Discretionary Access Controls for a Collaborative Virtual Environment

Timothy E. Wright and Greg Madey

Computer Science & Engineering
University of Notre Dame
Notre Dame, IN 46556
Email: twright@nd.edu
Email: gmadey@nd.edu

Abstract

As collaborative virtual environments (CVEs) are more widely used, participant access to CVE objects and information becomes a significant concern. In virtual reality games, storefronts, classrooms, and laboratories for example, the need to control access to spaces and objects is integral to the security of activities in these virtual realms. However, limited access controls are typically available in CVEs. Often, these mechanisms are course-grained, only protecting against movements by unauthorized participants into specific areas. In answer to this deficiency, we offer a discretionary access control (DAC) system based on the traditional concepts of users and groups, and tailored to the needs of a CVE. Our system, called WonderDAC, includes the ability to restrict movement into areas, as well as control interactions with objects. A basic WonderDAC prototype has been implemented within the Project Wonderland CVE.

Keywords: Virtual reality, collaborative virtual environment, access control, discretionary access

1 Introduction

We take the term "collaborative virtual environment" (CVE) to mean a graphical, virtual reality environment capable of operating on a typical, modern workstation. Often such environments are associated with game playing (e.g., the Quake series of games, massively multiplayer online role-playing games [MMORPGs] such as World of Warcraft), though commerce, educational, research, and social networking systems are also being realized through CVE technology [Ben07] [BBPK05] [BK07]. The essential quality of any such environment is, of course, the collaboration among participants; however, this can also serve as an avenue for problems. Players in a game may elect to cheat, patrons of a virtual store might try to steal, researchers could inadvertently damage information or interfere with an experiment, and participants of a social networking system might have sensitive personal information exposed. Problems of this ilk have manifested themselves in commercial CVEs such as Second Life. For example, the December 2006 malicious disruption of a CNET interview taking place in Second Life [Ter06], the March 2007 defacement attack carried out on John Edwards' presidential campaign headquarters [Bro07], the April 2007 attack on Toyota's Scion storefront [Wag07], or the myriad of other, similar attacks and disruptions caused by miscreants (also called "griefers") in Second Life [Dib08] [Fas07]. Malware exploits within CVE worlds are also a concern, as demonstrated by the November 2007 QuickTime vulnerability within Second Life [Lin07c]: here, an attacker could leverage a bug in the QuickTime player to "crash or exploit the Second Life viewer." Through the presence of better privacy and integrity controls, incidents and weaknesses such as these could have been substantially mitigated, if not outright prevented.

To manage the integrity and privacy risks of a CVE, we note that security works best in layers, and that there are at least three layers where controls should be employed. The use of layered security controls, or *defense-in-depth*, is a well understood means for reducing the chances of malicious and accidental damage to information systems [Bro04] [SWJ06] [Sch01]. The basic concept is that multiple controls can operate in a synergistic fashion, collectively reducing

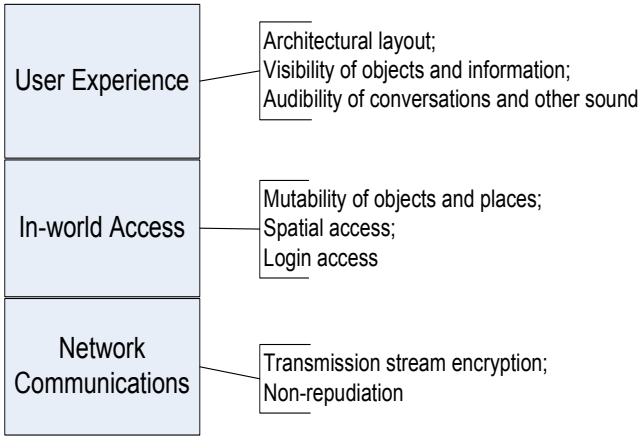

Figure 1: Layers of security in a CVE.

risk more broadly than as individual controls. Moreover, if one control becomes disabled or compromised, the other controls will still be in effect and may compensate. For the purposes of security, we believe a CVE consists of the following layers: network communications, in-world access, and user experience. The network communications layer includes those mechanisms that transmit data between a CVE client and server, or among CVE peers (as the case might be). The in-world access layer encompasses course-grained access to a CVE system, its map, and resources. Finally, the user experience layer deals with the immersive qualities of a CVE, including the sense of awareness and privacy the virtual world's architecture imparts, and a participant's virtual ability to see and hear. Figure 1 depicts these layers.

We choose to focus on security in the middle layer for two reasons. First, network layer security is already an area of long-standing, significant research. Regardless of whether a CVE or some other system resides on top of this layer, the options and techniques to secure network communications are the same. Second, although there is no strong coupling between the network communications and in-world access layers, there is between the top two layers. In particular, the means of authentication and authorization used for the in-world access layer should be leveraged by the user experience layer, too. Hence, this dependency compels us to deal with the in-world access layer initially.

Our approach to securing the in-world access layer follows that of classic, UNIX-style, file system discretionary access controls (DAC). We imitate this type of DAC primarily because it is well-known in many general operating systems, easy to comprehend, and easy to deploy. We avoid more complex access control list systems, such as that found in Microsoft Windows XP/2000/2003, because their complexity can lead to misconfiguration [GA06].

UNIX-style DACs are based upon the assignment of permissions to three roles: the owner of a file system object, the group to which that object belongs, and all other system users. The basic form of UNIX-style DAC employs read, write, and execute permissions to permit or deny access to a directory or file; we propose something analogous though even simpler. The tenets of our system may be concisely stated as follows.

- Access controls must always be inherently simple to understand and manage
- The CVE's virtual world should be viewed as a collection of objects: terrain, avatars, decor, accessories, media (i.e., sound, video, still image), etc.
- Access should be controlled through two different permissions: *interact* and *alter*
- All virtual world objects must have an assigned owner and participant group; a default *other* role includes all participants who are neither the owner nor in the object's group
- Interact and alter permissions should be specified for an object's group and all other participants—the owner of an object will always have full access to the object
- Any participant should be able to determine an object's ownership; the owner of an object should be able to determine that object's permissions, and may transfer ownership to a different participant

Semantics of the interact and alter permissions are comparable to the classic read, write, and execute file system capabilities. Interact denotes read and execute, while alter is equivalent to write. In a VR context, interacting with an object entails viewing/hearing ("reading") and possibly using ("executing"), while altering an object involves changing, deleting, or updating in some way ("writing").

We have implemented a prototype of this DAC system, called WonderDAC, by extending Project Wonderland. We selected this CVE because of three important factors: first, the project is at an early, formative stage, making it easy to integrate the components

of our DAC system; next, Wonderland's approach to handling VR objects (called *cells* in Wonderland parlance) maps very well to the tenets we state above; and, finally, the server component of a client-server CVE, such as Wonderland, offers a logical, secure place to implement controls [WP07].

The remainder of this paper is organized as follows: the next section summarizes related works; Section 3 presents background and details about Project Wonderland and its core architecture; Section 4 outlines several use case scenarios that we propose to enable with the WonderDAC prototype, Section 5 illustrates our prototype implementation; and Section 6 ends the paper with summary remarks and a brief discussion of proposed research and development.

2 Related Work

As discussed in the previous section, there are at least three layers wherein security controls should be implemented for a CVE: network communications, in-world access, and user experience. Protecting the network communications layer has been a carefully studied problem in many other research endeavors, the outcome of which is applicable to CVEs or any other systems that transmit data using network protocols. For example, among other techniques, [DDG+03], [GMP+02], and [SMM00] all propose encrypting the network protocols that underly their respective CVEs. Other network research peculiar to CVEs has touched on the integrity of the gaming experience. Along these lines, [BLL07], [Chi97], [NPVS07] each consider how to provide consistent and fair participant interactions through methods such as dead-reckoning and bandwidth reduction.[1]

Regarding the middle CVE layer, in-world access, we find a limited number of efforts that deal with security controls. For example, [BB99] derives a spatial access control scheme, called SPACE, and notes that through SPACE's adoption "a natural part of the environment is exploited, making it possible to hide explicit security mechanisms from end users through the natural spatial makeup of the environment." SPACE works by leveraging an access graph (a structure denoting constraints when moving from one location to another) to build adjacency and classification matrices. The adjacency matrix serves as a map, while the classification matrix determines the permissions a participant needs in order to move about a CVE. In [PM01] a more refined, but complex, approach to access control is proposed, wherein CVE participants can interact with VR objects and exchange messages with one and other in order to gain access to resources. Central in this approach is participants' use of keys to prove their identities and authority when undertaking some action.

Other means of in-world access control exploit VR metaphors. In [BBF98], so-called privacy lamps and vampire mirrors are utilized to affect and verify object access. Privacy lamps shine a spot light on virtual objects that are to be hidden from view, while vampire mirrors reveal objects that are private (such objects won't show up in the mirror) and can make an object private when a participant touches its mirror image.

Finally, some commercial, social networking CVEs offer in-world access controls, although these are typically geared toward the ownership, sale, and rental of virtual property. Second Life, for example, has extensive controls dealing with these issues, as do There, and Activeworlds [Act08c] [Act08b] [Act08a] [Lin07b] [Lin08] [Lin07a] [Mak04] [Mak08] [Mak06]. The concept of role-based access (through a group) is also available in Second Life relative to land management. All three CVEs provide for other access control mechanisms that are tailored to the social networking experience (e.g., permissions to restrict flying and building). Often such controls require expertise with the CVE in question in order to be correctly applied and managed.

In the top CVE layer, user experience, consideration has been given to how structural divisions and social conventions play a role in participant behavior and privacy. Each of [HD96], [HTK+97b], and [HTK+97a] explores how the design of virtual spaces can enable or discourage privacy by controlling views into work areas. The intersection of technical and social components in media environments (such as CVEs) is also examined by [MAIO97], wherein it is noted that successful virtual communities must adapt and evolve practices and conventions from the real world. In so doing, visual cues (e.g., the layout of a space, signs) can affect participant behavior and the respect for others' virtual world privacy. Along these lines, [RZ04] notes that the privacy of a CVE partici-

[1] Dead-reckoning is widely used to improve the appearance of movement among participants and other objects in a virtual reality context. In essence, the position of a moving object is predicted based on it's velocity and previous location. Updated information about the object's position is only required if it differs substantially from the prediction.

pant (i.e., the right of that participant not to be hounded in the CVE) may sometimes be an issue, too. Unfortunately, resetting a participant's virtual identity (their name and avatar appearance) may disrupt associations with friends or fail to work if the new identity becomes linked to their old one.

3 Project Wonderland

3.1 Background

Project Wonderland is an open source, Java-based CVE that is constructed around a client-server model. Developed primarily by Sun Microsystems, Wonderland draws upon the technologies of four other open source projects (also operated by Sun):

- Project Darkstar: the server component for Wonderland; designed to offer a scalable, distributed, transaction-based game server infrastructure
- jVoiceBridge: the audio server and client for Wonderland; provides spatial, stereo sound as well as the ability to place telephone calls through the Session Initiation Protocol (SIP)
- Java 3D: responsible for providing the Wonderland client's 3D graphics and scene graph
- Project Looking Glass - used by the Wonderland client for 3D scene management (e.g., to create the client's user interface)

Compared to similar technologies, such as Croquet or Second Life, Wonderland is a relatively new offering in the realm of CVEs. First available in January 2007, this system incorporates a significant set of features, including, but not limited to: a scalable, virtual environment specified through a simple, XML file hierarchy; a transaction-based, fault-tolerant server that enables atomic participant activities; built-in avatar support (including a simple editor); spatial sound and audio chat; a virtual whiteboard; the ability to operate and share X Window applications within the Wonderland virtual world; and the means to connect a virtual phone with any SIP compliant phone system. The impetus for these features is a result of Sun's vision that Wonderland be "an environment that is robust enough in terms of security, scalability, reliability, and functionality that organizations can rely on it as a place to conduct real business [Sun08b]." These goals are meant to encompass remote office, educational, and other types of multi-user collaboration.

3.2 Client-Server versus Peer-to-Peer

Our ability to extend Wonderland with a DAC mechanism is enabled by Wonderland's client-server architecture. The client-server approach gives us a central authority, in the form of a server, that can be protected from malicious parties. By contrast, a peer-to-peer environment complicates the use of security controls, since each participant is on equal footing with full access to resources and information; serious problems can arise if one or more peers is under the control of a malicious user. As discussed in [NPVS07], with regards to cheating in a peer-to-peer CVE-based game, numerous obstacles and issues must be resolved to enable reliable security controls:

> First, it may be very complicated to apply cheating resistance extensively to a whole peer-to-peer architecture, that is mostly made of uncontrolled hosts. Moreover, it is unclear where to place the mechanisms to be used: on all the devices that take part to the game, or only on a subset of them? In this latter case how is this subset selected? Aside from this, in the case of a fully decentralized approach, the management of trust between the devices and between the players is a complex issue. How can we be sure that a device or a player acts faithfully? How is trust established and maintained? How is slandering handled?

3.3 Wonderland Cells and the Wonderland File System

Our implementation of the WonderDAC prototype takes place only in Wonderland's server components. However, as we will demonstrate in Section **??**, there is still a need to enhance the client-side code in order to hide some of the cosmetic side-effects of WonderDAC—see below for details. As mentioned above in Section 1, a Wonderland "...virtual world is composed of a collection of 'cells', each of which represents a 3D volume in the world [Sun08a]." A cell can denote a single decorative object in a virtual room, the room's architecture, or the entire virtual world. Hence, there is a hierarchical, parent-child structure at work in the use of cells, where a given cell may contain zero or more other cells. Moreover, this structure is represented by the XML files that prescribe the layout of a Wonderland virtual world and make up the Wonder-

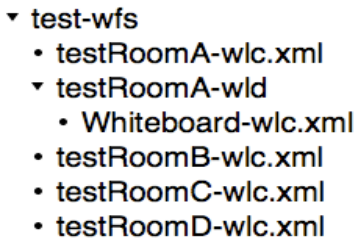

Figure 2: XML file hierarchy for a test Wonderland world.

land file system (WFS) [Sun08c]. Figure 2 provides an example WFS hierarchy comprising a simplistic world, while Appendix B includes the XML source code for these files. Each XML file is actually the serialization of a Java bean (called BasicCellGLOSetup) that handles the fundamentals of creating a generic cell. At the root of any WFS is a directory with a name ending in the *-wfs* suffix; within this directory XML files that end with a *-wlc.xml* suffix correspond to individual cells. It is possible to load up a cell file with an array of virtual object models: in this case, all objects are constituents of the cell. Alternatively, one can establish a parent-child relationship by creating a directory named after a given cell, but with a *-wld* suffix, and then populating that directory with new cell files (this is seen in Figure 2 where *testRoomA* is the parent of the cell given by *Whiteboard-wlc.xml*. All cells have virtual world coordinates that are relative to their parent's coordinates.

3.4 Communications

Two forms of communication are employed within the Wonderland client-server architecture: direct and publish/subscribe channels [Sun08a]. The former is primarily used in activities such as a client's initial connection to the Wonderland server, error message transmission, the management of shared applications, and handling virtual phone calls. The latter facilitates communications between clients and the server, as well as among clients (by way of the server). In the publish/subscribe channels, communications involve participants entering and leaving the Wonderland environment, cell management, avatar setup and movement, and server management (through a dedicated server management client).

The driving force behind using a combination of direct and publish/subscribe communications is efficiency: some situations call for the transmission of data between a given client and the server, while other

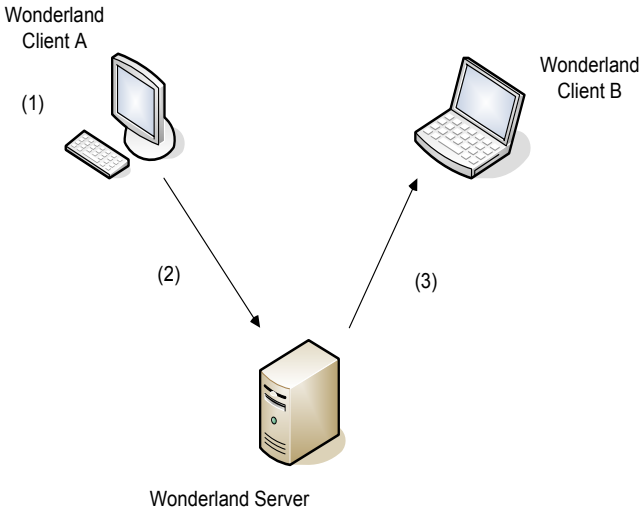

Figure 3: Server state change: communications directly between a client and the server.

situations may not involve the server in a logical sense. For example, if a participant enters or leaves the Wonderland environment, the server must be notified directly; if a participant causes a cosmetic change that only they and other nearby participants should see (e.g., a virtual object lights up when a mouse hovers over it) there is no need to update information on the server. In other words, there can be state changes on a client, and state changes on the server—figures 3 and 4 illustrate this. In Figure 3, a server state change has taken place (step 1) as a result of some activity by Client A. In this circumstance, Client A directly contacts the server with an appropriate message (step 2); Client B is then updated by the server over a channel to which Client B subscribes (step 3). In Figure 4, however, a cosmetic change has taken place within a cell that Client A is accessing (step 1). Since there is no corresponding change in server state for the cell, the cell publishes an update to its channel (step 2a), which is subscribed to by Client B (step 2b).

4 Use case Scenarios

In Section refintro, our list of tenets for the WonderDAC prototype started with a goal of simplicity and ease of understanding. As we consider what we believe to be representative use case scenarios for CVE access control, we bear this initial goal in mind. To that end, our approach with WonderDAC is modeled after the classic, UNIX-style DAC system, and assigns access permissions to three roles, *owner*, *group*, and *other*, for a given VR object. As opposed to

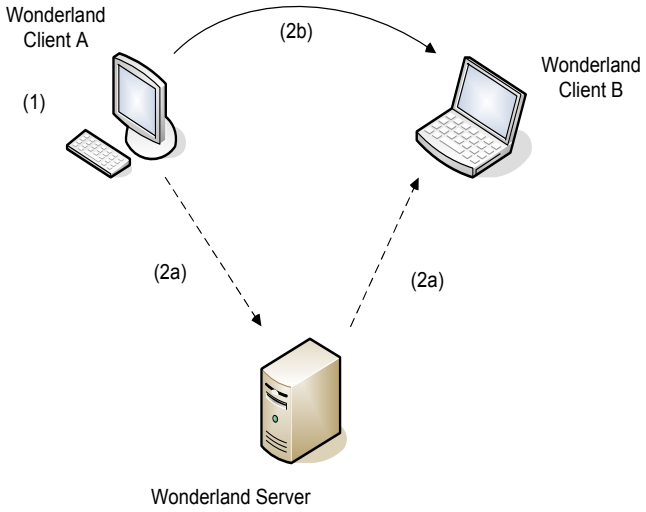

Figure 4: Client-only state change: communications among Wonderland clients (by way of the server).

read, write, and execute, WonderDAC employs the VR analogs of *interact* (similar to read/execute) and *alter* (similar to write). By having the interact permission for an object, one can use, but not change, the object. By having the alter permission, one can change (i.e., update or delete) the object. As a participant operates within Wonderland, the privileges they are assigned for a given object are mutually exclusive: if the participant is a member of an object's group, then they will be assigned only the object's group permissions; permissions for an object's *other* role are assigned to a participant when the *group* role cannot be applied. This means that an object's *group* role can be used to grant or deny access to a select list of participants.

Not all of the following use cases have been implemented in the current WonderDAC prototype. Instead, we have focused on very basic components for use cases 1 and 2, laying the foundation for handling the remaining use cases in future work (see Section 6).

4.1 Use Case 1: Spatial Object Restrictions

A participant should only be able to enter, hear, and see into a virtual space if they are the owner of the space, a member of the space's group (and the group has interact permission), or the space is defined with interact permission for the *other* role. When a participant's avatar has entered a space, the avatar should only be visible and heard by other participants who also have interact permission (but may or may not be inside the space).

A participant should only be able to change a virtual space (by speaking in, or deleting/updating the space) if they are the owner of the space, a member of the space's group (and the group has alter permission), or the space is defined with alter permission for the *other* role. Although it is possible for a space to have only the alter permission assigned to the *group* or *other* roles, the result of such a configuration would be meaningless, since non-owners would be incapable of entering the space to begin with.

4.2 Use Case 2: Non-spatial Object Restrictions

A participant should only be able to view and, if applicable, hear a non-spatial object (e.g., a whiteboard, phone, or X Window application) if they are the owner of the object, a member of the object's group (and the group has interact permission), or the object is defined with interact permission for the *other* role. For example, a participant with only interact permission for a whiteboard can view the whiteboard and its contents, but cannot change or add to those contents. Similarly, a participant with only interact permission for a conference phone, can view and hear the phone, but cannot join in the conversation.

A participant should only be able to change/utilize a non-spatial object if they are the owner of the object, a member of the object's group (and the group has alter permission), or the object is defined with alter permission for the *other* role. Similar to Use Case 1, although it is possible for a non-spatial object to have only the alter permission assigned to the *group* and *other* roles, such a configuration is meaningless since non-owners would be unable to view/hear the object in the first place.

4.3 Use Case 3: Audio Conversation Restrictions

Two or more participants engaged in audio chat should be able to restrict their conversation to themselves (somewhat like having a "cone of silence" at their disposal). Other participants should be able to solicit the initial group to join in the conversation if the group wishes. This amounts to a specialized version of Use Case 1 except that an ephemeral, invisible space is created around the participants:

1. A temporary group role, to which all of the conversation participants belong, is created and assigned to the invisible space

2. Interact and alter permissions are assigned to the space's group role

3. New participants may be added to the temporary group role as desired by the initial conversation participants

It should be possible to assign/remove the interact and alter permissions for the invisible space's *other* role. This could enable, through the assignment of only the interact permission, members of the temporary group to converse, while all others can listen but not interrupt. The use of alter permissions alone for the invisible space's *other* role may or may not make sense, depending on the desires of the conversation participants: a participant in the *other* role would be able to speak in, but unable to hear, the conversation.

4.4 Use Case 4: Avatar Cloaking

A participant should be able to disguise their avatar's appearance. Each avatar should have a group role that is unique to its owner. By assigning or removing interact permissions to this *group* or to the avatar's *other* role, the appearance of the avatar should be controllable by its owner. A participant should be able to see an avatar's true image if they are a member of the avatar's group (and the group has interact permission), or the avatar is defined with interact permission for the *other* role. Otherwise, a generic image should be displayed by the avatar. The use of alter permissions should be ignored, here: only the owner should be permitted to make changes to an avatar's appearance.

4.5 Use Case 5: Permissions and Ownership Changes

All participants should be able to determine the ownership of a VR object. Also, a participant should be able to change permissions for the *group* and *other* roles of any object they own, or, if desired, assign ownership of the object to another participant/group. The interface to carry out these activities should be simple and accessible through mouse interactions with the object in question.

4.6 Summary Use case Table

Table 1, at the end of this article, presents the VR objects discussed in the use case scenarios above.

5 Prototype Implementation

As discussed in Section 4, our focus during the implementation of this prototype was on very basic components for use cases 1 and 2. To begin with, we leveraged Wonderland's ability to use the lightweight directory access protocol (LDAP) as the means of authenticating a participant.[2] We extended this to include authorization by storing a group membership list in each participant's LDAP record. When a participant logged into Wonderland to authenticate, their membership list was accessed and parsed into an array of group names, which was then stored in an associated WonderlandIdentity object. Due to contraints in Darkstar, we had to implement a special Darkstar service to enable access to a participant's identity information later in the login process.

Next, we took advantage of a pre-existing, empty, class stub that Wonderland developers had inserted into their source code as a placeholder for future access control checks. Specifically, we replaced the stub (called CellAccessControl) with a new class that could compare a given cell's group with the group membership of a participant, and then, based on the outcome, perform subsequent permissions checks. This new class put into effect the mutually exclusive assignment of permissions discussed at the outset of Section 4: "...if the participant is a member of an object's group, then they will be assigned only the object's group permissions; permissions for an object's *other* role are assigned to a participant when the *group* role cannot be applied."

Finally, we made modifications to other central Wonderland classes. This included, but was not limited to: CellGLO, the parent class for all Wonderland cells; BasicCellGLOSetup, a utility class to aid in configuring a cell; and WhiteboardCellGLO, the server component of a Wonderland whiteboard cell. Our modifications to BasicCellGLOSetup enabled us to add the following property tags to the XML files that specify Wonderland cells:

- accessOwner – The name of the cell's owner
- accessGroup – The name of the cell's group
- accessGroupPermissions – A number indicating the interact and alter permissions assigned to the cell's group ("I A" = 3, "I -" = 2, "- A" = 1, "- -" = 0)

[2]This ability also includes support for strongly encrypted communications between the LDAP and Wonderland servers.

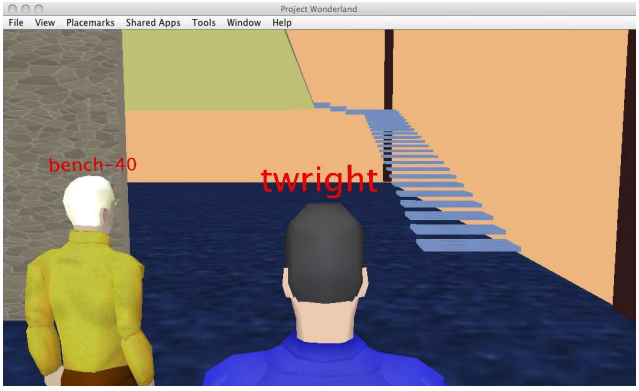

Figure 5: The avatar twright looks on at testRoomD and testRoomC.

Figure 6: The avatar bench-40 looks on at testRoomC, but cannot see testRoomD.

- accessOtherPermissions – A number indicating the interact and alter permissions assigned to the cell's *other* role

5.1 Spatial Object Access

Here, we demonstrate how the WonderDAC prototype was able to control spatial access. For this situation, there were two cells (spaces) of interest: a two-story room referred to as testRoomC and, within this space, a second story loft with stairs leading up, referred to as testRoomD. For the purposes of this demonstration, testRoomC was configured as follows: (twright, admin, 2, 2). This tuple is shorthand for denoting that the cell's owner is *twright*, the cell's group is *admin*, the group permissions are *I* - and the permissions for the *other* role are *I* -. The loft, testRoomD, was configured with: (twright, admin, 0, 0), where a 0 denotes no permissions granted. In Figure 5, we see two avatars, twright and bench-40, looking at the spaces created by testRoomC and testRoomD; this figure is

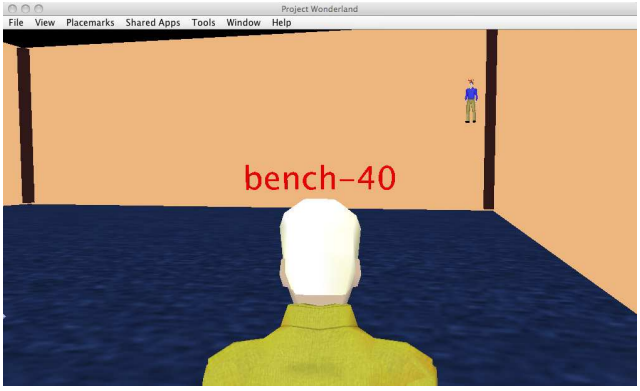

Figure 7: The avatar bench-40 looks in the direction of testRoomD as twright climbs the stairs.

from twright's perspective. Because twright is the owner of both cells, he is able to see both rooms. Figure 6, however, is from bench-40's perspective, and does not include the presence of testRoomD. This is because bench-40 is a guest user who does not belong to the admin group, and testRoomD permits neither interact nor alter access for its group or the *other* role.

Although the bench-40 participant is unable to view testRoomD, our current WonderDAC prototype stops short of concealing the avatars that enter testRoomD from those participants whose avatars cannot also enter the room. Such concealment is required to fully implement Use Case 1, and is planned future work. Figure 7 depicts what happens when twright enters into the testRoomD cell as bench-40 (an avatar that cannot enter this space) watches.

5.2 Non-spatial Object Access

To demonstrate non-spatial object access, we created a whiteboard cell in our test Wonderland environment and configured it in the same manner as testRoomC: (twright, admin, 2, 2). Both the twright and bench-40 participants were able to see the whiteboard object, including images drawn there. However, only twright was supposed to be able to change the contents, since, as the owner, twright was the only participant with alter permissions for the whiteboard object. This proved to be an interesting situation for Wonderland due to its use of publish/subscribe channels. In particular, when a participant uses a whiteboard, they are operating on a local copy maintained by their Wonderland client. Periodically, their client publishes updates of the whiteboard's state so that other clients observing/participating with the whiteboard may stay syn-

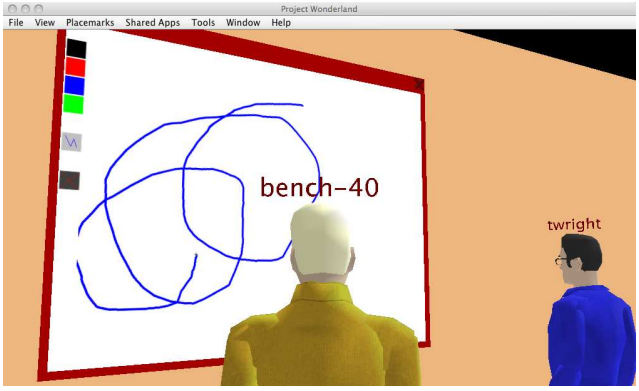

Figure 8: The bench-40 participant draws on a local copy of the whiteboard shared with twright.

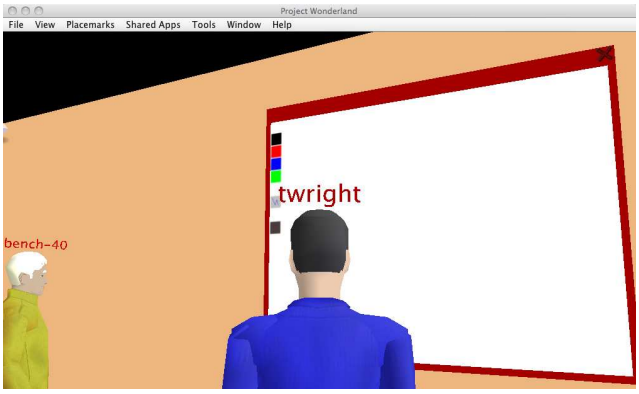

Figure 9: Because bench-40 has no alter permission, the twright participant cannot see bench-40's whiteboard updates.

chronized. If a participant lacks interact permission for the whiteboard they will, of course, be unable to see or use it in any way. If they lack just alter permission, they will still be able to draw on their local copy, even though the Wonderland server will ignore any whiteboard updates their client tries to send. Hence, there is a cosmetic issue: even though no other participants will see an unauthorized participant's whiteboard updates, the unauthorized participant can continue to use their whiteboard locally. Figures 8 and 9 illustrate this happening with twright and bench-40.

As an authorized participant uses a whiteboard, their updates will be added to the local copies of all participants—some of whom may not have alter permission, but have drawn on their local whiteboards nonetheless. The place to fix this issue is at the Wonderland client. Because there is no security risk involved (just the nuisance of a cluttered whiteboard for some participants), enhancing the client to block whiteboard use when a participant has no alter permission should be a reasonable solution. A complete implementation of Use Case 2 requires such an enhancement, and is planned for future work. We do not believe a similar circumstance exists for Wonderland's virtual phone.

6 Conclusion and Future Work

As CVEs continue their evolution into more ubiquitous, useful technologies, we find them enabling entertainment, commercial, educational, and scientific endeavors. A critical component to the success of any collaborative technology is the control of access to information and resources. For most CVEs (commercial and open source) such control is often limited in scope and not designed to fully leverage the assignment of roles to a participant. In answer to this, we have implemented a limited prototype of WonderDAC, a discretionary access control system for the Project Wonderland CVE.

WonderDAC is modeled after a classic, UNIX-style DAC system, and treats all virtual objects according to three roles: *owner*, *group*, and *other*. Interact and alter permissions are assigned to, or removed from the *group* and *other* roles to control access by CVE participants, while an object's owner always maintains full privileges. The interact permission is analogous to read/execute in a UNIX-style DAC system, while alter is similar to write. By defining five representative use case scenarios for managing CVE access control, we formulated a guide for our prototype implementation.

Although our prototype is limited to just the first two use cases (access control for spatial and non-spatial objects), it provides an important foundation for future work. We propose undertaking additional research to expand upon WonderDAC, including short-term and long-term objectives. The short-term work deals with cosmetic issues raised by the prototype: hiding avatars that reside in privileged spaces from the view of unauthorized participants, and modifying the Wonderland client to prevent unauthorized changes to a participant's local whiteboard object. The long-term work addresses the remaining use case scenarios and includes building an interface for participants to view and manage object ownership and permissions information.

Table 1: Summary of access control use cases

Use Case	Permissions	Meaning for Group Role	Meaning for Other Role
Spatial Object	I A	Any member of the object's group can enter, see, and hear into the space. All such constituents can speak within and change the space, too.	All participants who are not members of the object's group can enter, see, and hear into the space. All such constituents can speak within and change the space, too.
	I -	Any member of the object's group can enter, see, and hear into the space. All such constituents are unable to speak within and change the space.	All participants who are not members of the object's group can enter, see, and hear into the space. All such constituents are unable to speak within and change the space.
	- A	Meaningless: a member of the object's group can speak within and change the space, but they are unable to enter the space.	Meaningless: all participants who are not members of the object's group can speak within and change the space, but they are unable to enter the space.
	- -	No member of the object's group can enter, see, or hear into the space. All such constituents are unable to speak within and change the space.	All participants who are not members of the object's group cannot enter, see, or hear into the space. All such constituents are unable to speak within and change the space.
Non-spatial Object	I A	Any member of the object's group can see and, if applicable, hear the object. All such constituents can change the object, too.	All participants who are not members of the object's group can see and, if applicable, hear the object. All such constituents can change the object, too.
	I -	Any member of the object's group can see and, if applicable, hear the object. All such constituents are unable to change the object.	All participants who are not members of the object's group can see and, if applicable, hear the object. All such constituents are unable to change the object.
	- A	Meaningless: a member of the object's group can change the object, but they are unable to see or, if applicable, hear the object.	Meaningless: all participants who are not members of the object's group can change the object, but they are unable to see or, if applicable, hear the object.

Table 1: *Continued*

Use Case	Permissions	Meaning for Group Role	Meaning for Other Role
	- -	No member of the object's group can see, change, or, if applicable, hear the object.	All participants who are not members of the object's group cannot enter, see, change, or, if applicable, hear the object.
Audio Conversation	I A	All member of the conversation's temporary group can hear and speak in the conversation.	All participants who are not members of the conversation's temporary group can hear and speak in the conversation.
	I -	Not allowed: all members of the conversation's temporary group can hear the conversation, but not speak. This would prevent the conversation from taking place.	All participants who are not members of the conversation's temporary group can hear, but not speak in, the conversation.
	- A	Not allowed: all members of the conversation's temporary group can speak in, but not hear the conversation. This would prevent the conversation from taking place.	All participants who are not members of the conversation's temporary group can speak in, but not hear, the conversation. This may or may not be useful, contingent on the desires of the conversation's temporary group members.
	- -	Not allowed: no member of the conversation's temporary group can speak in or hear the conversation. This would prevent the conversation from taking place.	All participants who are not members of the object's group cannot speak in or hear the conversation.
Avatar Cloak	I A	Any member of the object's group can see the true appearance of the avatar. The alter permission is ignored: only the owner is allowed to change an avatar.	All participants who are not members of the avatar's group can see the true appearance of the avatar. The alter permission is ignored: only the owner is allowed to change an avatar.
	I -	Any member of the object's group can see the true appearance of the avatar. (Effectively the same as *I A* permissions above)	All participants who are not members of the avatar's group can see the true appearance of the avatar. (Effectively the same as *I A* permissions above)

Table 1: *Continued*

Use Case	Permissions	Meaning for Group Role	Meaning for Other Role
	- A	No member of the avatar's group sees the avatar's true image; instead, they see a generic disguise. The alter permission is ignored: only the owner is allowed to change an avatar.	All participants who are not members of the avatar's group cannot see the avatar's true image; instead, they see a generic disguise. The alter permission is ignored: only the owner is allowed to change an avatar.
	- -	No member of the avatar's group sees the avatar's true image; instead, they see a generic disguise. (Effectively the same as - *A* permissions above)	All participants who are not members of the avatar's group cannot see the avatar's true image; instead, they see a generic disguise. (Effectively the same as - *A* permissions above)

References

[Act08a] Activeworlds, Inc., *Online help manual: Shared privileges*, Found on the World Wide Web at http://www.activeworlds.com/help/aw41, April 2008.

[Act08b] ———, *Online help manual: World admin features dialog*, Found on the World Wide Web at http://www.activeworlds.com/help/aw41, April 2008.

[Act08c] ———, *Online help manual: World admin rights dialog*, Found on the World Wide Web at http://www.activeworlds.com/help/aw41, April 2008.

[BB99] Adrian Bullock and Steve Benford, *An access control framework for multi-user collaborative environments*, GROUP '99: Proceedings of the international ACM SIGGROUP conference on Supporting group work (New York, NY, USA), ACM, 1999, pp. 140–149.

[BBF98] Andreas Butz, Clifford Beshers, and Steven Feiner, *Of vampire mirrors and privacy lamps: privacy management in multi-user augmented environments*, UIST '98: Proceedings of the 11th annual ACM symposium on User interface software and technology (New York, NY, USA), ACM, 1998, pp. 171–172.

[BBPK05] Anja Le Blanc, Jonathan Bunt, Jim Petch, and Yien Kwok, *The virtual learning space: an interactive 3d environment*, Web3D '05: Proceedings of the tenth international conference on 3D Web technology (New York, NY, USA), ACM, 2005, pp. 93–102.

[Ben07] Jared E. Bendis, *Developing educational virtual worlds with game engines*, SIGGRAPH '07: ACM SIGGRAPH 2007 educators program (New York, NY, USA), ACM, 2007, p. 26.

[BK07] David A. Bray and Benn R. Konsynski, *Virtual worlds: multi-disciplinary research opportunities*, SIGMIS Database **38** (2007), no. 4, 17–25.

[BLL07] Nathaniel E. Baughman, Marc Liberatore, and Brian Neil Levine, *Cheat-proof playout for centralized and peer-to-peer gaming*, IEEE/ACM Trans. Netw. **15** (2007), no. 1, 1–13.

[Bro04] Aaron B. Brown, *Oops! coping with human error in IT systems*, Queue **2** (2004), no. 8, 34–41.

[Bro07] John Brownlee, *John edwards meets second life 'feces spewing obscenity'*, Found on the World Wide Web at http://blog.wired.com/tableofmalcontents/2007/03/john_edwards_me.html, March 2007.

[Chi97] Tzi-Cker Chiueh, *Distributed systems support for networked games*, Operating Systems, 1997., The Sixth Workshop on Hot Topics in (1997), 99–104.

[DDG+03] Stéphane Louis Dit, Samuel Degrande, Christophe Gransart, Christophe Chaillou, and Grégory Saugis, *Vrml97 distributed authoring interface*, Web3D '03: Proceeding of the eighth international conference on 3D Web technology (New York, NY, USA), ACM, 2003, pp. 135–ff.

[Dib08] Julian Dibbell, *Mutilated furries, flying phalluses: Put the blame on griefers, the sociopaths of the virtual world*, Found on the World Wide Web at http://www.wired.com/print/gaming/virtualworlds/magazine/16-02/mf_goons, January 2008.

[Fas07] Allison Fass, *Outfront: Sex, pranks and reality*, Found on the World Wide Web at http://www.forbes.com/home/free_forbes/2007/0702/048.html, July 2007.

[GA06] Sudhakar Govindavajhala and Andrew W. Appel, *Windows access control demystified*, Tech. Report TR-744-06, Department of Computer Science, Princeton University, 2006.

[GMP+02] Pedro García, Oriol Montalà, Carles Pairot, Robert Rallo, and Antonio Gómez Skarmeta, *Move:: component groupware foundations for collaborative virtual environments*, CVE '02: Proceedings of the 4th international conference on Collaborative virtual environments (New York, NY, USA), ACM, 2002, pp. 55–62.

[HD96] Steve Harrison and Paul Dourish, *Replace-ing space: the roles of place and space in collaborative systems*, CSCW '96: Proceedings of the 1996 ACM conference on Computer supported cooperative work (New York, NY, USA), ACM, 1996, pp. 67–76.

[HTK+97a] S. Honda, H. Tomioka, T. Kimura, T. Ohsama, K. Okada, and Y. Matsushita, *A virtual office environment for home office worker based on 3d virtual space*, Virtual Systems and MultiMedia, 1997. VSMM '97. Proceedings., International Conference on (10-12 Sep 1997), 38–47.

[HTK+97b] Shinkuro Honda, Hironari Tomioka, Takaaki Kimura, Takaharu Ohsawa, Kenichi Okada, and Yutaka Matsushita, *A virtual office environment based on a shared room realizing awareness space and transmitting awareness information*, UIST '97: Proceedings of the 10th annual ACM symposium on User interface software and technology (New York, NY, USA), ACM, 1997, pp. 199–207.

[Lin07a] Linden Research, Inc., *Knowledge base: Inworld issue (cooperative object editing)*, Found on the World Wide Web at https://support.secondlife.com/ics/support/security.asp, April 2007.

[Lin07b] ———, *Knowledge base: Permissions*, Found on the World Wide Web at https://support.secondlife.com/ics/support/security.asp, April 2007.

[Lin07c] ———, *Second life viewer susceptible to quicktime security flaw*, Found on the World Wide Web at http://blog.secondlife.com/2007/11/30/, November 2007.

[Lin08] ———, *Knowledge base: Land and the linden dollar economy*, Found on the World Wide Web at https://support.secondlife.com/ics/support/security.asp, February 2008.

[MAIO97] Elizabeth D. Mynatt, Annette Adler, Mizuko Ito, and Vicki L. O'Day, *Design for network communities*, CHI '97: Proceedings of the SIGCHI conference on Human factors in computing systems (New York, NY, USA), ACM, 1997, pp. 210–217.

[Mak04] Makena Technologies, Inc., *Getting around: Understanding permissions*, Found on the World Wide Web at http://info.there.com/article.php?id=105, August 2004.

[Mak06] ———, *Giving and receiving*, Found on the World Wide Web at http://info.there.com/article.php?id=076, May 2006.

[Mak08] ———, *Development lot setup manual*, Found on the World Wide Web at http://info.there.com/article.php?id=1681, March 2008.

[NPVS07] Christoph Neumann, Nicolas Prigent, Matteo Varvello, and Kyoungwon Suh, *Challenges in peer-to-peer gaming*, SIGCOMM Comput. Commun. Rev. **37** (2007), no. 1, 79–82.

[PM01] S. Pettifer and J. Marsh, *Collaborative access model for shared virtual environments*, Enabling Technologies: Infrastructure for Collaborative Enterprises, 2001. WET ICE 2001. Proceedings. Tenth IEEE International Workshops on (2001), 257–262.

[RZ04] M.J. Rissanen and X. Zheng, *On privacy of famous users in active worlds*, SICE 2004 Annual Conference **1** (2004), 35–38 vol. 1.

[Sch01] Paul Schmehl, *Barbarians at the gateway, defeating viruses in edu*, SIGUCCS '01: Proceedings of the 29th annual

ACM SIGUCCS conference on User services (New York, NY, USA), ACM, 2001, pp. 177–180.

[SMM00] F. Sato, K. Minamihata, and T. Mizuno, *A totally ordered and secure multicast protocol for distributed virtual environment*, Parallel and Distributed Systems: Workshops, Seventh International Conference on, 2000 (Oct 2000), 55–60.

[Sun08a] Sun Microsystems, *Project wonderland software architecture*, Found on the World Wide Web at http://wiki.java.net/bin/view/Javadesktop/ProjectWonderlandArchitecture, 2008.

[Sun08b] ———, *Project wonderland web site*, Found on the World Wide Web at http://lg3d-wonderland.dev.java.net, 2008.

[Sun08c] ———, *Tutorial: How to create a world using a wonderland filesystem (wfs)*, Found on the World Wide Web at http://wiki.java.net/bin/view/Javadesktop/ProjectWonderlandWFS, January 2008.

[SWJ06] Riccardo Scandariato, Bart De Win, and Wouter Joosen, *Towards a measuring framework for security properties of software*, QoP '06: Proceedings of the 2nd ACM workshop on Quality of protection (New York, NY, USA), ACM, 2006, pp. 27–30.

[Ter06] Daniel Terdiman, *Newsmaker: Virtual magnate shares secrets of success*, Found on the World Wide Web at http://news.cnet.com/2102-1043_3-6144967.html, December 2006.

[Wag07] Mitch Wagner, *Toyota under attack by second life griefers*, Found on the World Wide Web at http://www.informationweek.com/blog/main/archives/2007/04/toyota_under_at.html, April 2007.

[WP07] Jay C. Weber and Tony Parisi, *An open protocol for wide-area multi-user x3d*, Web3D '07: Proceedings of the twelfth international conference on 3D web technology (New York, NY, USA), ACM, 2007, pp. 133–136.