Throughout the development of pedagogical tools, we gathered some experience that we hope could be helpful for researcher in this field, and we include them in this paragraph: 1) It is always a good idea to do a little survey before designing a tool. Teaching/Sitting in a relevant class and making use of past homework or exams can provide important insight on when and which aspect of the topic become challenging for students. In our experience of developing UNIXvisual, the Permission Calculator that shows the conversion between letter and octal notations was added at last. We did not have it until we found many students made mistakes in this aspect in homework. 2) Provide the functionality that grow from one simplest aspect to a comprehensive form. In order to illustrate how access decision is made to an object in UNIX permissions, UNIXvisual started with showing the decision process to an object without considering its ancestor directories. From this step, users can understand how user, group and other bits work. Then the counter part with the directory traversal become easy when illustrated in another view. 3) Use visualizations for dynamic illustration. Visualization is quite expressive in showing dynamics of changes or transitions. Our user study shows that students enjoyed the guided walk-through by visualization. 4) An addition of self-evaluation component is always good. Problems can never be found until we start to use the knowledge to solve problems. Many students found they did not understand the materials thoroughly until exams. This is because remembering concepts and understanding small examples are different from being able to use the knowledge in a flexible way. Hence, providing self-evaluation component can expose users to problems at a practical scale and deepen their understanding.

6.2 Future Work

Our pedagogical tools currently facilitate learning through importing/create a policy file, providing interactive visualization to user operations, and offering a query/quiz system for enhancement. Most of the operations are user initiated. Thus, one direction for future work is to add more guided instructions. For first-time starters, canned examples could be shown for the mechanism of the access control model as well as the workflow including policy creation, policy modification, and queries. For previous users, they can opt to skip or review the demonstration of canned examples, and can easily find tutorials when using new features. As for the policy authoring and analysis tool ACvisual, more access control model besides RBAC could be incorporated to broaden users' selection of implementation model. The design of ACvisual has left space to include discretionary access control models such as UNIX permissions.

The concept of group can be implemented using the role concept; group containment be built through role inheritance. The SpaceTree layout is also efficient in showing access along a path from the root to the object of interest. This could make the access query with directory traversal, the most challenging aspect of the model, to be straightforward. Furthermore, attribute-based access control (ABAC) could be used in the place of RBAC. ABAC is an evolved form of RBAC; it allows adding attributes to all elements in RBAC. With this addition, users, objects, access and environment could be defined in a way that is closer to realistic cases.

As for the flow filed tools, future work could be in two directions. One is to improve in depth. That is, more key flow field concepts could be added to cover broader topics. The other one is to improve at the current level but with more quantitative information. Mathematical formulas could be incorporated to show the process of how different field-lines are generated, helping build a quantitative sense of flow field math at the very beginning of the course.

References

[1] A. Arora. *A foundation of fault-tolerant computing*. PhD thesis, University of Texas at Austin, 1992.

[2] U. S. S. at Arms. Report on the investigation into improper access to the senate judiciary committees computer system. `http://judiciary.senate.gov/testimony.cfm?id=1085&witid=2514`.

[3] L. Badger, D. F. Sterne, D. L. Sherman, K. M. Walker, and S. A. Haghighat. A domain and type enforcement UNIX prototype. In *Proceedings of the 5th Conference on USENIX UNIX Security Symposium*, SSYM'95, pages 12–12, Berkeley, CA, USA, 1995. USENIX Association.

[4] L. Badger, D. F. Sterne, D. L. Sherman, K. M. Walker, and S. A. Haghighat. Practical domain and type enforcement for unix. In *Proceedings of the 1995 IEEE Symposium on Security and Privacy*, SP '95, Washington, DC, USA, 1995. IEEE Computer Society.

[5] C. I. Baker. *Visual Processing in the Primate Brain*, chapter 4. American Cancer Society, 2012.

[6] J. Barkley and A. Cincotta. Managing role/permission relationships using object access types. In *Proceedings of the 3rd ACM Workshop on Role-based Access Control*, RBAC '98, pages 73–80, New York, NY, USA, 1998. ACM.

[7] J. F. Barkley, A. V. Cincotta, D. F. Ferraiolo, S. Gavrilla, and D. R. Kuhn. Role based access control for the world wide web. In *Proceedings of the 20th National Information System Security Conference. NIST/NSA*, 1997.

[8] R. Barrett, E. Kandogan, P. P. Maglio, E. M. Haber, L. A. Takayama, and M. Prabaker. Field studies of computer system administrators: Analysis of system management tools and practices. In *Proceedings of the 2004 ACM Conference on Computer Supported Cooperative Work*, CSCW '04, pages 388–395, New York, NY, USA, 2004. ACM.

[9] D. E. Bell and L. J. LaPadula. Secure computer systems: Mathematical foundations. Technical Report MTR-2547, Vol. 1, MITRE Corp., Bedford, MA, 1973.

[10] R. Berger. *The Undecidability of the Domino Problem.* Memoirs ; No 1/66. American Mathematical Society, 1966.

[11] K. Beznosov, P. Inglesant, J. Lobo, R. Reeder, and M. E. Zurko. Usability meets access control: Challenges and research opportunities. In *Proceedings of the 14th*

ACM Symposium on Access Control Models and Technologies, SACMAT '09, pages 73–74, New York, NY, USA, 2009. ACM.

[12] K. J. Biba. Integrity considerations for secure computer systems. Technical report, MITRE Corp., 1977.

[13] S. J. Bigelow. Implement access control systems successfully in your organization. `http://searchitchannel.techtarget.com/feature/The-importance-of-access-control`.

[14] M. Bishop. *Computer Security: Art and Science*. Addison-Wesley Professional, Arlington Street, Boston, MA, USA, 1st edition, 2002.

[15] P. Bonamy. *Maia and Mandos: Tools for Integrity Protection on Arbitrary Files*. PhD thesis, Michigan Technological University, 2016.

[16] D. Botta, R. Werlinger, A. Gagné, K. Beznosov, L. Iverson, S. Fels, and B. Fisher. Towards understanding it security professionals and their tools. In *Proceedings of the 3rd Symposium on Usable Privacy and Security*, SOUPS '07, pages 100–111, New York, NY, USA, 2007. ACM.

[17] C. Brodie, C.-M. Karat, J. Karat, and J. Feng. Usable security and privacy: A case study of developing privacy management tools. In *Proceedings of the 2005 Symposium on Usable Privacy and Security*, SOUPS '05, pages 35–43, New York, NY, USA, 2005. ACM.

[18] S. Brostoff, M. A. Sasse, D. Chadwick, J. Cunningham, U. Mbanaso, and S. Otenko. 'R-What': Development of a role-based access control policy-writing tool for e-scientists. *Software Practice and Experience*, 35(9):835–856, 2005.

[19] B. Cabral and L. C. Leedom. Imaging vector fields using line integral convolution. In *Proceedings of the 20th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '93, pages 263–270, New York, NY, USA, 1993. ACM.

[20] X. Cao and L. Iverson. Intentional access management: Making access control usable for end-users. In *Proceedings of the Second Symposium on Usable Privacy and Security*, SOUPS '06, pages 20–31, New York, NY, USA, 2006. ACM.

[21] X. Cao and L. Iverson. Intentional access management: Making access control usable for end-users. In *Proceedings of the Second Symposium on Usable Privacy and Security*, SOUPS '06, pages 20–31, New York, NY, USA, 2006. ACM.

[22] J. R. Crandall, S. L. Gerhart, and J. G. Hogle. Driving Home the Buffer Overflow Problem: A Training Module for Programmers and Managers. In *Proceedings of National Colloquium for Information Systems Security Education*, 2002.

[23] N. Damianou, N. Dulay, E. Lupu, and M. Sloman. The ponder policy specification language. In *Proceedings of the International Workshop on Policies for Distributed Systems and Networks*, POLICY '01, pages 18–38, London, UK, UK, 2001. Springer-Verlag.

[24] G. Dantzig. *Linear programming and extensions.* Rand Corporation Research Study. Princeton University Press, Princeton, NJ, 1963.

[25] J. C. F. de Winter. Using the student's *t*-test with extremely small sample sizes. *Practical Assessment, Research & Evaluation*, 18(10):1–12, 2013.

[26] D. Ebeling and R. Santos. Public Key Infrastructure Visualization. *The Journal of Computing Sciences in Colleges*, 23(1):247–254, 2007.

[27] A. Fabret and A. Petit. On the undecidability of deadlock detection in families of nets. In E. Mayr and C. Puech, editors, *STACS 95*, volume 900 of *Lecture Notes in Computer Science*, pages 479–490. Springer Berlin Heidelberg, 1995.

[28] FBK-IRST, C. M. University, and U. of Trento. Nusmv: a new symbolic model checker. `http://nusmv.fbk.eu/`.

[29] P. Felber and V. K. Garg, editors. *Stabilization, Safety, and Security of Distributed Systems - 16th International Symposium, SSS 2014, Paderborn, Germany, September 28 - October 1, 2014. Proceedings*, volume 8756 of *Lecture Notes in Computer Science*. Springer, 2014.

[30] D. Ferraiolo and R. Kuhn. Role-based access control. In *Proceedings of the 15th NIST-NCSC National Computer Security Conference*, pages 554–563, 1992.

[31] D. F. Ferraiolo, R. Sandhu, S. Gavrila, D. R. Kuhn, and R. Chandramouli.

Proposed NIST Standard for Role-based Access Control. *ACM Transactions on Information and System Security*, 4(3):224–274, 2001.

[32] N. S. Good and A. Krekelberg. Usability and privacy: A study of KaZaA P2P file-sharing. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '03, pages 137–144, New York, NY, USA, 2003. ACM.

[33] J. M. Greene. Locating three-dimensional roots by a bisection method. *Computational Physics*, 98(5):194–198, 1992.

[34] S. Hallyn and P. Kearns. Tools to administer domain and type enforcement. In *Proceedings of the 15th USENIX Conference on System Administration*, LISA '01, pages 151–156, Berkeley, CA, USA, 2001. USENIX Association.

[35] J. Helman and L. Hesselink. Representation and display of vector field topology in fluid flow data sets. *IEEE Computer*, 22(8):27–36, 1989.

[36] J. L. Helman and L. Hesselink. Representation and display of vector field topology in fluid flow data sets. *Computer*, 22(8):27–36, Aug. 1989.

[37] L. Hu. *A firewall model of file system security*. PhD thesis, Michigan Technological University, 2014.

[38] J. Hwang, T. Xie, V. Hu, and M. Altunay. ACPT: A tool for modeling and verifying access control policies. In *Proceedings of Policies for Distributed Systems and Networks (POLICY)*, pages 40–43. IEEE, 2010.

[39] P. Inglesant, M. A. Sasse, D. Chadwick, and L. L. Shi. Expressions of expertness: The virtuous circle of natural language for access control policy specification. In *Proceedings of the 4th Symposium on Usable Privacy and Security*, SOUPS '08, pages 77–88, New York, NY, USA, 2008. ACM.

[40] C.-M. Karat, J. Karat, C. Brodie, and J. Feng. Evaluating interfaces for privacy policy rule authoring. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '06, pages 83–92, New York, NY, USA, 2006. ACM.

[41] M. Kunz, L. Fuchs, M. Netter, and G. Pernul. *How to Discover High-Quality Roles? A Survey and Dependency Analysis of Quality Criteria in Role Mining*, pages 49–67. Springer International Publishing, Cham, 2015.

[42] J. Lamping, R. Rao, and P. Pirolli. A focus+context technique based on hyperbolic geometry for visualizing large hierarchies. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, pages 401–408, New York, NY, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[43] B. W. Lampson. Protection. *SIGOPS Oper. Syst. Rev.*, 8(1):18–24, Jan. 1974.

[44] Y. Li, S. Carr, J. Mayo, C.-K. Shene, and C. Wang. DTEvisual: A visualization system for teaching access control using domain type enforcement. *Journal of Computing Sciences in Colleges*, 28(1):125–132, Oct. 2012.

[45] R. A. Maxion and R. W. Reeder. Improving user-interface dependability through mitigation of human error. *International Journal of Human-Computer Studies*, 63(1-2):25–50, July 2005.

[46] T. McLoughlin, R. S. Laramee, and E. Zhang. Easy integral surfaces: A fast, quad-based stream and path surface algorithm. In *Proceedings of Computer Graphics International*, pages 73–82, 2009.

[47] N. I. of Standards and Technology. Combinatorial and pairwise testing. `http://csrc.nist.gov/groups/sns/acts/`.

[48] N. I. of Standards and Technology. Eiciel:gnome file acl editor. `https://rofi.roger-ferrer.org/eiciel/`.

[49] C. on National Security Systems. *National Information Assurance (IA) Glossary*. Committee on National Security Systems, 1996.

[50] C. Plaisant, J. Grosjean, and B. B. Bederson. SpaceTree: Supporting exploration in large node link tree, design evolution and empirical evaluation. In P. C. Wong and K. Andrews, editors, *INFOVIS*, pages 57–64. IEEE Computer Society, 2002.

[51] A. C. PRQC. *ATIS Telecom Glossary*. ATIS Committee PRQC, 2012.

[52] R. W. Reeder, L. Bauer, L. F. Cranor, M. K. Reiter, K. Bacon, K. How, and H. Strong. Expandable grids for visualizing and authoring computer security

policies. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '08, pages 1473–1482, New York, NY, USA, 2008. ACM.

[53] R. W. Reeder, L. Bauer, L. F. Cranor, M. K. Reiter, and K. Vaniea. More than skin deep: Measuring effects of the underlying model on access-control system usability. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '11, pages 2065–2074, New York, NY, USA, 2011. ACM.

[54] J. Rode, C. Johansson, P. DiGioia, R. S. Filho, K. Nies, D. H. Nguyen, J. Ren, P. Dourish, and D. Redmiles. Seeing further: Extending visualization as a basis for usable security. In *Proceedings of the Second Symposium on Usable Privacy and Security*, SOUPS '06, pages 145–155, New York, NY, USA, 2006. ACM.

[55] J. Rode, C. Johansson, P. DiGioia, R. S. Filho, K. Nies, D. H. Nguyen, J. Ren, P. Dourish, and D. Redmiles. Seeing further: Extending visualization as a basis for usable security. In *Proceedings of the Second Symposium on Usable Privacy and Security*, SOUPS '06, pages 145–155, New York, NY, USA, 2006. ACM.

[56] R. Sandhu. Roles versus groups. In *Proceedings of the First ACM Workshop on Role-based Access Control*, RBAC '95, New York, NY, USA, 1996. ACM.

[57] R. S. Sandhu, E. J. Coyne, H. L. Feinstein, and C. E. Youman. Role-based access control models. *Computer*, 29(2):38–47, Feb. 1996.

[58] D. Schweitzer and L. Baird. The design and use of interactive visualization applets for teaching ciphers. In *Proceedings of IEEE Workshop on Information Assurance*, pages 69–75, 2006.

[59] D. Schweitzer, L. Baird, M. Collins, W. Brown, and M. Sherman. Grasp: A visualization tool for teaching security protocols. In *Proceedings of National Colloquium for Information Systems Security Education*, pages 75–81, 2006.

[60] D. Schweitzer and W. Brown. Using Visualization To Teach Security. *The Journal of Computing Sciences in Colleges*, 24(5):143–150, 2009.

[61] D. Schweitzer, M. Collins, and L. Baird. A Visual Approach To Teaching Formal Access Models In Security. In *Proceedings of National Colloquium for Information Systems Security Education*, pages 69–75, 2007.

[62] D. Schweitzer, M. Collins, L. Baird, U. States, and A. F. Academy. A visual approach to teaching formal access models in security, 2007.

[63] S. Smalley. Configuring the SELinux Policy. Technical report, NSA/NAI, Jan. 2003.

[64] P. Software. Permission analyzer. `http://www.permissionanalyzer.com/?gclid=EAIaIQobChMIhc-J3NzQ1gIVDJFpCh16gQaJEAAYAyAAEgITJPD_BwE`.

[65] R. W. Software. Klocwork. `https://www.roguewave.com/products-services/klocwork`. Accessed: 2019-02-14.

[66] W. Stallings and L. Brown. *Computer Security: Principles and Practice*. Prentice Hall Press, Upper Saddle River, NJ, USA, 3rd edition, 2014.

[67] D. Sternadt Alexandre and J. Tavares. Introduction of human perception in visualization. *International Journal of Imaging*, 4, 01 2010.

[68] J. Tao, J. Ma, M. Keranen, J. Mayo, and C.-K. Shene. ECvisual: A Visualization Tool for Elliptic Curve Based Ciphers. In *Proceedings of ACM Technical Symposium on Computer Science Education*, pages 571–576, 2012.

[69] R. L. Trey Guerin. How role-based access control can provide security and business benefits. `http://www.computerworld.com/article/2573892/security0/how-role-based-access-control-can-provide-security-and-business-benefits.html`.

[70] V. Verma, D. Kao, and A. Pang. A flow-guided streamline seeding strategy. In *Proceedings of the Conference on Visualization '00*, VIS '00, pages 163–170, Los Alamitos, CA, USA, 2000. IEEE Computer Society Press.

[71] V. Verma, D. Kao, and A. Pang. A flow-guided streamline seeding strategy. In *Proceedings of IEEE Visualization Conference*, pages 163–170, 2000.

[72] H. Wang. Proving theorems by pattern recognition II. *Bell System Technical Journal*, 40:1–42, 1961.

[73] M. Wang, S. Carr, J. Mayo, C.-K. Shene, and C. Wang. MLSvisual: A visualization tool for teaching access control using multi-level security. In *Proceedings of the 2014 Conference on Innovation and Technology in Computer Science Education*, ITiCSE '14, pages 93–98, New York, NY, USA, 2014. ACM.

[74] M. Wang, J. Mayo, C.-K. Shene, S. Carr, and C. Wang. UNIXvisual: A visualization tool for teaching the unix permission model. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE '16, pages 356–356, New York, NY, USA, 2016. ACM.

[75] M. Wang, J. Mayo, C.-K. Shene, S. Carr, and C. Wang. UNIXvisual: A visualization tool for teaching unix permissions. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE '17, pages 194–199, New York, NY, USA, 2017. ACM.

[76] M. Wang, J. Mayo, C.-K. Shene, T. Lake, S. Carr, and C. Wang. RBACvisual: A visualization tool for teaching access control using role-based access control. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education*, ITiCSE '15, pages 141–146, New York, NY, USA, 2015. ACM.

[77] M. Wang, J. Tao, J. Ma, Y. Shen, and C. Wang. FlowVisual: A visualization app for teaching and understanding 3D flow field concepts. In *Visualization*

and Data Analysis 2016, San Francisco, California, USA, February 14-18, 2016, pages 1–10, 2016.

[78] M. Wang, J. Tao, C. Wang, C.-K. Shene, and S. H. Kim. FlowVisual: Design and evaluation of a visualization tool for teaching 2D flow field concepts. In *Proceedings of American Society for Engineering Education Annual Conference*, 2013.

[79] T. Weinkauf, H. Theisel, H.-C. Hege, and H.-P. Seidel. Boundary switch connectors for topological visualization of complex 3D vector fields. In *Proceedings of Joint Eurographics - IEEE TCVG Symposium on Visualization*, pages 183–192, 2004.

[80] T. Whalen, D. Smetters, and E. F. Churchill. User experiences with sharing and access control. In *CHI '06 Extended Abstracts on Human Factors in Computing Systems*, CHI EA '06, pages 1517–1522, New York, NY, USA, 2006. ACM.

[81] A. Whitten and J. D. Tygar. Why Johnny Can'T Encrypt: A usability evaluation of pgp 5.0. In *Proceedings of the 8th Conference on USENIX Security Symposium - Volume 8*, SSYM'99, pages 14–14, Berkeley, CA, USA, 1999. USENIX Association.

[82] Y. Wu, W. Shi, H. Liang, Q. Shang, C. Yuan, and L. Bin. Security on-demand

architecture with multiple modules support. In *Proceedings of the First International Conference on Information Security Practice and Experience*, ISPEC'05, pages 121–131, Berlin, Heidelberg, 2005. Springer-Verlag.

[83] X. Ye, D. Kao, and A. Pang. Strategy for seeding 3D streamlines. In *Proceedings of IEEE Visualization Conference*, pages 471–478, 2005.

[84] W. Zimmermann and S. Cunningham, editors. *Visualization in teaching and learning mathematics*. Mathematical Association of America, Washington, DC, USA, 1991.

[85] M. E. Zurko, R. Simon, and T. Sanfilippo. A user-centered, modular authorization service built on an rbac foundation. *2012 IEEE Symposium on Security and Privacy*, 00(undefined):0057, 1999.

[86] M. E. Zurko and R. T. Simon. User-centered security. In *Proceedings of the 1996 Workshop on New Security Paradigms*, NSPW '96, pages 27–33, New York, NY, USA, 1996. ACM.

[87] M. E. Zurko, R. T. Simon, and T. Sanfilippo. A user-centered, modular authorization service built on an rbac foundation. In *IEEE Symposium on Security and Privacy*, pages 57–71. IEEE Computer Society, 1999.

Appendix A

Example SMV File

```
MODULE main
VAR
  USERS: {dummy, oscar, sally, sam, tina, alan};
  ROLES: {dummy, role_tina, role_sally, role_sam, role0, role_oscar};
  OBJECTS: {dummy, _classes_os, _home, _, _classes_security, _tools, ↩
      _classes_os_public, _classes_security_public};
  OPERATIONS: {dummy, read, write, execute};
  RECURSIVE: {true, false};
  RBAC_RolePerms: RBAC_RolePerms( ROLES, OBJECTS, OPERATIONS, RECURSIVE);
  RBAC_UserPerms: RBAC_UserPerms( USERS, OBJECTS, OPERATIONS, RECURSIVE);
ASSIGN
```

```
  next(USERS)  := USERS;

  next(ROLES)  := ROLES;

  next(OBJECTS) := OBJECTS;

  next(OPERATIONS):= OPERATIONS;

  next(RECURSIVE):= RECURSIVE;


MODULE RBAC_RolePerms( ROLES, OBJECTS, OPERATIONS, RECURSIVE)

VAR

  decision: {Permit, Deny};

ASSIGN

  init(decision) := Deny;

  next(decision) := case

  ROLES = role_tina & OBJECTS = _classes_security & OPERATIONS = execute ↩

      & RECURSIVE = true : Permit;

  ROLES = role_tina & OBJECTS = _classes_security & OPERATIONS = read & ↩

      RECURSIVE = true : Permit;

  ROLES = role_tina & OBJECTS = _classes_security & OPERATIONS = write & ↩

      RECURSIVE = true : Permit;

  ROLES = role_tina & OBJECTS = _classes_os & OPERATIONS = execute & ↩

      RECURSIVE = true : Permit;

  ROLES = role_tina & OBJECTS = _classes_os & OPERATIONS = read & ↩

      RECURSIVE = true : Permit;
```

```
ROLES = role_tina & OBJECTS = _classes_os & OPERATIONS = write & ↩
   RECURSIVE = true : Permit;
ROLES = role_tina & OBJECTS = _classes_security_public & OPERATIONS = ↩
   write & RECURSIVE = true : Permit;
ROLES = role_tina & OBJECTS = _classes_security_public & OPERATIONS = ↩
   read & RECURSIVE = true : Permit;
ROLES = role_tina & OBJECTS = _tools & OPERATIONS = execute & RECURSIVE↩
    = true : Permit;
ROLES = role_tina & OBJECTS = _tools & OPERATIONS = read & RECURSIVE = ↩
   true : Permit;
ROLES = role_tina & OBJECTS = _classes_os_public & OPERATIONS = read & ↩
   RECURSIVE = true : Permit;
ROLES = role_sally & OBJECTS = _classes_security_public & OPERATIONS = ↩
   write & RECURSIVE = true : Permit;
ROLES = role_sally & OBJECTS = _classes_security_public & OPERATIONS = ↩
   read & RECURSIVE = true : Permit;
ROLES = role_sally & OBJECTS = _tools & OPERATIONS = execute & ↩
   RECURSIVE = true : Permit;
ROLES = role_sally & OBJECTS = _tools & OPERATIONS = read & RECURSIVE =↩
    true : Permit;
ROLES = role_sam & OBJECTS = _classes_security_public & OPERATIONS = ↩
   read & RECURSIVE = true : Permit;
```

```
ROLES = role_sam & OBJECTS = _tools & OPERATIONS = execute & RECURSIVE ↩
    = true : Permit;
ROLES = role_sam & OBJECTS = _tools & OPERATIONS = read & RECURSIVE = ↩
    true : Permit;
ROLES = role0 & OBJECTS = _ & OPERATIONS = execute & RECURSIVE = true :↩
     Permit;
ROLES = role0 & OBJECTS = _ & OPERATIONS = read & RECURSIVE = true : ↩
    Permit;
ROLES = role0 & OBJECTS = _ & OPERATIONS = write & RECURSIVE = true : ↩
    Permit;
ROLES = role0 & OBJECTS = _classes_security & OPERATIONS = execute & ↩
    RECURSIVE = true : Permit;
ROLES = role0 & OBJECTS = _classes_security & OPERATIONS = read & ↩
    RECURSIVE = true : Permit;
ROLES = role0 & OBJECTS = _classes_security & OPERATIONS = write & ↩
    RECURSIVE = true : Permit;
ROLES = role0 & OBJECTS = _classes_os & OPERATIONS = execute & ↩
    RECURSIVE = true : Permit;
ROLES = role0 & OBJECTS = _classes_os & OPERATIONS = read & RECURSIVE =↩
     true : Permit;
ROLES = role0 & OBJECTS = _classes_os & OPERATIONS = write & RECURSIVE ↩
    = true : Permit;
```

```
  ROLES = role0 & OBJECTS = _classes_security_public & OPERATIONS = write↩
      & RECURSIVE = true : Permit;
  ROLES = role0 & OBJECTS = _classes_security_public & OPERATIONS = read ↩
     & RECURSIVE = true : Permit;
  ROLES = role0 & OBJECTS = _tools & OPERATIONS = execute & RECURSIVE = ↩
     true : Permit;
  ROLES = role0 & OBJECTS = _tools & OPERATIONS = read & RECURSIVE = true↩
      : Permit;
  ROLES = role0 & OBJECTS = _classes_os_public & OPERATIONS = read & ↩
     RECURSIVE = true : Permit;
  ROLES = role_oscar & OBJECTS = _tools & OPERATIONS = execute & ↩
     RECURSIVE = true : Permit;
  ROLES = role_oscar & OBJECTS = _tools & OPERATIONS = read & RECURSIVE =↩
      true : Permit;
  ROLES = role_oscar & OBJECTS = _classes_os_public & OPERATIONS = read &↩
      RECURSIVE = true : Permit;
  TRUE: Deny;
  esac;
-- Cyclic inheritance property
SPEC AG ((ROLES = role_sally) & (OBJECTS = _classes_security) & (↩
   OPERATIONS = execute) & (RECURSIVE = true) -> AF decision = Deny)
```

```
SPEC AG ((ROLES = role_oscar) & (OBJECTS = _classes_security) & (↩
    OPERATIONS = execute) & (RECURSIVE = true) -> AF decision = Deny)
SPEC AG ((ROLES = role_sam) & (OBJECTS = _classes_security_public) & (↩
    OPERATIONS = write) & (RECURSIVE = true) -> AF decision = Deny)
SPEC AG ((ROLES = role_tina) & (OBJECTS = _) & (OPERATIONS = execute) & (↩
    RECURSIVE = true) -> AF decision = Deny)
MODULE RBAC_UserPerms( USERS, OBJECTS, OPERATIONS, RECURSIVE)
VAR
  decision: {Permit, Deny};
ASSIGN
  init(decision) := Deny;
  next(decision) := case
  USERS = oscar & OBJECTS = _tools & OPERATIONS = execute : Permit;
  USERS = oscar & OBJECTS = _tools & OPERATIONS = read : Permit;
  USERS = oscar & OBJECTS = _classes_os_public & OPERATIONS = read : ↩
      Permit;
  USERS = sally & OBJECTS = _classes_security_public & OPERATIONS = write↩
       : Permit;
  USERS = sally & OBJECTS = _classes_security_public & OPERATIONS = read ↩
      : Permit;
  USERS = sally & OBJECTS = _tools & OPERATIONS = execute : Permit;
  USERS = sally & OBJECTS = _tools & OPERATIONS = read : Permit;
USERS = sam & OBJECTS = _classes_security_public & OPERATIONS = read : ↩
    Permit;
USERS = sam & OBJECTS = _tools & OPERATIONS = execute : Permit;
USERS = sam & OBJECTS = _tools & OPERATIONS = read : Permit;
USERS = tina & OBJECTS = _classes_security & OPERATIONS = execute : ↩
    Permit;
USERS = tina & OBJECTS = _classes_security & OPERATIONS = read : Permit↩
    ;
USERS = tina & OBJECTS = _classes_security & OPERATIONS = write : ↩
    Permit;
USERS = tina & OBJECTS = _classes_os & OPERATIONS = execute : Permit;
USERS = tina & OBJECTS = _classes_os & OPERATIONS = read : Permit;
USERS = tina & OBJECTS = _classes_os & OPERATIONS = write : Permit;
USERS = tina & OBJECTS = _classes_security_public & OPERATIONS = write ↩
    : Permit;
USERS = tina & OBJECTS = _classes_security_public & OPERATIONS = read :↩
     Permit;
USERS = tina & OBJECTS = _tools & OPERATIONS = execute : Permit;
USERS = tina & OBJECTS = _tools & OPERATIONS = read : Permit;
USERS = tina & OBJECTS = _classes_os_public & OPERATIONS = read : ↩
    Permit;
```

```
USERS = alan & OBJECTS = _classes_security & OPERATIONS = execute : ↩
   Permit;
USERS = alan & OBJECTS = _classes_security & OPERATIONS = read : Permit↩
   ;
USERS = alan & OBJECTS = _classes_security & OPERATIONS = write : ↩
   Permit;
USERS = alan & OBJECTS = _classes_os & OPERATIONS = execute : Permit;
USERS = alan & OBJECTS = _classes_os & OPERATIONS = read : Permit;
USERS = alan & OBJECTS = _classes_os & OPERATIONS = write : Permit;
USERS = alan & OBJECTS = _classes_security_public & OPERATIONS = write ↩
   : Permit;
USERS = alan & OBJECTS = _classes_security_public & OPERATIONS = read :↩
    Permit;
USERS = alan & OBJECTS = _tools & OPERATIONS = execute : Permit;
USERS = alan & OBJECTS = _tools & OPERATIONS = read : Permit;
USERS = alan & OBJECTS = _classes_os_public & OPERATIONS = read : ↩
   Permit;
USERS = dummy & OBJECTS = _tools & OPERATIONS = read : Permit;
USERS = dummy & OBJECTS = _tools & OPERATIONS = execute : Permit;
USERS = dummy & OBJECTS = _classes_security & OPERATIONS = read : ↩
   Permit;
```

```
USERS = dummy & OBJECTS = _classes_security & OPERATIONS = write : ↩
   Permit;
USERS = dummy & OBJECTS = _classes_security & OPERATIONS = execute : ↩
   Permit;
USERS = dummy & OBJECTS = _classes_os_public & OPERATIONS = read : ↩
   Permit;
USERS = dummy & OBJECTS = _classes_security_public & OPERATIONS = write↩
    : Permit;
USERS = dummy & OBJECTS = _classes_security_public & OPERATIONS = read ↩
   : Permit;
USERS = dummy & OBJECTS = _classes_os & OPERATIONS = read : Permit;
USERS = dummy & OBJECTS = _classes_os & OPERATIONS = write : Permit;
USERS = dummy & OBJECTS = _classes_os & OPERATIONS = execute : Permit;
USERS = oscar & OBJECTS = dummy & OPERATIONS = read : Permit;
USERS = oscar & OBJECTS = dummy & OPERATIONS = execute : Permit;
USERS = sally & OBJECTS = dummy & OPERATIONS = write : Permit;
USERS = sally & OBJECTS = dummy & OPERATIONS = read : Permit;
USERS = sally & OBJECTS = dummy & OPERATIONS = execute : Permit;
USERS = tina & OBJECTS = dummy & OPERATIONS = read : Permit;
USERS = tina & OBJECTS = dummy & OPERATIONS = write : Permit;
USERS = tina & OBJECTS = dummy & OPERATIONS = execute : Permit;
USERS = sam & OBJECTS = dummy & OPERATIONS = read : Permit;
```

```
  USERS = sam & OBJECTS = dummy & OPERATIONS = execute : Permit;
  USERS = alan & OBJECTS = dummy & OPERATIONS = read : Permit;
  USERS = alan & OBJECTS = dummy & OPERATIONS = write : Permit;
  USERS = alan & OBJECTS = dummy & OPERATIONS = execute : Permit;
  TRUE: Deny;
  esac;
-- User permission check
SPEC AG ((USERS = oscar) & (OBJECTS = _tools) & (OPERATIONS = execute) ->↩
     AF decision = Permit)
SPEC AG ((USERS = oscar) & (OBJECTS = _tools) & (OPERATIONS = read) -> AF↩
     decision = Permit)
SPEC AG ((USERS = oscar) & (OBJECTS = _classes_os_public) & (OPERATIONS =↩
     read) -> AF decision = Permit)
SPEC AG ((USERS = sally) & (OBJECTS = _classes_security_public) & (↩
    OPERATIONS = write) -> AF decision = Permit)
SPEC AG ((USERS = sally) & (OBJECTS = _classes_security_public) & (↩
    OPERATIONS = read) -> AF decision = Permit)
SPEC AG ((USERS = sally) & (OBJECTS = _tools) & (OPERATIONS = execute) ->↩
     AF decision = Permit)
SPEC AG ((USERS = sally) & (OBJECTS = _tools) & (OPERATIONS = read) -> AF↩
     decision = Permit)
```

```
SPEC AG ((USERS = sam) & (OBJECTS = _classes_security_public) & (↩
    OPERATIONS = read) -> AF decision = Permit)
SPEC AG ((USERS = sam) & (OBJECTS = _tools) & (OPERATIONS = execute) -> ↩
    AF decision = Permit)
SPEC AG ((USERS = sam) & (OBJECTS = _tools) & (OPERATIONS = read) -> AF ↩
    decision = Permit)
SPEC AG ((USERS = tina) & (OBJECTS = _classes_security) & (OPERATIONS = ↩
    execute) -> AF decision = Permit)
SPEC AG ((USERS = tina) & (OBJECTS = _classes_security) & (OPERATIONS = ↩
    read) -> AF decision = Permit)
SPEC AG ((USERS = tina) & (OBJECTS = _classes_security) & (OPERATIONS = ↩
    write) -> AF decision = Permit)
SPEC AG ((USERS = tina) & (OBJECTS = _classes_os) & (OPERATIONS = execute↩
    ) -> AF decision = Permit)
SPEC AG ((USERS = tina) & (OBJECTS = _classes_os) & (OPERATIONS = read) ↩
    -> AF decision = Permit)
SPEC AG ((USERS = tina) & (OBJECTS = _classes_os) & (OPERATIONS = write) ↩
    -> AF decision = Permit)
SPEC AG ((USERS = tina) & (OBJECTS = _classes_security_public) & (↩
    OPERATIONS = write) -> AF decision = Permit)
SPEC AG ((USERS = tina) & (OBJECTS = _classes_security_public) & (↩
    OPERATIONS = read) -> AF decision = Permit)
```

```
SPEC AG ((USERS = tina) & (OBJECTS = _tools) & (OPERATIONS = execute) -> ↩
    AF decision = Permit)
SPEC AG ((USERS = tina) & (OBJECTS = _tools) & (OPERATIONS = read) -> AF ↩
    decision = Permit)
SPEC AG ((USERS = tina) & (OBJECTS = _classes_os_public) & (OPERATIONS = ↩
    read) -> AF decision = Permit)
SPEC AG ((USERS = alan) & (OBJECTS = _classes_security) & (OPERATIONS = ↩
    execute) -> AF decision = Permit)
SPEC AG ((USERS = alan) & (OBJECTS = _classes_security) & (OPERATIONS = ↩
    read) -> AF decision = Permit)
SPEC AG ((USERS = alan) & (OBJECTS = _classes_security) & (OPERATIONS = ↩
    write) -> AF decision = Permit)
SPEC AG ((USERS = alan) & (OBJECTS = _classes_os) & (OPERATIONS = execute↩
    ) -> AF decision = Permit)
SPEC AG ((USERS = alan) & (OBJECTS = _classes_os) & (OPERATIONS = read) ↩
    -> AF decision = Permit)
SPEC AG ((USERS = alan) & (OBJECTS = _classes_os) & (OPERATIONS = write) ↩
    -> AF decision = Permit)
SPEC AG ((USERS = alan) & (OBJECTS = _classes_security_public) & (↩
    OPERATIONS = write) -> AF decision = Permit)
SPEC AG ((USERS = alan) & (OBJECTS = _classes_security_public) & (↩
    OPERATIONS = read) -> AF decision = Permit)
```

```
SPEC AG ((USERS = alan) & (OBJECTS = _tools) & (OPERATIONS = execute) -> ↩
    AF decision = Permit)
SPEC AG ((USERS = alan) & (OBJECTS = _tools) & (OPERATIONS = read) -> AF ↩
    decision = Permit)
SPEC AG ((USERS = alan) & (OBJECTS = _classes_os_public) & (OPERATIONS = ↩
    read) -> AF decision = Permit)
```

Appendix B

RBAC Technical Questions

Answer the following questions about the RBAC system defined on the last page.

You should not use ACvisual in this part.

1. Indicate all the roles that *Ringo* can occupy.

   ☐ *Programmer*

   ☐ *Antivirus Programmer*

   ☐ *Firewall Programmer*

   ☐ *Tester*

   ☐ None

2. List the type of access (read,write,exec) that *Ringo* can acquire for each of the following objects. If he cannot access an object, write `no access`.

    (a) A: ____________________

    (b) B: ____________________

    (c) C: ____________________

    (d) D: ____________________

    (e) E: ____________________

3. Indicate all the roles that inherit the *Programmer* role. (There may be more than one.)

    ☐ *Programmer*

    ☐ *Antivirus Programmer*

    ☐ *Firewall Programmer*

    ☐ *Tester*

    ☐ None

4. Indicate all the roles inherited by the *Tester* role. (There may be more than one.)

   ☐ *Programmer*

   ☐ *Antivirus Programmer*

   ☐ *Firewall Programmer*

   ☐ *Tester*

   ☐ None

RBAC EXAMPLE

A site plans to use RBAC (under the NIST model) to implement its security requirements. The site has identified the following set of users U and roles R.

$U = \{John, Paul, Ringo, George\}$

$R = \{Programmer, AntivirusProgrammer, FirewallProgrammer, Tester\}$

The permissions assignment PA is given by the following access control matrix.

	OBJECTS				
ROLE	A	B	C	D	E
Programmer	read, exec				
Antivirus Programmer	read, exec	read, write, exec			
Firewall Programmer	read		read, exec		
Tester	read, exec	read, exec	read, exec		read, write

The role assignment RA is given by the following access control matrix.

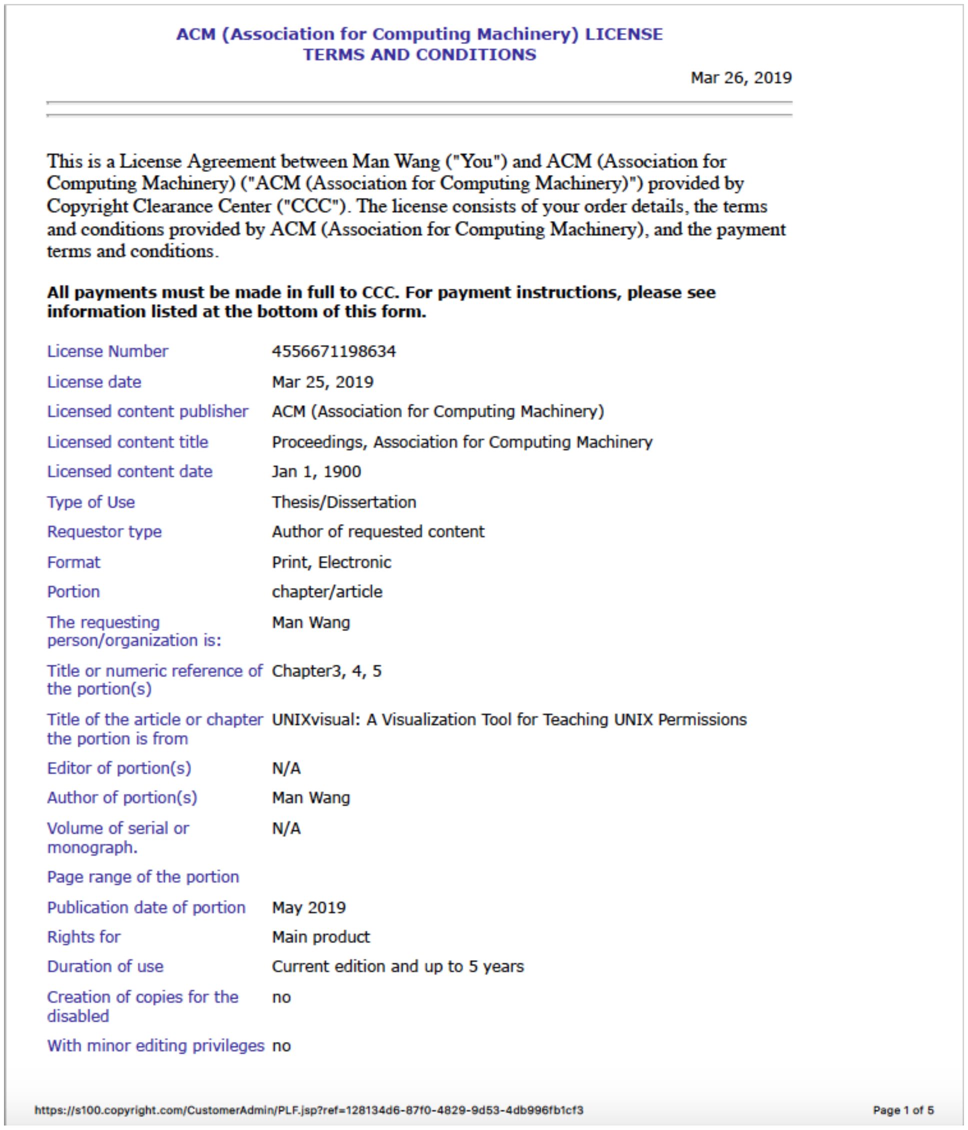
	ROLE			
USER	Programmer	Antivirus Programmer	Firewall Programmer	Tester
John	X			
Paul		X		
Ringo	X		X	
George	X		X	X

Appendix C

Letters of Permission

This dissertation reuses materials from five of my previous publications. At the beginning of each chapter, in which the published materials are reused, a footnote is presented to indicate the reused materials and cite the original paper(s). The copyright of respective publisher is also indicated in the caption of each figure with reusing materials. The permissions to reuse the published materials are listed as follows:

[73] M. Wang, J. Mayo, C.-K. Shene, S. Carr, and C. Wang. UNIXvisual: A visualization tool for teaching the UNIX permission model. In Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE '16, pages 356–356, New York, NY, USA, 2016. ACM.



ACM (Association for Computing Machinery) LICENSE
TERMS AND CONDITIONS

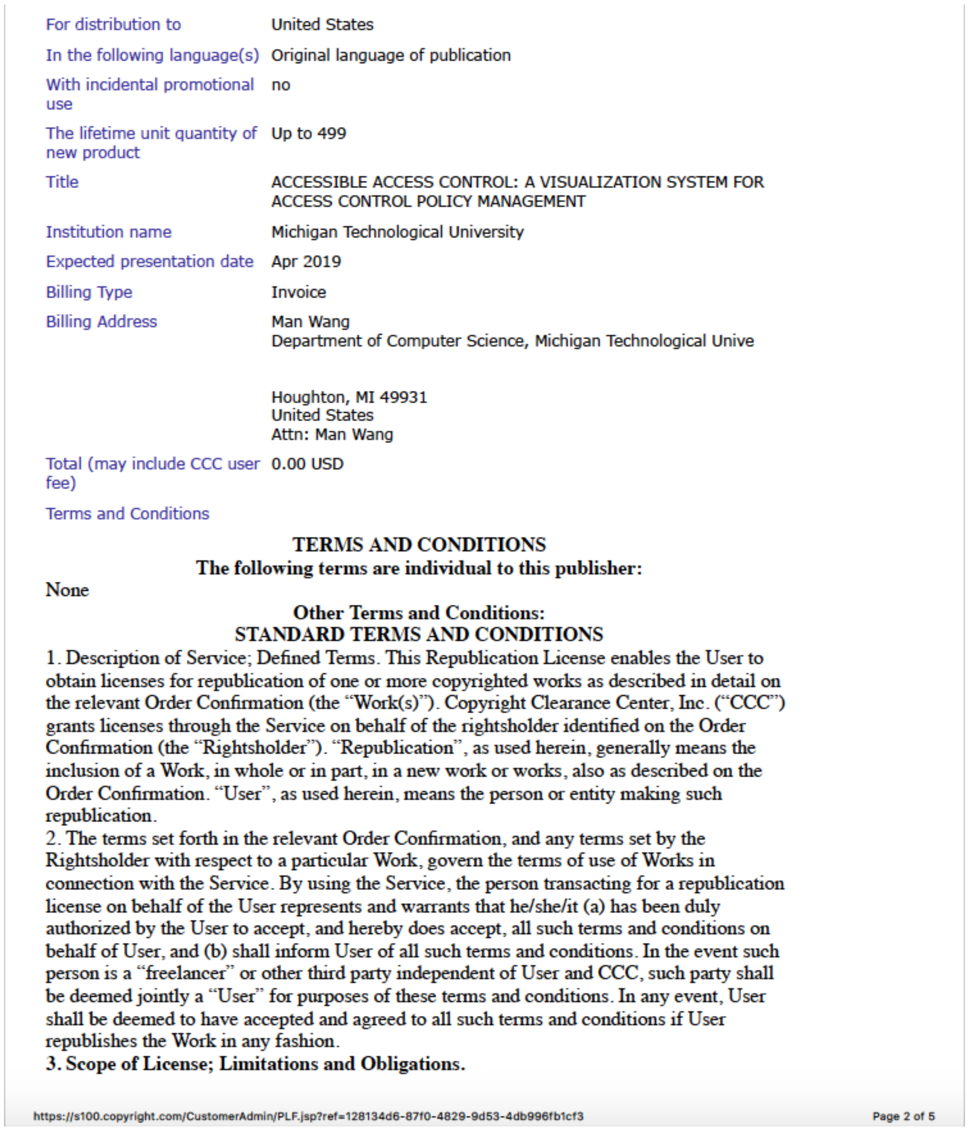
Mar 26, 2019

This is a License Agreement between Man Wang ("You") and ACM (Association for Computing Machinery) ("ACM (Association for Computing Machinery)") provided by Copyright Clearance Center ("CCC"). The license consists of your order details, the terms and conditions provided by ACM (Association for Computing Machinery), and the payment terms and conditions.

All payments must be made in full to CCC. For payment instructions, please see information listed at the bottom of this form.

License Number	4556671198634
License date	Mar 25, 2019
Licensed content publisher	ACM (Association for Computing Machinery)
Licensed content title	Proceedings, Association for Computing Machinery
Licensed content date	Jan 1, 1900
Type of Use	Thesis/Dissertation
Requestor type	Author of requested content
Format	Print, Electronic
Portion	chapter/article
The requesting person/organization is:	Man Wang
Title or numeric reference of the portion(s)	Chapter3, 4, 5
Title of the article or chapter the portion is from	UNIXvisual: A Visualization Tool for Teaching UNIX Permissions
Editor of portion(s)	N/A
Author of portion(s)	Man Wang
Volume of serial or monograph.	N/A
Page range of the portion	
Publication date of portion	May 2019
Rights for	Main product
Duration of use	Current edition and up to 5 years
Creation of copies for the disabled	no
With minor editing privileges	no

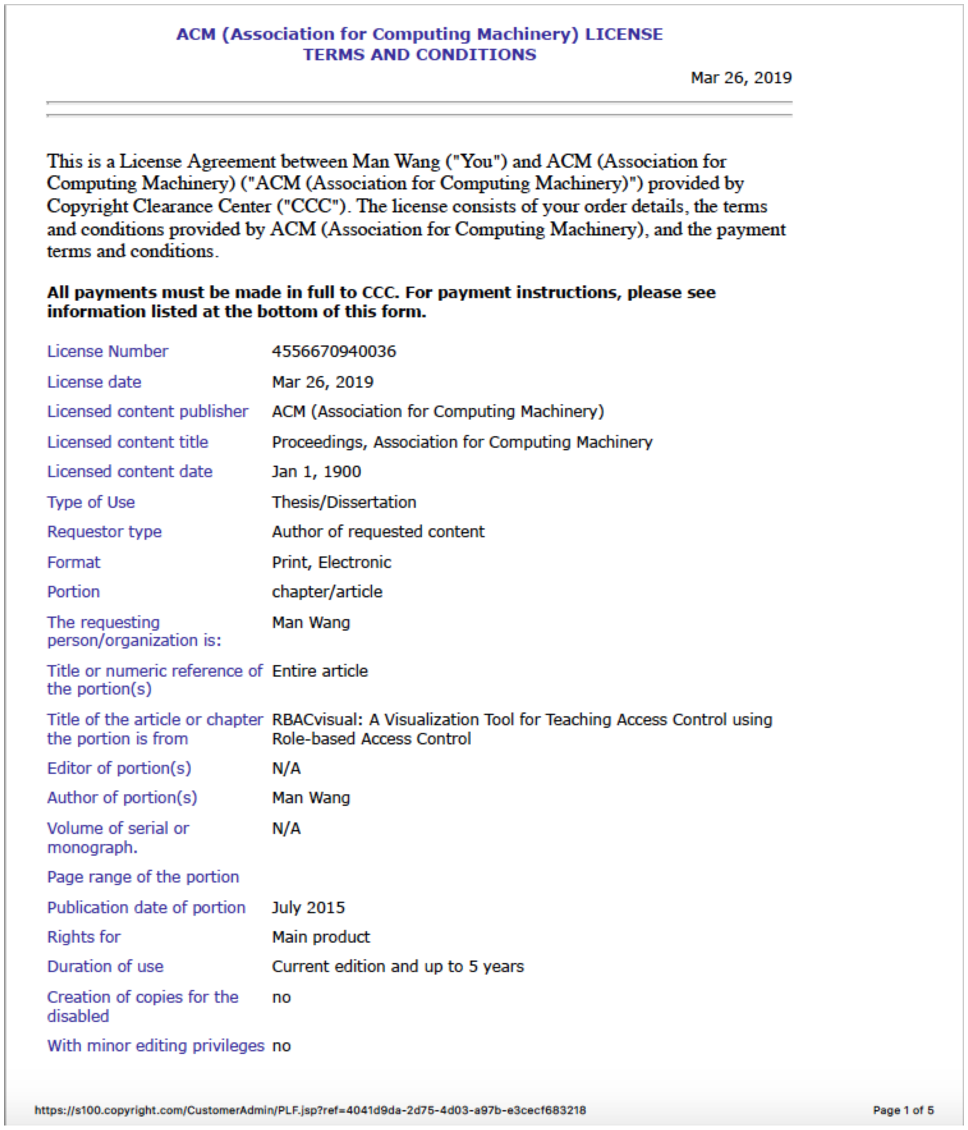
https://s100.copyright.com/CustomerAdmin/PLF.jsp?ref=128134d6-87f0-4829-9d53-4db996fb1cf3 Page 1 of 5



For distribution to	United States
In the following language(s)	Original language of publication
With incidental promotional use	no
The lifetime unit quantity of new product	Up to 499
Title	ACCESSIBLE ACCESS CONTROL: A VISUALIZATION SYSTEM FOR ACCESS CONTROL POLICY MANAGEMENT
Institution name	Michigan Technological University
Expected presentation date	Apr 2019
Billing Type	Invoice
Billing Address	Man Wang Department of Computer Science, Michigan Technological Unive Houghton, MI 49931 United States Attn: Man Wang
Total (may include CCC user fee)	0.00 USD
Terms and Conditions	

https://s100.copyright.com/CustomerAdmin/PLF.jsp?ref=128134d6-87f0-4829-9d53-4db996fb1cf3 Page 2 of 5

[74] M. Wang, J. Mayo, C.-K. Shene, T. Lake, S. Carr, and C. Wang. RBACvisual: A visualization tool for teaching access control using Role-based Access Control. In Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education, ITiCSE '15, pages 141–146, New York, NY, USA, 2015. ACM.



ACM (Association for Computing Machinery) LICENSE
TERMS AND CONDITIONS

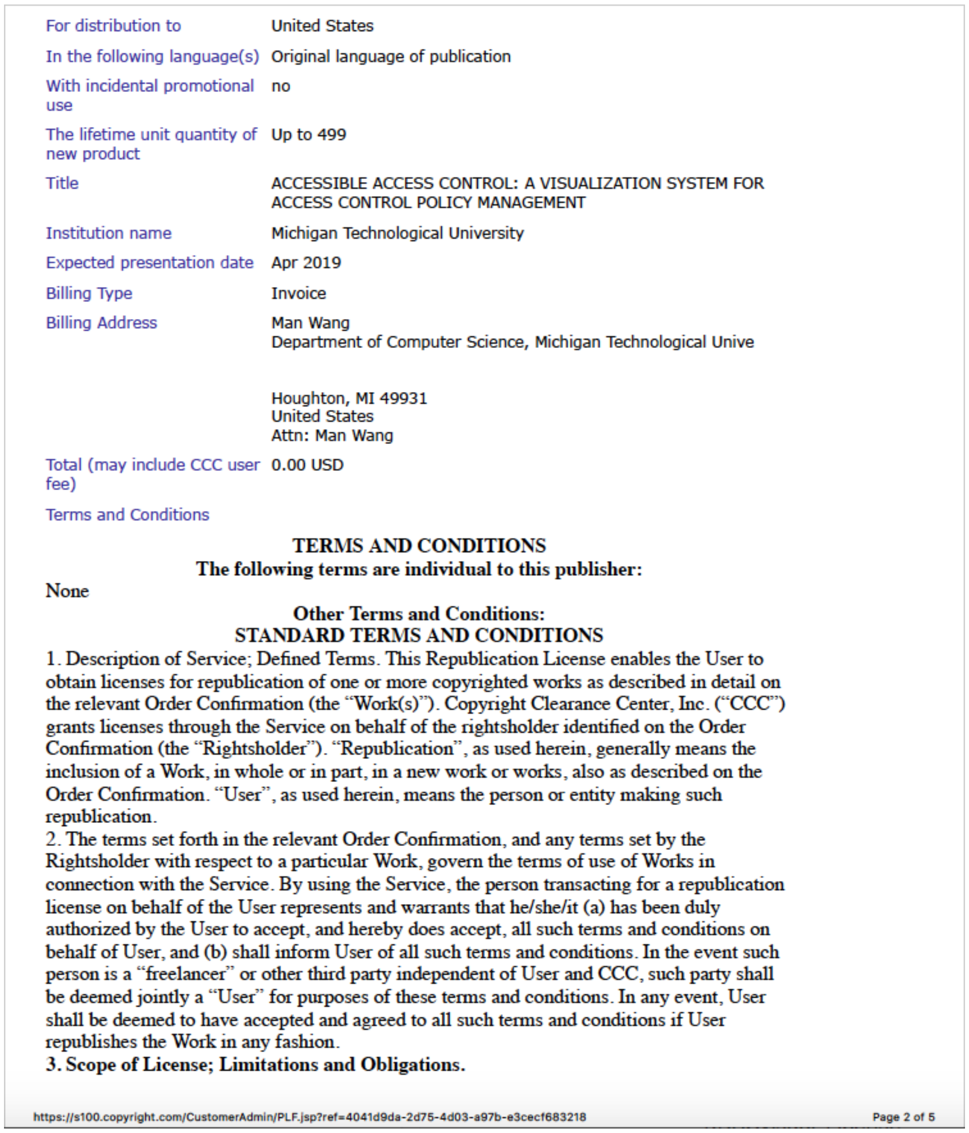
Mar 26, 2019

This is a License Agreement between Man Wang ("You") and ACM (Association for Computing Machinery) ("ACM (Association for Computing Machinery)") provided by Copyright Clearance Center ("CCC"). The license consists of your order details, the terms and conditions provided by ACM (Association for Computing Machinery), and the payment terms and conditions.

All payments must be made in full to CCC. For payment instructions, please see information listed at the bottom of this form.

License Number	4556670940036
License date	Mar 26, 2019
Licensed content publisher	ACM (Association for Computing Machinery)
Licensed content title	Proceedings, Association for Computing Machinery
Licensed content date	Jan 1, 1900
Type of Use	Thesis/Dissertation
Requestor type	Author of requested content
Format	Print, Electronic
Portion	chapter/article
The requesting person/organization is:	Man Wang
Title or numeric reference of the portion(s)	Entire article
Title of the article or chapter the portion is from	RBACvisual: A Visualization Tool for Teaching Access Control using Role-based Access Control
Editor of portion(s)	N/A
Author of portion(s)	Man Wang
Volume of serial or monograph.	N/A
Page range of the portion	
Publication date of portion	July 2015
Rights for	Main product
Duration of use	Current edition and up to 5 years
Creation of copies for the disabled	no
With minor editing privileges	no

https://s100.copyright.com/CustomerAdmin/PLF.jsp?ref=4041d9da-2d75-4d03-a97b-e3cecf683218 Page 1 of 5



For distribution to	United States
In the following language(s)	Original language of publication
With incidental promotional use	no
The lifetime unit quantity of new product	Up to 499
Title	ACCESSIBLE ACCESS CONTROL: A VISUALIZATION SYSTEM FOR ACCESS CONTROL POLICY MANAGEMENT
Institution name	Michigan Technological University
Expected presentation date	Apr 2019
Billing Type	Invoice
Billing Address	Man Wang Department of Computer Science, Michigan Technological Unive Houghton, MI 49931 United States Attn: Man Wang
Total (may include CCC user fee)	0.00 USD
Terms and Conditions	

TERMS AND CONDITIONS

The following terms are individual to this publisher:

None

Other Terms and Conditions:

STANDARD TERMS AND CONDITIONS

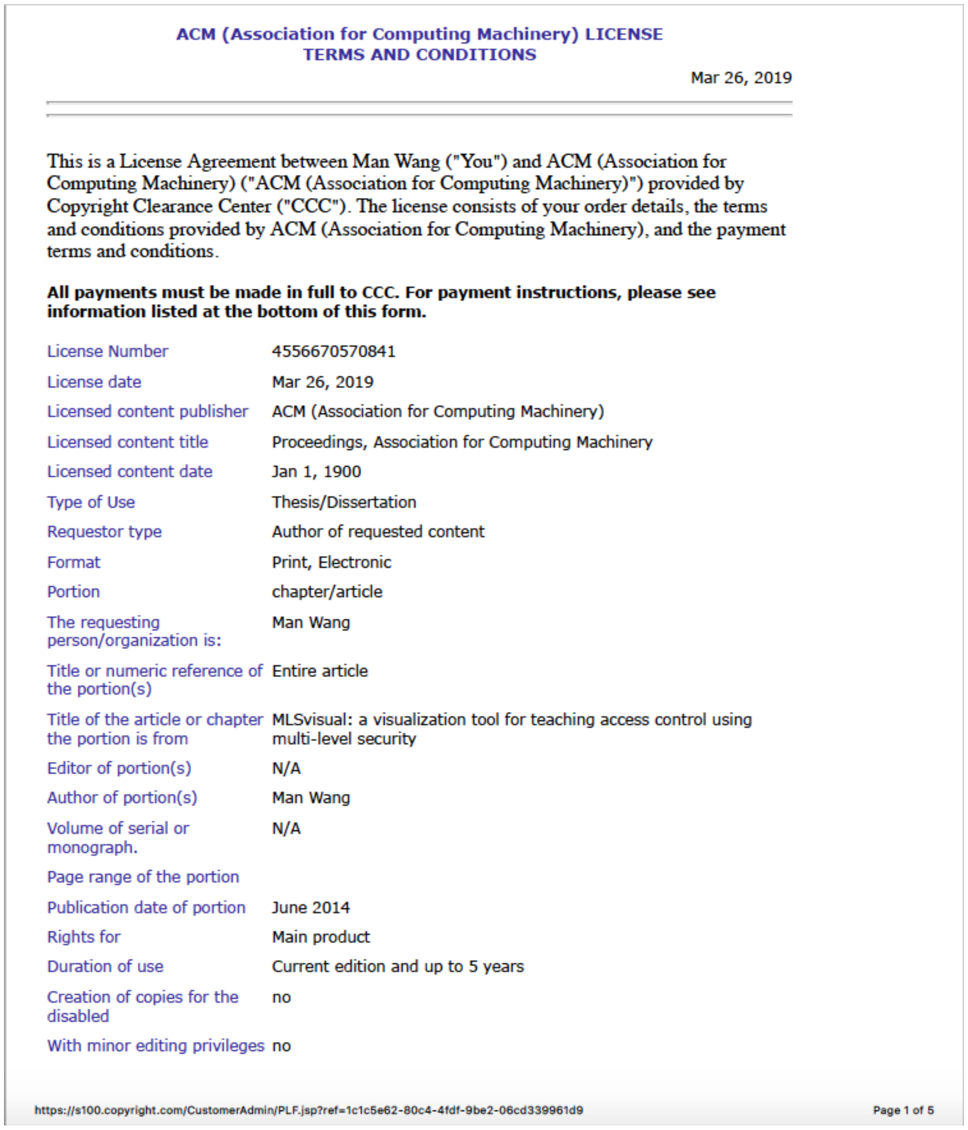
1. Description of Service; Defined Terms. This Republication License enables the User to obtain licenses for republication of one or more copyrighted works as described in detail on the relevant Order Confirmation (the "Work(s)"). Copyright Clearance Center, Inc. ("CCC") grants licenses through the Service on behalf of the rightsholder identified on the Order Confirmation (the "Rightsholder"). "Republication", as used herein, generally means the inclusion of a Work, in whole or in part, in a new work or works, also as described on the Order Confirmation. "User", as used herein, means the person or entity making such republication.

2. The terms set forth in the relevant Order Confirmation, and any terms set by the Rightsholder with respect to a particular Work, govern the terms of use of Works in connection with the Service. By using the Service, the person transacting for a republication license on behalf of the User represents and warrants that he/she/it (a) has been duly authorized by the User to accept, and hereby does accept, all such terms and conditions on behalf of User, and (b) shall inform User of all such terms and conditions. In the event such person is a "freelancer" or other third party independent of User and CCC, such party shall be deemed jointly a "User" for purposes of these terms and conditions. In any event, User shall be deemed to have accepted and agreed to all such terms and conditions if User republishes the Work in any fashion.

3. Scope of License; Limitations and Obligations.

https://s100.copyright.com/CustomerAdmin/PLF.jsp?ref=4041d9da-2d75-4d03-a97b-e3cecf683218 Page 2 of 5

[72] M. Wang, S. Carr, J. Mayo, C.-K. Shene, and C. Wang. MLSvisual: A visualization tool for teaching access control using Multi-level Security. In Proceedings of the 2014 Conference on Innovation and Technology in Computer Science Education, ITiCSE '14, pages 93–98, New York, NY, USA, 2014. ACM.



ACM (Association for Computing Machinery) LICENSE
TERMS AND CONDITIONS

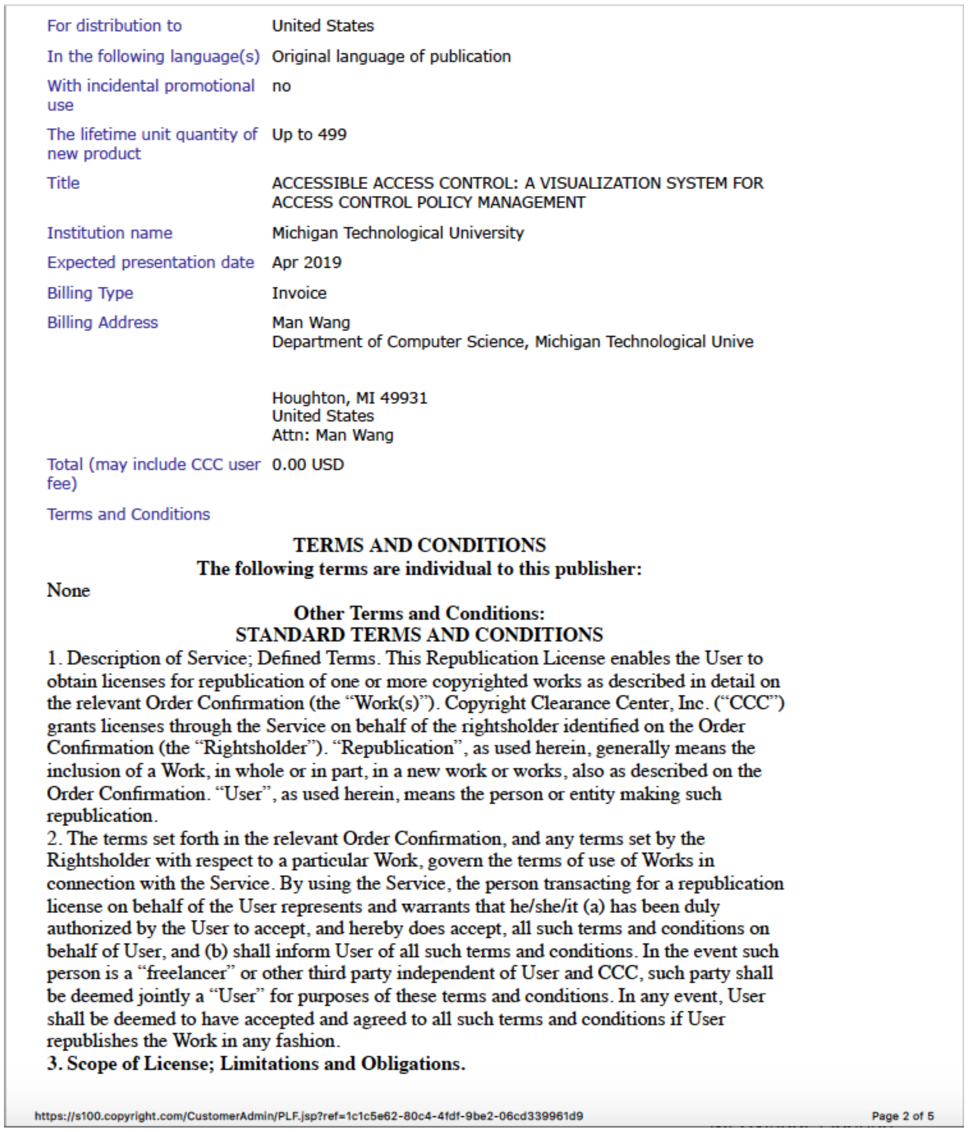
Mar 26, 2019

This is a License Agreement between Man Wang ("You") and ACM (Association for Computing Machinery) ("ACM (Association for Computing Machinery)") provided by Copyright Clearance Center ("CCC"). The license consists of your order details, the terms and conditions provided by ACM (Association for Computing Machinery), and the payment terms and conditions.

All payments must be made in full to CCC. For payment instructions, please see information listed at the bottom of this form.

License Number	4556670570841
License date	Mar 26, 2019
Licensed content publisher	ACM (Association for Computing Machinery)
Licensed content title	Proceedings, Association for Computing Machinery
Licensed content date	Jan 1, 1900
Type of Use	Thesis/Dissertation
Requestor type	Author of requested content
Format	Print, Electronic
Portion	chapter/article
The requesting person/organization is:	Man Wang
Title or numeric reference of the portion(s)	Entire article
Title of the article or chapter the portion is from	MLSvisual: a visualization tool for teaching access control using multi-level security
Editor of portion(s)	N/A
Author of portion(s)	Man Wang
Volume of serial or monograph.	N/A
Page range of the portion	
Publication date of portion	June 2014
Rights for	Main product
Duration of use	Current edition and up to 5 years
Creation of copies for the disabled	no
With minor editing privileges	no

https://s100.copyright.com/CustomerAdmin/PLF.jsp?ref=1c1c5e62-80c4-4fdf-9be2-06cd339961d9 Page 1 of 5



For distribution to	United States
In the following language(s)	Original language of publication
With incidental promotional use	no
The lifetime unit quantity of new product	Up to 499
Title	ACCESSIBLE ACCESS CONTROL: A VISUALIZATION SYSTEM FOR ACCESS CONTROL POLICY MANAGEMENT
Institution name	Michigan Technological University
Expected presentation date	Apr 2019
Billing Type	Invoice
Billing Address	Man Wang Department of Computer Science, Michigan Technological Unive Houghton, MI 49931 United States Attn: Man Wang
Total (may include CCC user fee)	0.00 USD
Terms and Conditions	

TERMS AND CONDITIONS
The following terms are individual to this publisher:

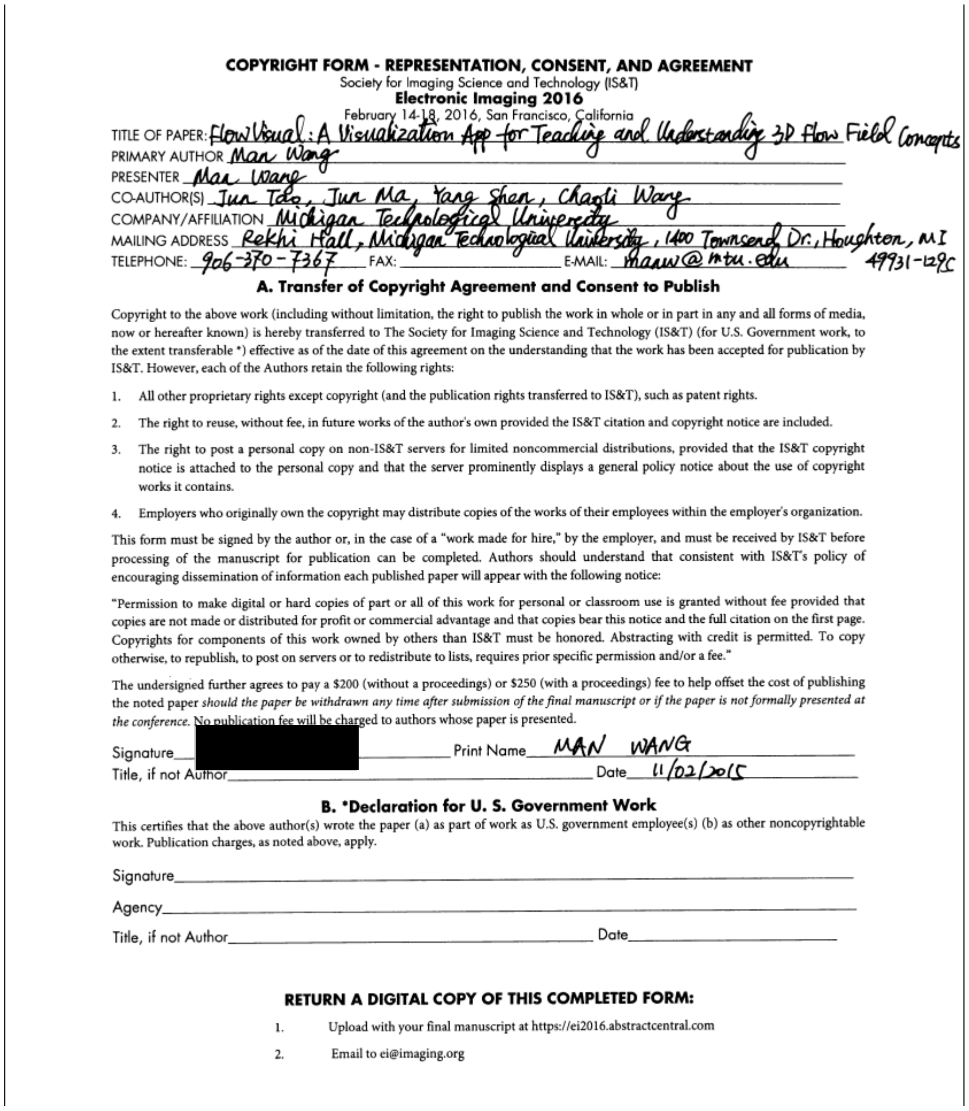
None

Other Terms and Conditions:
STANDARD TERMS AND CONDITIONS

1. Description of Service; Defined Terms. This Republication License enables the User to obtain licenses for republication of one or more copyrighted works as described in detail on the relevant Order Confirmation (the "Work(s)"). Copyright Clearance Center, Inc. ("CCC") grants licenses through the Service on behalf of the rightsholder identified on the Order Confirmation (the "Rightsholder"). "Republication", as used herein, generally means the inclusion of a Work, in whole or in part, in a new work or works, also as described on the Order Confirmation. "User", as used herein, means the person or entity making such republication.
2. The terms set forth in the relevant Order Confirmation, and any terms set by the Rightsholder with respect to a particular Work, govern the terms of use of Works in connection with the Service. By using the Service, the person transacting for a republication license on behalf of the User represents and warrants that he/she/it (a) has been duly authorized by the User to accept, and hereby does accept, all such terms and conditions on behalf of User, and (b) shall inform User of all such terms and conditions. In the event such person is a "freelancer" or other third party independent of User and CCC, such party shall be deemed jointly a "User" for purposes of these terms and conditions. In any event, User shall be deemed to have accepted and agreed to all such terms and conditions if User republishes the Work in any fashion.
3. Scope of License; Limitations and Obligations.

https://s100.copyright.com/CustomerAdmin/PLF.jsp?ref=1c1c5e62-80c4-4fdf-9be2-06cd339961d9 Page 2 of 5

[75] M. Wang, J. Tao, J. Ma, Y. Shen, and C. Wang. FlowVisual: A visualization app for teaching and understanding 3D flow field concepts. In Visualization and Data Analysis 2016, San Francisco, California, USA, February 14-18, 2016, pages 1-10, 2016.



COPYRIGHT FORM - REPRESENTATION, CONSENT, AND AGREEMENT

Society for Imaging Science and Technology (IS&T)

Electronic Imaging 2016

February 14-18, 2016, San Francisco, California

TITLE OF PAPER: FlowVisual: A Visualization App for Teaching and Understanding 3D Flow Field Concepts

PRIMARY AUTHOR Man Wang

PRESENTER Man Wang

CO-AUTHOR(S) Jun Tao, Jun Ma, Yang Shen, Chaoli Wang

COMPANY/AFFILIATION Michigan Technological University

MAILING ADDRESS Rekhi Hall, Michigan Technological University, 1400 Townsend Dr., Houghton, MI 49931-1295

TELEPHONE: 906-370-7367 FAX: ______ E-MAIL: manw@mtu.edu

A. Transfer of Copyright Agreement and Consent to Publish

Copyright to the above work (including without limitation, the right to publish the work in whole or in part in any and all forms of media, now or hereafter known) is hereby transferred to The Society for Imaging Science and Technology (IS&T) (for U.S. Government work, to the extent transferable *) effective as of the date of this agreement on the understanding that the work has been accepted for publication by IS&T. However, each of the Authors retain the following rights:

1. All other proprietary rights except copyright (and the publication rights transferred to IS&T), such as patent rights.
2. The right to reuse, without fee, in future works of the author's own provided the IS&T citation and copyright notice are included.
3. The right to post a personal copy on non-IS&T servers for limited noncommercial distributions, provided that the IS&T copyright notice is attached to the personal copy and that the server prominently displays a general policy notice about the use of copyright works it contains.
4. Employers who originally own the copyright may distribute copies of the works of their employees within the employer's organization.

This form must be signed by the author or, in the case of a "work made for hire," by the employer, and must be received by IS&T before processing of the manuscript for publication can be completed. Authors should understand that consistent with IS&T's policy of encouraging dissemination of information each published paper will appear with the following notice:

"Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than IS&T must be honored. Abstracting with credit is permitted. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee."

The undersigned further agrees to pay a $200 (without a proceedings) or $250 (with a proceedings) fee to help offset the cost of publishing the noted paper *should the paper be withdrawn any time after submission of the final manuscript or if the paper is not formally presented at the conference.* No publication fee will be charged to authors whose paper is presented.

Signature [redacted] Print Name MAN WANG

Title, if not Author ______ Date 11/02/2015

B. *Declaration for U. S. Government Work

This certifies that the above author(s) wrote the paper (a) as part of work as U.S. government employee(s) (b) as other noncopyrightable work. Publication charges, as noted above, apply.

Signature ______

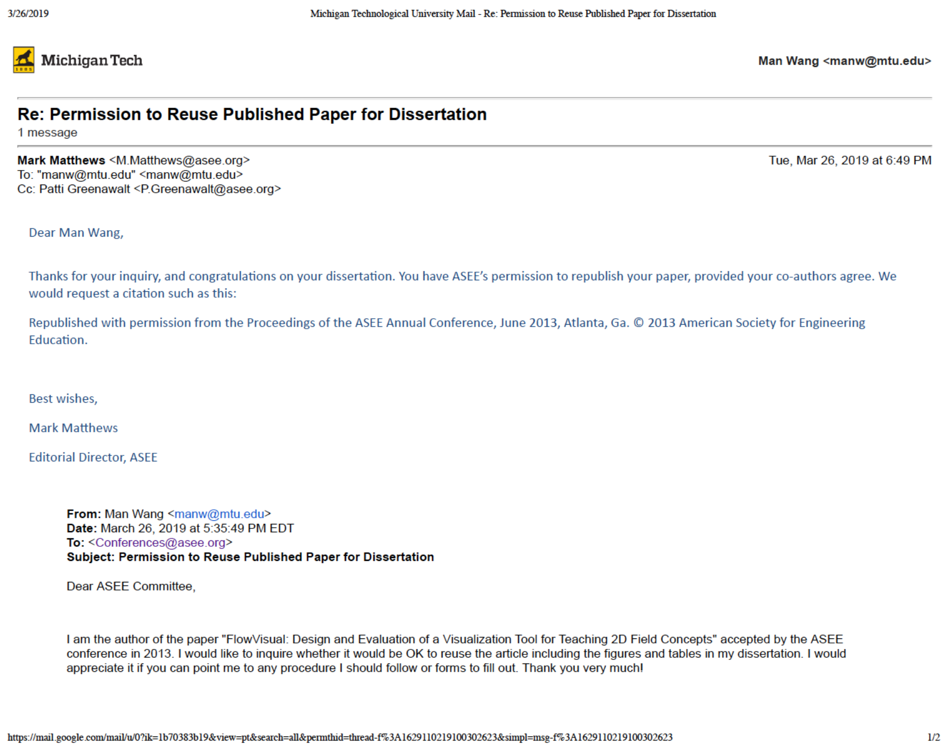
Agency ______

Title, if not Author ______ Date ______

RETURN A DIGITAL COPY OF THIS COMPLETED FORM:

1. Upload with your final manuscript at https://ei2016.abstractcentral.com
2. Email to ei@imaging.org

[76] M. Wang, J. Tao, C. Wang, C.-K. Shene, and S. H. Kim. FlowVisual: Design and evaluation of a visualization tool for teaching 2D flow field concepts. In Proceedings of American Society for Engineering Education Annual Conference, 2013.



3/26/2019 Michigan Technological University Mail - Re: Permission to Reuse Published Paper for Dissertation

Michigan Tech Man Wang <manw@mtu.edu>

Re: Permission to Reuse Published Paper for Dissertation
1 message

Mark Matthews <M.Matthews@asee.org> Tue, Mar 26, 2019 at 6:49 PM
To: "manw@mtu.edu" <manw@mtu.edu>
Cc: Patti Greenawalt <P.Greenawalt@asee.org>

Dear Man Wang,

Thanks for your inquiry, and congratulations on your dissertation. You have ASEE's permission to republish your paper, provided your co-authors agree. We would request a citation such as this:

Republished with permission from the Proceedings of the ASEE Annual Conference, June 2013, Atlanta, Ga. © 2013 American Society for Engineering Education.

Best wishes,

Mark Matthews

Editorial Director, ASEE

From: Man Wang <manw@mtu.edu>
Date: March 26, 2019 at 5:35:49 PM EDT
To: <Conferences@asee.org>
Subject: Permission to Reuse Published Paper for Dissertation

Dear ASEE Committee,

I am the author of the paper "FlowVisual: Design and Evaluation of a Visualization Tool for Teaching 2D Field Concepts" accepted by the ASEE conference in 2013. I would like to inquire whether it would be OK to reuse the article including the figures and tables in my dissertation. I would appreciate it if you can point me to any procedure I should follow or forms to fill out. Thank you very much!

https://mail.google.com/mail/u/0?ik=1b70383b19&view=pt&search=all&permthid=thread-f%3A1629110219100302623&simpl=msg-f%3A1629110219100302623 1/2