Role-Based Access Control

lesson #lesson10

James L. Parry
B.C. Institute of Technology

Role-Based Access Control...

Authentication usually refers to the strategies and techniques used to ensure that only authorized users access sensitive functionality in a webapp.

CodeIgniter purposefully does not include provision for authentication, relying instead on third-party packages.

Role-Based Access Control is the most common technique to secure access to different parts of a webapp.

Agenda

THE THEORY

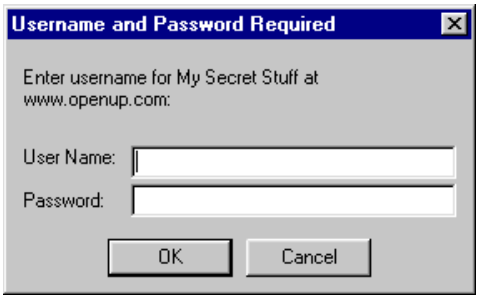
Security normally involves four facets:

- Authentication - who is this?
- Authorization - what can they do?
- Access - enforcing the above
- Audit - what did they do?

Two Security Models

Identity authorities

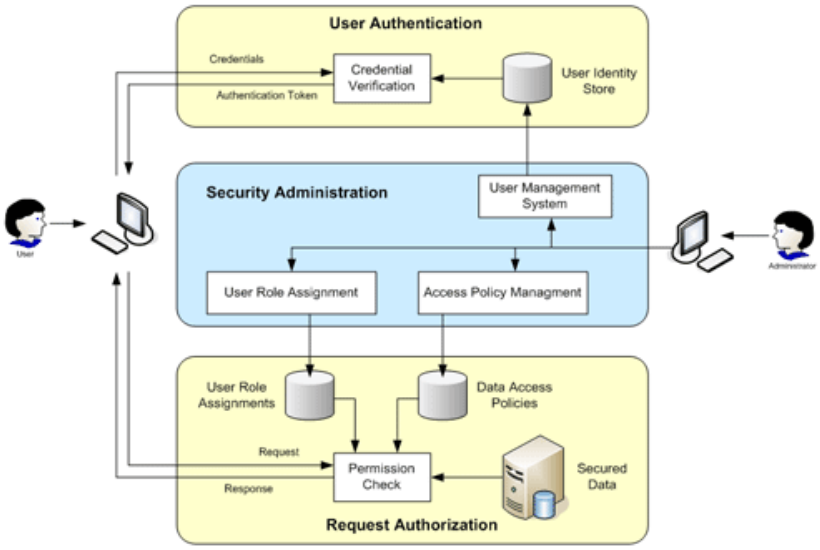
- Operating system
- Active Directory
- Certificate authority
- OpenID



Federations of trust

- Kerberos
- OAuth (Facebook, Google)
- OpenID ...

Typical Framework

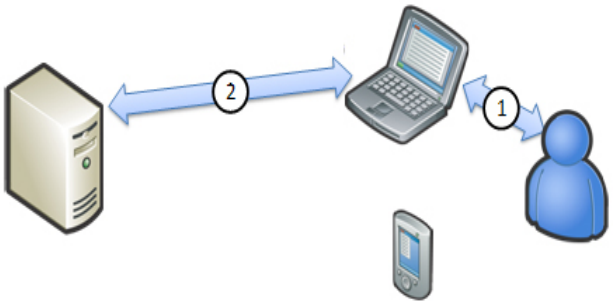


THE PRACTICE

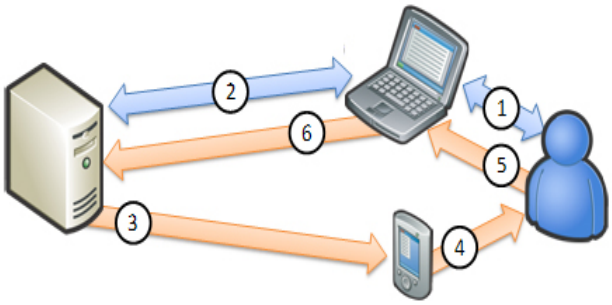
- Authenticate somehow
- Extract authorization from the user object or policy configuration
- Enforce roles on access
- Log as needed

Authentication Choices

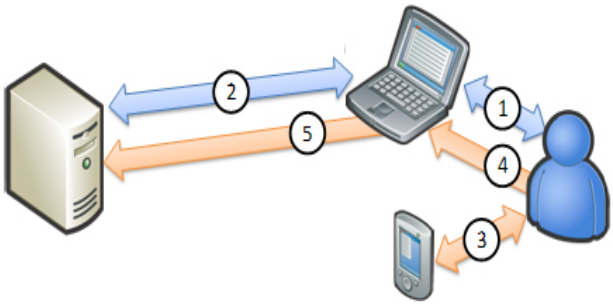
Knowledge-based



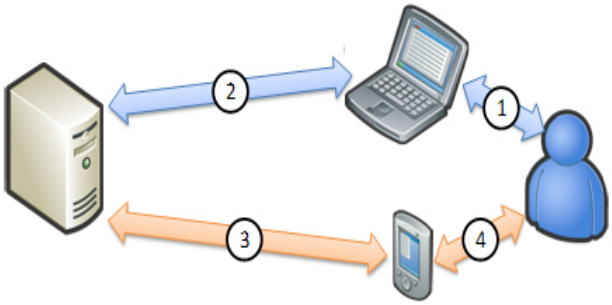
Server-Generated OTP



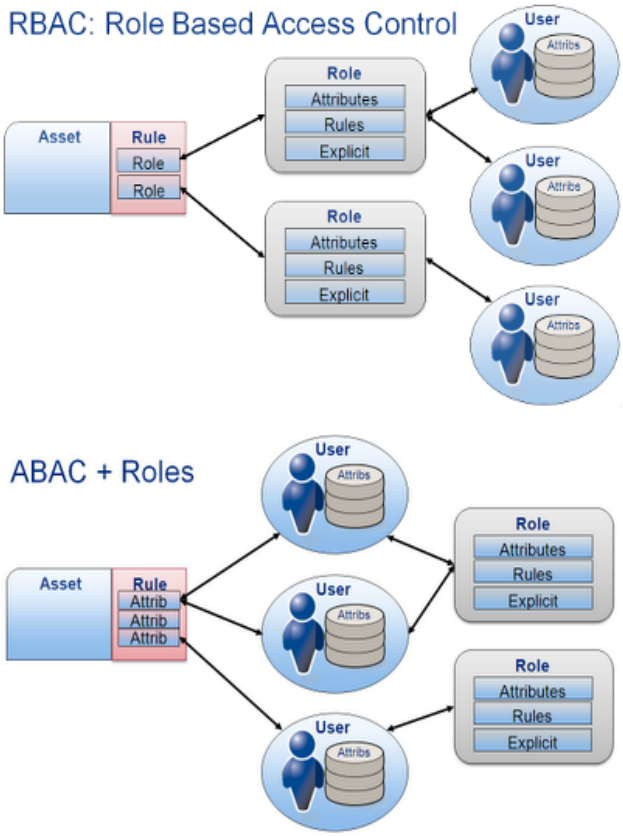
Client-generated OTP

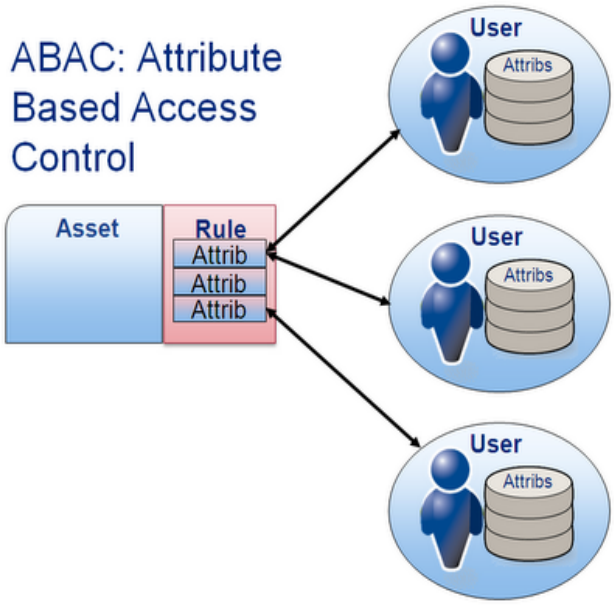


Out-of-band



Authorization Choices





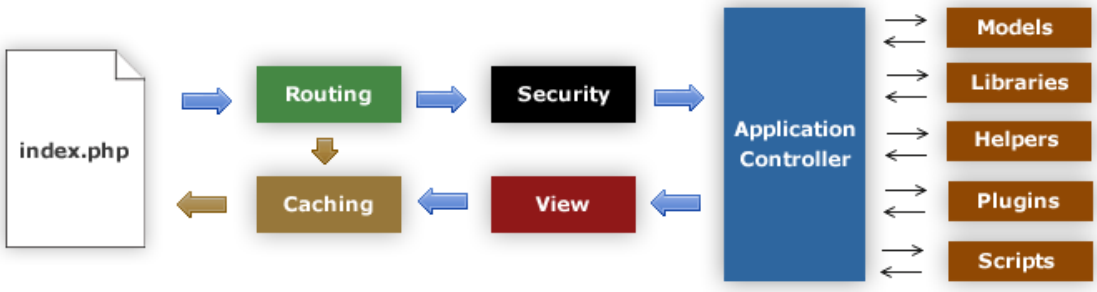
Boils down to access control list (ACL) or user capability list (UCL)

Access Control Choices

- Declarative (stored as policy or configuration data) vs Programmatic (rules coded into access control points in components)
- Hook (intercept request and redirect if inappropriate) vs Explicit (make decision as part of request handler)
- User & authorization data in session - so you don't have to revalidate continuously

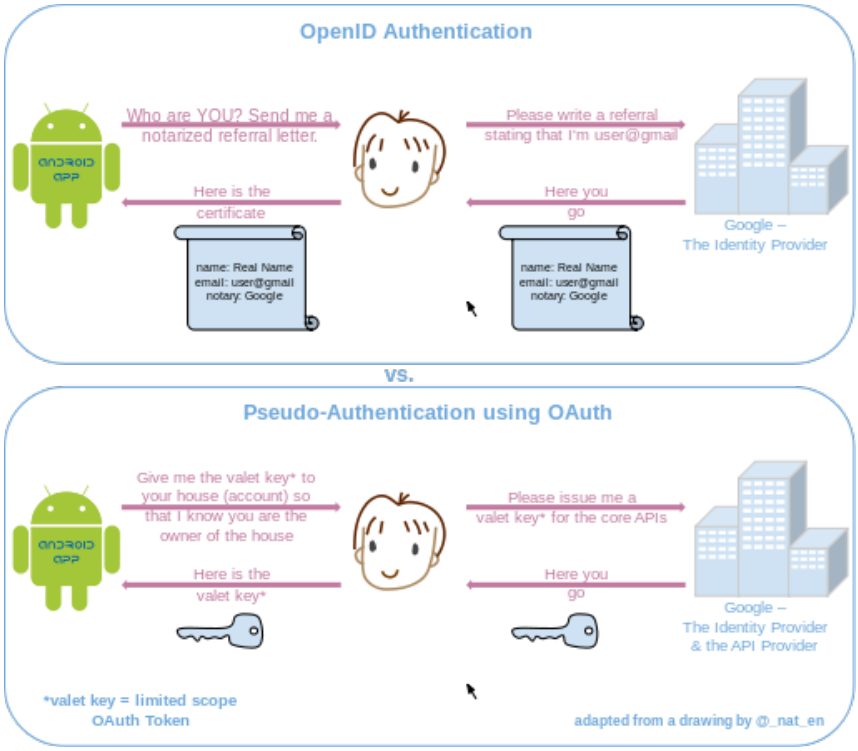
CODEIGNITER CHOICES

- Hooks - pre-controller, as part of routing
- 3rd party libraries or packages
- Roll your own - as part of base controller



Popular Technology Choices

Third party tool which does one or both of...



CODEIGNITER AUTHENTICATION

"Roll your own"...

We'll start with:

- Users table
- Login form, with controller or subcontroller
- Logout controller

Things to keep in mind:

- Don't store password in plaintext! (You will be fired)
- Add users model to config/autoload?

Users Table

SQL to create your table

```
CREATE TABLE IF NOT EXISTS `users` (
`id` varchar(10) NOT NULL,
`name` varchar(20) NOT NULL,
`password` varchar(64) NOT NULL,
`role` varchar(20) NOT NULL,
PRIMARY KEY (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

Your model:

```
class Users extends MY_Model
```

You can add more fields to the table, and you might want validation logic in your model.

Passwords are stored hashed (you might need to increase the length of the password field) using... hash(ALGO,$plaintext), password_hash($plaintext,ALGO) (for PHP 5.5+), or crypt($plaintext). See the password hashing FAQ.

Login controller

Shown below is the gist of login, validating a supplied password. If valid, some user data is stored in the session. The controller is not complete ... it needs error checking, redirection, login form display, and so on.

```
class Login extends Application {
  function submit() {
    $key = $_POST['userid'];
    $password = password_hash($_POST['password'],PASSWORD_DEFAULT);
    $user = $this->users->get($key);
    if ($password == (string) $user->password) {
      $this->session->set_userdata('userID',$key);
      $this->session->set_userdata('userName',$user->name);
      $this->session->set_userdata('userRole',$user->role);
    }
  }
}
```

Logout Controller

Shown below is the gist of your logout handling.

```
class Logout extends Application {
  function index() {
    $this->session->sess_destroy();
    $this->load->helper('url');
    redirect('/');
  }
}
```

Registration Notes

Good practices:

- Form based
- Avoid barriers to entry
- Defeating bots... see right
- Email verification
- Token field(s) in users table?

Bot detection:

- Use the CI User Agent ...
  `$this->load->library('user_agent');`
  `if ($this->agent->is_robot()) ...`
- Use a “comment” field in your HTML form, styled with display:none
  `if (isset($_POST['comment']) && strlen($_POST['comment']) > 0) ...`
- Captcha?

CODEIGNITER AUTHORIZATION

Use user roles for authorization.

Extract the role from the User object, or determine it using other rules.

Store the user role in the session.

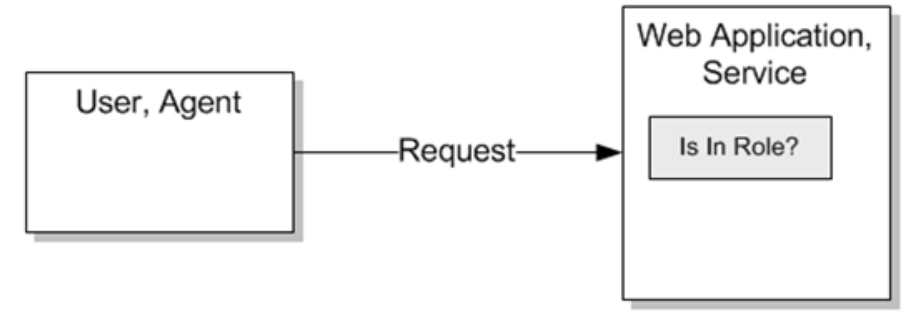
Some suggested roles are shown to the right

- Guest - not logged in
- Visitor - logged in using social login (you know they are human, you just don't know who); maybe they can post comments
- User - someone registered on your site, and who logged in conventionally (you know who they are); maybe they can post or update their own entries and comments
- Admin - "user" that you trust more than others; maybe they can post, update or delete anyone's entries or comments

CI ACCESS CONTROL

Restrict access based on role

Do this per controller



Access Control Pseudocode

Shown below is the gist of your access control enforcement.

```
if role needed
  if not logged in
    redirect to login page
if user has that role
  allow request to proceed
  else redirect to landing page, maybe with error message(s)
```

Enforcement in Base Controller

```
class Application extends CI_Controller {
  ...
  function restrict($roleNeeded=null) {
    $userRole = $this->session->userdata('userRole');
    if ($roleNeeded != null) {
      if (is_array($roleNeeded)) {
        if (!in_array($userRole, $roleNeeded)) {
          redirect("/"); return;
        }
      } else
        if ($userRole != $roleNeeded) {
          redirect("/"); return;
      }
    }
  }
}
```

Applying This

```
class Whatever extends Application {
  function __construct() {
    parent::__construct();
    $this->restrict(ROLE_USER);
  }
...
}

class Admin extends Application {
  function __construct() {
    parent::__construct();
    $this->restrict(array(ROLE_ADMIN,ROLE_WEBMASTER));
  }
...
}
```

ALTERNATIVES FOR CI

Food for thought... in increasing order of difficulty

- Social media integration - make it easy for your users to share, like or tweet about specific content or pages, whether they use social login or not
- Social Login - allow users to login with their Facebook or Google+ account
- Social media management? - an example: if you or a user post an image to Instagram, repost that on "your" Facebook wall, with caption or comments, and then tweet it to your faithful followers

Congratulations!

You have completed lesson #lesson10: Role-Based Access Control

If you would take a minute to provide some feedback, we would appreciate it!

The next activity in sequence is: tutorial08 Authentication

You can use your browser's back button to return to the page you were on before starting this activity, or you can jump directly to the course homepage, organizer, or reference page.