ORNL/TM-2013/45085

A Role-Based Access Control (RBAC) Schema for REAP

September 2013

Prepared by
H.B. Klasky, P.T. Williams, S.K. Tadinada, B.R. Bass
ORNL

ORNL/TM-2013/45085

Computational Sciences and Engineering Division

A ROLE-BASED ACCESS CONTROL SCHEMA FOR REAP

Author(s)
H.B. Klasky
P.T. Williams
S.K. Tadinada
B.R. Bass

Date Published: September 2013

Prepared by
OAK RIDGE NATIONAL LABORATORY
Oak Ridge, Tennessee 37831-6283

Managed by
UT-BATTELLE, LLC
for the
U.S. DEPARTMENT OF ENERGY
under contract DE-AC05-00OR22725

CONTENTS

Page

LIST OF FIGURES

LIST OF TABLES

Table **Page**

ABSTRACT

This document discusses various issues in implementing a robust access control schema to secure the Reactor Embrittlement Archive Project (REAP) web application. REAP application has been designed to allow different types of users. Consequently, it is vital to ensure that REAP website has a robust security framework in order to prevent intrusion, to protect data and resources from misuse, and protect data from malicious or unintentional modification by users.

Various possible alternatives for authentication frameworks in ASP.NET - Windows, Forms, Passport and custom (JOSSO) authentication frameworks are evaluated. It is concluded that Windows Authentication is apt for REAP. Various elements of implementation are discussed with examples.

1. INTRODUCTION

1.1 BACKGROUND

Exposure to neutrons in the beltline region of the reactor pressure vessel surrounding the reactor core degrades the fracture toughness of RPV steels and results in the increase of a ductile-to-brittle transition temperature (DBTT) that marks the transition between low toughness brittle and high toughness ductile fracture regimes, see Figure 1.

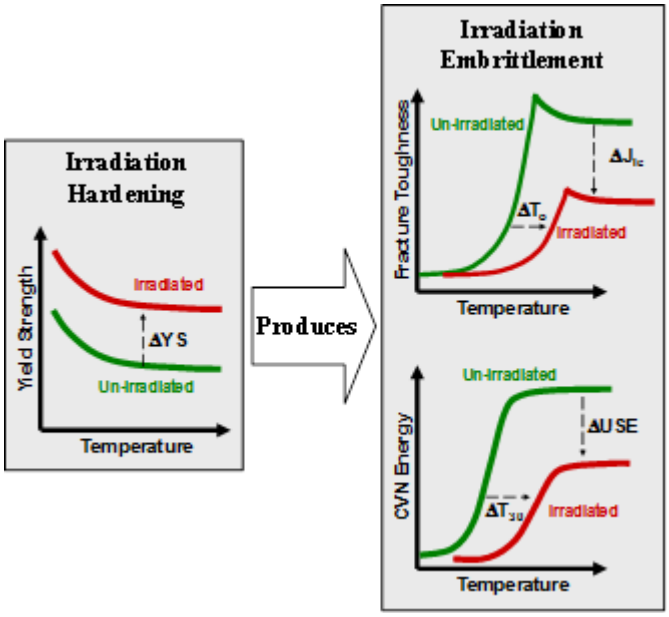

Figure 1 Effects of Neutron Embrittlement in Ferritic Steels

It is well recognized that neutron irradiation embrittlement of ferritic steels in pressure-retaining components of reactor coolant pressure boundary in light water nuclear power reactors could sometimes limit the service life of a nuclear power plant.

Within the U.S. nuclear regulatory framework, Title 10 of the Code of Federal Regulations Part 50, "Domestic Licensing of Production and Utilization Facilities,", Appendix H, "Reactor Vessel Material Surveillance Program Requirements" stipulates a material surveillance program required to monitor changes in the fracture toughness properties of ferritic materials in the reactor vessel beltline region of light water nuclear power reactors which result from exposure of these materials to neutron

irradiation and the thermal environment. Under the program, fracture toughness tests are performed on material specimens exposed in surveillance capsules which are withdrawn periodically from the reactor vessel. The fracture toughness test data is compared against the fracture toughness requirements described in Section IV of Appendix G to Part 50 in Title 10 CFR in order to ascertain if the adequate margins of safety are provided during any normal operation, including anticipated operational occurrences and system hydrostatic tests, to which the pressure boundary may be subjected over its service lifetime. These surveillance reports and data form crucial elements in development of predictive models of Neutron Irradiation Embrittlement for light water reactor (LWR) RPVs.

The Reactor Embrittlement Archive Project (REAP) [10] conducted by the Oak Ridge National Laboratory (ORNL) under funding from the United States Nuclear Regulatory Commission's (NRC) Office of Nuclear Regulatory Research, aims to provide a web-based archival source of information concerning the effects of neutron radiation on the properties of reactor pressure vessel steels. The REAP website is designed to provide access to information in two forms:

1. A **Document Archive**, which provides access to files in PDF format of original source documents (e.g., technical reports), and
2. A **Data Archive**, which provides access, in a relational database format, to information extracted from the document archive.

The initial release of REAP focuses on data collected as part of surveillance programs for light-water, moderated, nuclear power reactor vessels operated in the United States. This includes data on Charpy V-notch energy, tensile properties, composition, exposure temperature, flux, and fluence. Additionally, REAP contains some data from surveillance programs conducted in other countries.

REAP application hosts archives of data from multiple sources and it is expected that the web application will allow several types of users work concurrently. Consequently, it is vital to ensure that REAP website has a robust security framework in order to prevent intrusion, to protect data and resources from misuse, and protect files from malicious or unintentional modification by users.

This document details a proposal of security features for the REAP web application. Various possible alternatives for authentication frameworks in ASP.NET - Windows, Forms, Passport and custom (JOSSO) authentication frameworks are evaluated. It is concluded that Windows Authentication is apt for REAP. Various elements of implementation are discussed with examples.

1.2 BASICS OF WEB APPLICATION SECURITY

Security is a critical part of any web-application in order to ensure that it performs reliably in all user-environments and is not susceptible to malicious attacks that may result in loss of important information. Web applications allow users access to a central resource — the Web server — and through it, to others such as database servers. Figure 2 presents a schematic representation of the REAP web-application's data flow.

The three basic standards for information security of any web application are sometimes referred to as the "C-I-A triad" [1,2] :

1. **Confidentiality** (C) - prevent the disclosure of information to unauthorized individuals/systems
2. **Integrity** (I) - protect data from modification or deletion by unauthorized parties
3. **Availability** (A) - ensure reliable functioning of systems, access channels, and authentication mechanisms so that the information they provide and protect is available when needed.

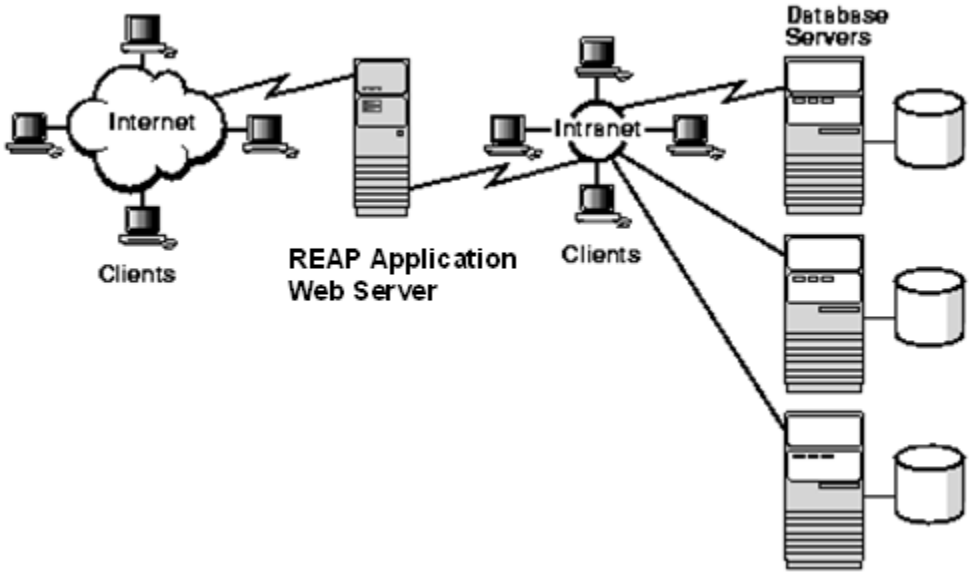

Figure 2 Data Flow in REAP Web Application

Of the many aspects of security like authentication, authorization, data privacy, data integrity etc., the fundamental security concepts are:

- **Authentication** is the process of identifying and verifying a client seeking to use the application. It confirms that users are who they say they are.
- **Authorization** is the process of determining if a particular client is cleared against accessing specific information on the server.

1.2.1 Access Control

A security model that implements the Authentication and Authorization policy is called an **Access Control model**. There are various access control models [3]:

a. Discretionary Access Control
b. Mandatory Access Control
c. Role-based Access Control

Of the three access control models, the most widely adopted access control for commercial and non-military web applications is the Role-based Access control model [4]. Figure 3 delineates the difference between the traditional and the role-based access control models [5].

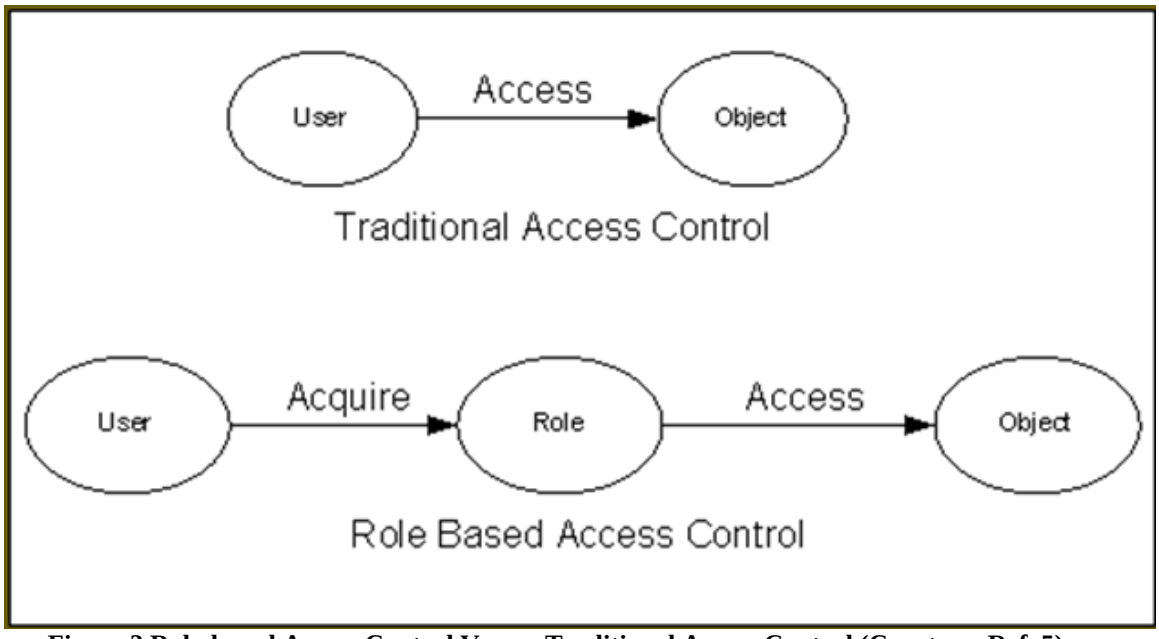

Figure 3 Role-based Access Control Versus Traditional Access Control (Courtesy: Ref. 5)

For complex web-applications with a large number of anticipated users, it is unwieldy to use traditional methods of access control, which require defining a large number of individual access policies or rules for each specific user. Instead, the Role Based Access Control (RBAC) model works by grouping similar kind of users into various "user roles". And the "permission" to access specific information is assigned to the designated user-roles. This introduction of "Roles" and "Permissions" facilitates easy description of complex access control policies while reducing administrative errors and costs.

Based on the discussion above, it can be seen that the Role-Based Access Control model is a flexible, scalable and appropriate security model and is therefore adopted for the REAP web application as well.

2. DATA SECURITY REQUIREMENTS FOR REAP WEB APPLICATION

In order to implement the Role-Based Access Control (RBAC) schema for the REAP application, we need to first do the following:

(1) Identify all possible actions over the archival data in the embrittlement database
(2) Categorize expected users into “user roles”
(3) Assign permissions for each action in (1) for each user role defined in (2).

2.1 ACTION LIST

2.1.1 Actions on the Embrittlement Database

The embrittlement data on REAP is stored as a relational database organized as a collection of the following data-tables:

Table 1 Descriptions of various database tables in REAP

TABLE	DESCRIPTION
Citation	References to the surveillance reports
Plant	Power reactor nuclear plants
Capsule	The containment vessel within a nuclear power plant
Material	Information about the material of which a plant's capsule is made
Chemistry	Information about the chemical composition of a material
Heat Treatment	Information about the process of subjecting a material to a cycle of heating and cooling to change the metallurgy of the material.
Charpy Specimens	Information about a Charpy test that is performed on a material
Tensile Specimens	Information about a tensile test that is performed on a material

The REAP database model design is reproduced from Ref. [10] in Figure 4:

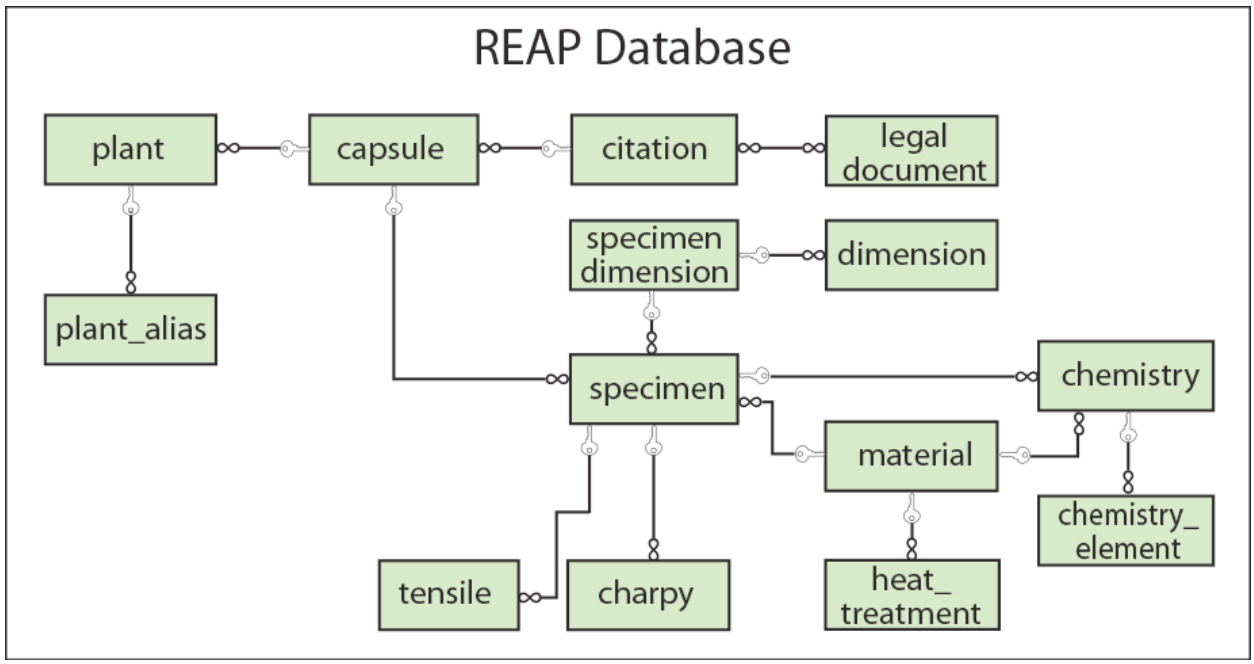

Figure 4 REAP Database Model Design

The various possible actions on these data-tables include the CRUD [11] and are listed below:

1. Create (C)
2. Delete (D)
3. Edit (E)
4. Read (R)
5. Use (U)

All users may be allowed some or none of the above actions on each of the data-tables.

2.1.2 Generic Actions

Additionally, the following generic / system-level actions may be possible:

1. Login
2. Logout
3. Registering
4. Searching through the website
5. Reporting

2.2 ROLES

Based on the list of all possible actions listed in Section 2.1, users on the REAP application may be grouped into three "user-roles":

1. Admin – Embrittlement Web Application Administrator
2. Checker – Data checker
3. Viewer

2.3 PERMISSIONS

Having defined clearly the set of possible actions and the user roles, we can now proceed to assign the permissions of each user role to perform a particular action. This activity is presented in Table 2.

Table 2 Permissions of each user-role to perform a specified action

Actions on the REAP Database			
Tables	**Roles**		
	Admin	Checker	Viewer
Citations	X	CER	U
Plant/ plant_alias	X	CER	U
Capsules	X	CER	U
Materials	X	CER	U
Specimens (Charpy/Tensile)	X	CER	U
Heat Treatment	X	CER	U
Chemistry	X	CER	U

Generic Actions			
Roles ->	Admin	Checker	Viewer
Login	U	U	U
Logout	U	U	U
Registering	U	U	N
Searching	U	U	U
Reporting	U	U	U

Legend	
Privileges:	
ALL	X
CREATE	C
EDIT	E
READ	R
USE	U
NONE	N

2.4 SPECIFIC REQUIREMENTS

Having defined the basic requirements for implementing the RBAC schema, the specific requirements of the REAP application are summarized as follows:

- The identity of every user on REAP must be authenticated on the ORNL server upon verifying the credentials supplied through the XCAMS/UCAMS login page.
- Upon successful login, a session is created and the authenticated user is designated his pre-specified user-role.
- The website shall always render views and display only those links as applicable to the user-role of the authenticated user.
- The web application shall accept only authorized HTTP requests as applicable to the user-role of the authenticated user.
- The web application shall allow only authorized users to perform allowable actions on the data such as Create, Read, Use, Delete and Edit in the various sections of the embrittlement database

- Masking proprietary data to the degree required by the contributor (that is, in some cases just obscuring the reactor that the data came from, or in other cases, obscuring the entire data record)
- The website shall automatically end the user's session and log out the user if inactivity is detected for more than a stipulated time (say 30 minutes).

3. CURRENT DESIGN

3.1 DESCRIPTION OF EXISTING APPROACH FOR REAP'S SECURITY

3.1.1 Databases

Currently three tables in the REAP database are used to store the various information regarding authentication and authorized roles for each user:

(i) **xcams_person**: A list of all the authenticated users and their information including user names first name , last name, email, affiliation, address, phone etc.

(ii) **user_role**: A list of all user roles. Currently, there are three roles: admin, checker, viewer1

(iii) **role**: This table maps each user in **xcams_person** to all the roles in **user_role** that he or she is authorized for.

3.1.2 Handling permissions to actions / viewing information.

When a user tries to execute a particular action by calling on a function or tries to access some information on the server, the first step is to make sure that he is allowed to do so. REAP does this by querying the database first and obtaining the current role for the user:

```
public class XYZController : DefaultController
    {
        private NRC_PREDBEntities db = new NRC_PREDBEntities();


        public ActionResult someAction()
        {
            ViewBag.role = (string)TempData.Peek("user_role");

            return View();
        }
```

The `(string)TempData.Peek("user_role")` peeks into the database and returns the role of the user. The `view()` for the action uses the `ViewBag` variable `ViewBag.role` to display the appropriate content on the browser. For Example:

```
@if (ViewBag.role == "checker" || (ViewBag.role == "admin"))
{
    <li style="font-size: 0.80em;">@Html.ActionLink("How to Upload
    Citations", "HowToUploadCitations", "UserGuides", null, new {
    title = "Get help on how to upload citations." })</li>

    <li style="font-size: 0.80em;">@Html.ActionLink("How to Mine
    Data", "HowToMineData", "UserGuides", null, new { title = "Get
    help on how to upload data." })</li>

}
```

In the above example, using a IF control statement of `@if (ViewBag.role == "checker" || (ViewBag.role == "admin"))`, the `view()` checks first if the user is either a "checker" or an "admin" and only then, proceeds to display the content.

3.2 LIMITATIONS OF EXISTING DESIGN AND NEED FOR A BETTER ONE?

As we have seen, while REAP implements a fairly intuitive and easily implementable approach to assign roles and permissions, there exists two important limitations for this design:

- Simple, straight-forward querying through the database to obtain the role of the current user may be inefficient and may have increased processing time especially if the database is large.
- Also currently, every action method independently queries the database each time a user decides to perform a function on REAP. This leads to duplication of the same query a large number of times in the same session and can slow down the efficiency of the application.

In view of these limitations, we now try to explore a more robust access control schema that addresses the above limitations.

4. TYPES OF AVAILABLE ACCESS CONTROL FRAMEWORKS COMPATIBLE WITH ASP.NET

4.1 AUTHENTICATION

REAP hosting a database with specialized information is not deemed for open public access and thus cannot allow anonymous visitors. Every user visiting the REAP must be authenticated i.e. the application must first recognize and identify each user trying to access information on the website. Authenticating a user requires evidence, also known as credentials. An important consideration in authentication is to decide what kind of credentials to accept. For example, credentials may be in form of a password, thumbprints etc.

REAP is built using the ASP.NET MVC web application framework. The various authentication modes supported by ASP.NET include:

1. Windows Authentication
2. Forms Authentication
3. Passport Authentication
4. Custom Authentication like JOSSO etc.

We now present the details of an initial evaluation conducted to compare various authentication modes in order to decide an efficient authentication mechanism for REAP application.

4.1.1 **Windows Authentication** [6]

- Can be coupled with IIS authentication so that you don't have to write any custom code.
- The username and password are never sent over the network - they are held by the browser and used to answer challenges from the remote web server.
- No firewall issues is user accounts are maintained by a Microsoft® Windows NT domain controller or within Microsoft Windows Active Directory

4.1.2 Forms Authentication

- Requires additional coding for implementing a detailed AccountsController to specify functions like:
  - Logon
  - Register
  - Password retrieval/modify etc.
- Requires attention to firewall issues
- Ensuring the integrity and privacy of data as it flows across public and internal networks

4.1.3 Passport Authentication

Passport authentication relies on a centralized service provided by Microsoft. Passport authentication identifies a user with using his or her e-mail address and a password and a single Passport account can be used with many different Web sites. Passport authentication is primarily used for public Web sites with thousands of users. Since REAP is a stand-alone web application, this type of authentication is not optimal.

4.1.4 Custom Authentication: JOSSO

JOSSO, or Java Open Single Sign-On, is an open source Internet SSO solution for rapid and standards-based Internet-scale Single Sign-On implementations, allowing secure Internet access to the Web-based applications or services of customers, suppliers, and business partners [7].

- Extremely useful when the same user must be granted access to multiple applications and services that are related (but independent) with a Single Sign On (SSO) rather than multiple authentications
- Requires extensive implementation of custom controllers pertaining to managing user accounts, secured flow of data between client, application and the SSO service.
- Requires attention to firewall issues
- Ensuring the integrity and privacy of data as it flows across public and internal networks

4.1.5 Framework Authentication Selection

Based on this preliminary evaluation, it can be seen that the compared to other authentication mechanisms, Windows authentication does not pass the user credentials over the wire, requires no custom code for coupling with IIS authentication. Moreover, REAP is hosted at an ORNL-based server which requires the user by default to register for a XCAMS/UCAMS account for accessing any of the web pages on ORNL's intranet. This means that implementing the Microsoft Windows Authentication is not only easy but also effects a seamless transition to the new authentication framework.

4.2 AUTHORIZATION

Authorization verifies if a user is allowed to do what he wants to do with respect to the website. It is not desirable that not every authenticated user be allowed unlimited access to all the data on the REAP application. We may want certain clients to have full access allowing them to create or modify data on the server, others may only be allowed to access a subset of the data, and others may have read-only access while some other clients may be granted only a "read-only" access to data. These permissions are typically managed by defining various user-roles. A "user-role" determines the access controls for all authenticated users belonging to this "user-role".

ASP.NET's `System.Web.Security` namespace provides simple, direct methods pertaining to role management in order to manage authorization easily and allow us to specify resources that users of a web-application are allowed to access. Adopting the role management classes provided with the ASP.NET framework for the REAP application is deemed to be efficient and requires less effort. More details on role management in ASP.NET applications may be found in references [8,9].

5. IMPLEMENTATION THE WINDOWS AUTHENTICATION FRAMEWORK ON REAP

Implementing the windows authentication mode on REAP is very simple. Since the Log On with Windows Authentication is handled outside of the web application, there is no need to implement a separate controller to manage user accounts/logins as in Forms or Custom type authentication modes. To configure Windows Authentication, the following line must be included in `web.config` [6]:

```
<authentication mode="Windows" />
```

This can be performed from the security tab of the ASP.NET website administration tool provided along with the Microsoft Visual Studio (See Figure 5):

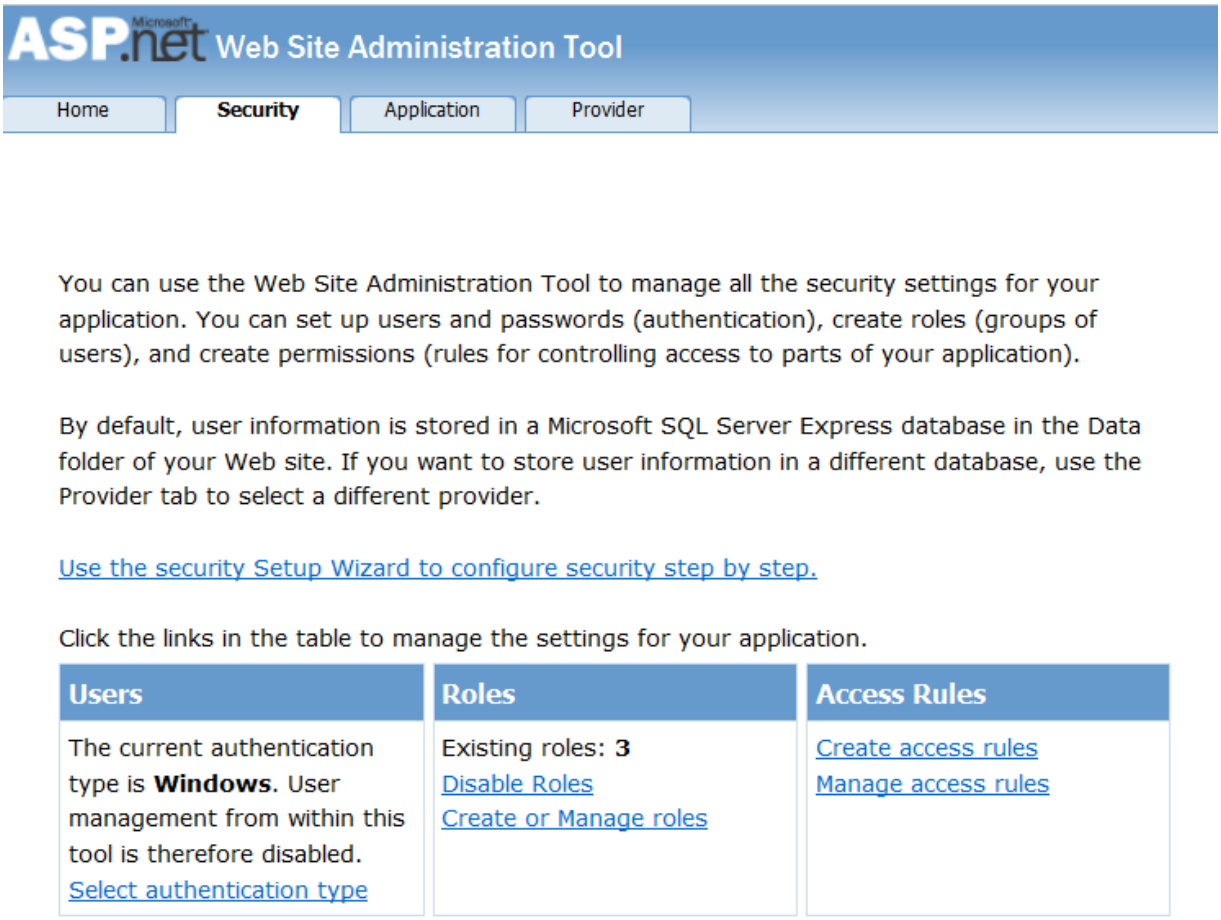

Figure 5 Screenshot of the security tab of the ASP.NET website administration tool

5.1 AUTHORIZATION

Authorization verifies if a user is allowed to do what he wants to do with respect to the website. It is not desirable that not every authenticated user be allowed unlimited access to all the data on the REAP application. We may want certain clients to have full access allowing them to create or modify data on the server, others may only be allowed to access a subset of the data, and others may have read-only access while some other clients may be granted only a "read-only" access to data. These permissions are typically managed by defining various user-roles. A "user-role" determines the access controls for all authenticated users belonging to this "user-role".

In REAP, once a user's identity is authenticated through the XCAMS/UCAMS on the REAP server, he is authorized under any one of the three user-roles: `admin, checker, viewer1.` These roles can be created and managed using the ASP.NET website administration tool (See Fig. 6)

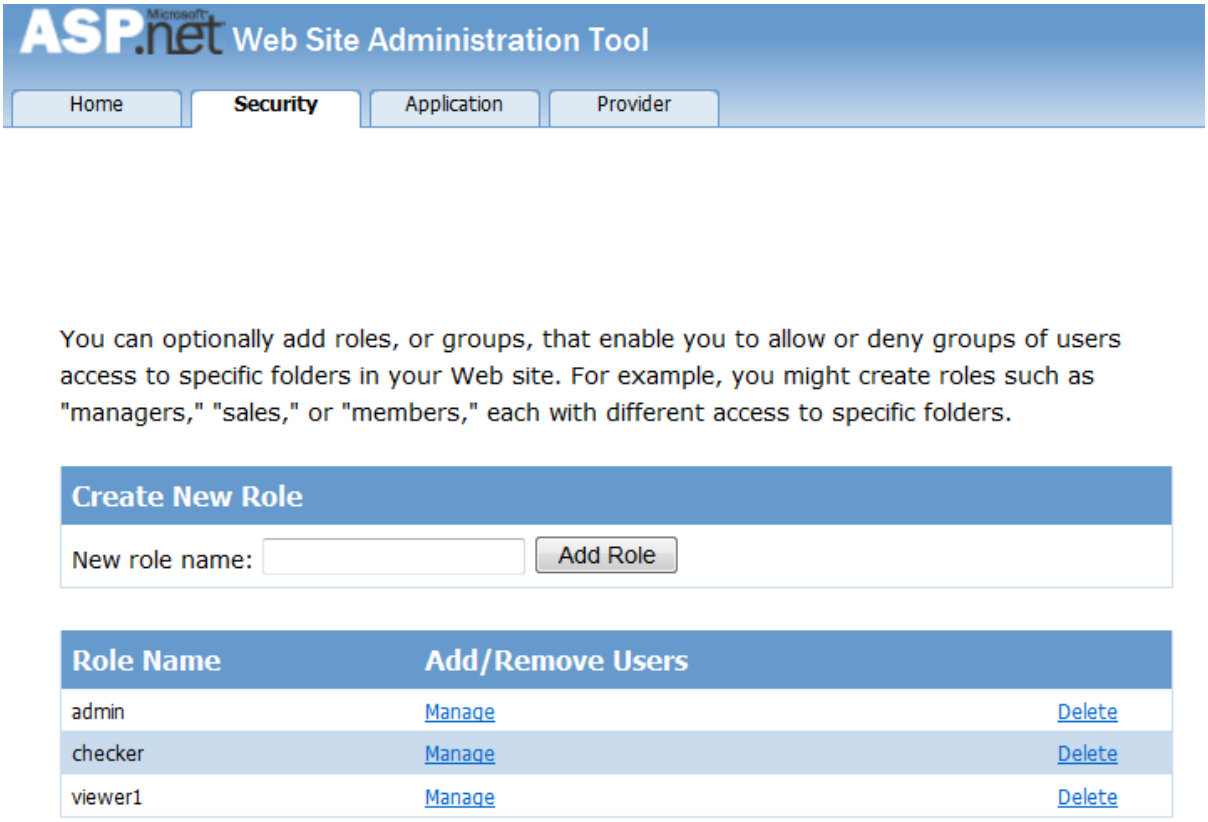

Figure 6 Screenshot of the role management page in the ASP.NET website administration tool

Having defined user-roles, the access permissions can be configured for specifying:

a) Custom access
b) Custom views

5.2 EXAMPLE: CUSTOM ACCESS

Authentication and authorization in ASP.NET are built on top of the Role and Membership classes found in the `System.Web.Security` namespace. For example, if want a specific page to

be accessible only by a user in the user-role "checker", we can use the "Authorize" attribute to limit access to the page checkerSecrets by:

```
        [Authorize(Roles = "Checker")]
        public ActionResult checkerSecrets()
        {
            ViewBag.Message = "CHECKER'S SECRET. YOU CAN WATCH THIS
ONLY IF YOU ARE AN CHECKER";
            return View();
        }
```

5.3 EXAMPLE: CUSTOM VIEW

In this case, the text " I am an admin" is displayed on a page if the user is in user-role "admin". This can be executed by modifying the view page by including:

```
        @if (Roles.IsUserInRole("admin")) {
            <li>
            <h3> I am an admin </h3>
            </li>
        }
```

Figure 7 An example screenshot depicting customizable view in the authorization framework

6. CONCLUSION

The goal of this exercise is to implement a robust security framework for authentication and authorization of clients using the REAP web application. This section presented:

- A brief preliminary evaluation of various modes of authentication in ASP.NET
- Implementing the features of Windows Authentication mode to REAP application

- Defining Authorization roles and permissions in the REAP application
    - 3 user-role types were defined : `admin, checker, viewer1`
- Examples to demonstrate using the authorization attributes to control access depending on the specified user-role

6.1 FUTURE WORK

Preliminary testing of the features based on Windows Authentication framework are presented thus far. In order to migrate the security framework for the REAP application to Windows Authentication Framework, the following future tasks are proposed:

- Review user roles and permissions table.
- Map permissions to REAP screen actions.
- Identify lines of code to modify to include Windows Authentication Framework calls
- Implementation and testing
- Implement a mechanism to log out users when the session times out.

7. REFERENCES

1. http://en.wikipedia.org/wiki/Information_security#Basic_principles
2. http://docs.oracle.com/cd/A97630_01/network.920/a96582/overview.htm
3. http://en.wikipedia.org/wiki/Access_control#Access_control_models
4. http://csrc.nist.gov/groups/SNS/rbac/
5. Rajput, S. and Cherukuri, R., "Role-Based Access Control Models", Available at www.cse.fau.edu/~security/public/RBAC_present.ppt
6. http://msdn.microsoft.com/en-us/library/ff647405.aspx
7. http://www.atricore.com/products/josso.html
8. http://msdn.microsoft.com/en-us/library/5k850zwb(v=vs.85).aspx
9. http://msdn.microsoft.com/en-us/library/t32yf0a9(v=vs.100).aspx
10. Klasky, H.B., Bass B.R., Williams, P.T., Phillips, R.D., Erickson, M., Kirk, M.T., Stevens, G.L., “Radiation Embrittlement Archive Project,” Transactions, SMiRT-22, San Francisco, California, USA, 2013.
11. Heller, M., "REST and CRUD: the Impedance Mismatch". *Developer World*. InfoWorld, 29 January 2007. Available at: http://www.infoworld.com/d/developer-world/rest-and-crud-impedance-mismatch-927