



ibuildings [i]
THE PHP PROFESSIONALS

Practical Applications of Zend_Acl

Rowan Merewood

Who is this?

@rowan_m

Software Engineer
Team Lead

<http://merewood.org>

Why do this?

So you don't have to.

Problems encountered,
solutions discovered,
lessons learned.

What do you want?

More concept?

- or -

More code?

What does this solve?

The “Gold Standard” for
security.

Gold
79
Authentication
196.97

Gold
79
Authorisation
196.97

Gold
79
Auditng
196.97

B.W. Lampson. Computer Security in the Real World. Computer, 37(6):37–46, 2004
<http://research.microsoft.com/en-us/um/people/blampson/69-SecurityRealIEEE/69-SecurityRealIEEE.htm>

Gold
79
Authentication
196.97

Gold
79
Authorisation
196.97

Gold
79
Auditng
196.97



You are here.

Role-based Access Control

Roles

Resources

Privileges

Assertions

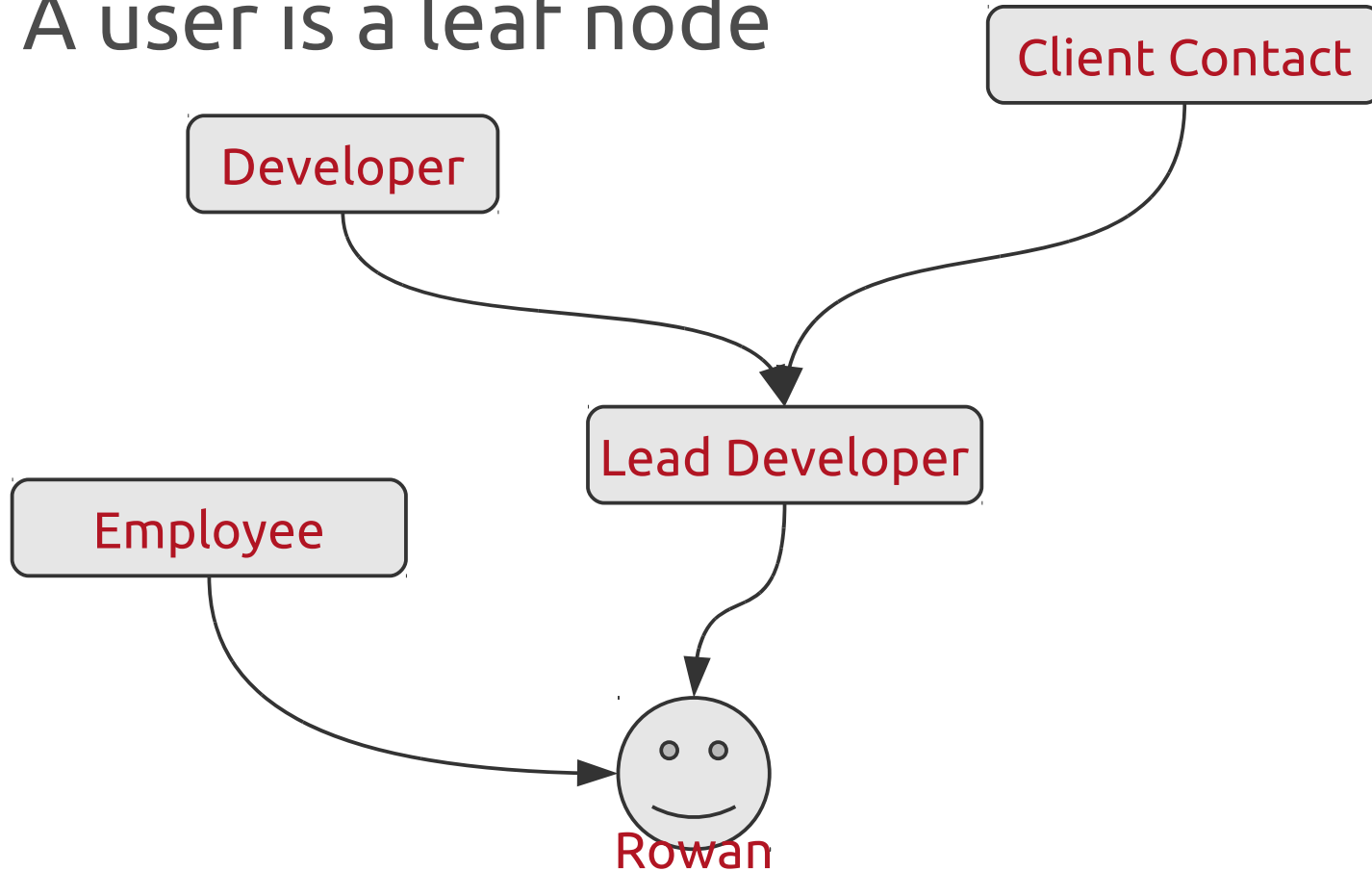
Oh my!

Warning!

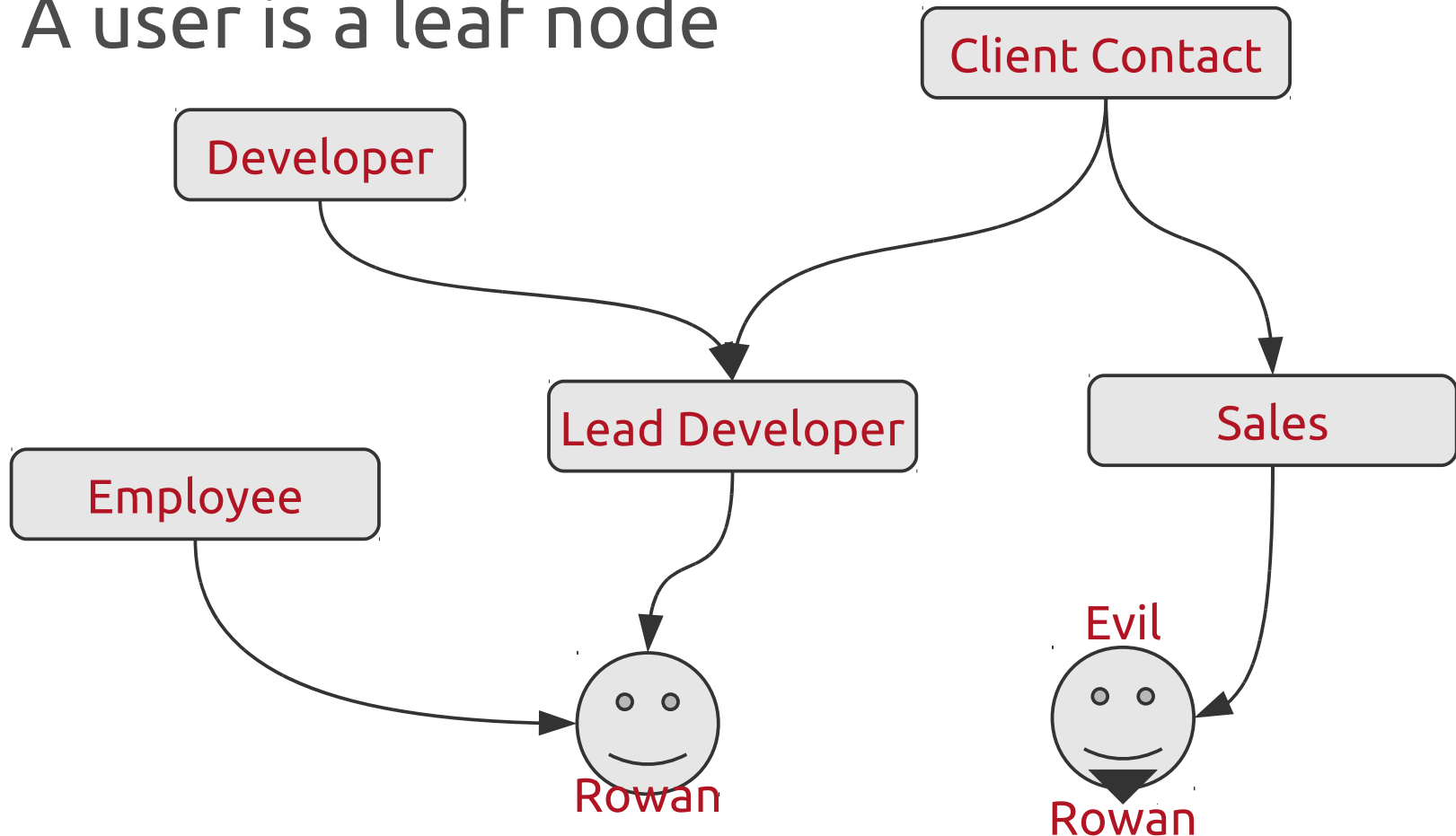
There is no right answer

- A named group of privileges for a resource.
- A role may inherit from many parent roles
- Build the tree from leaf to root

- A user is a leaf node

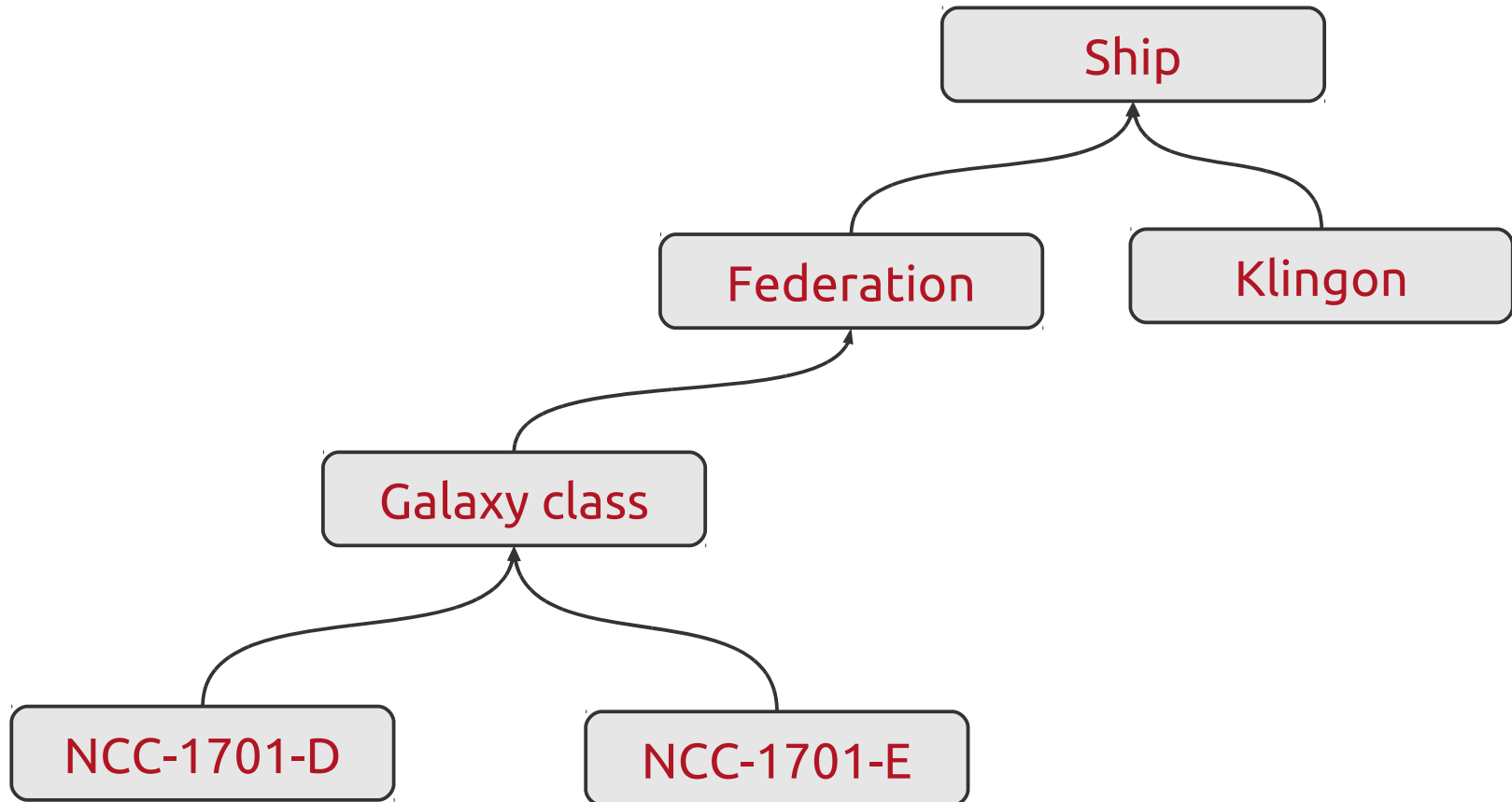


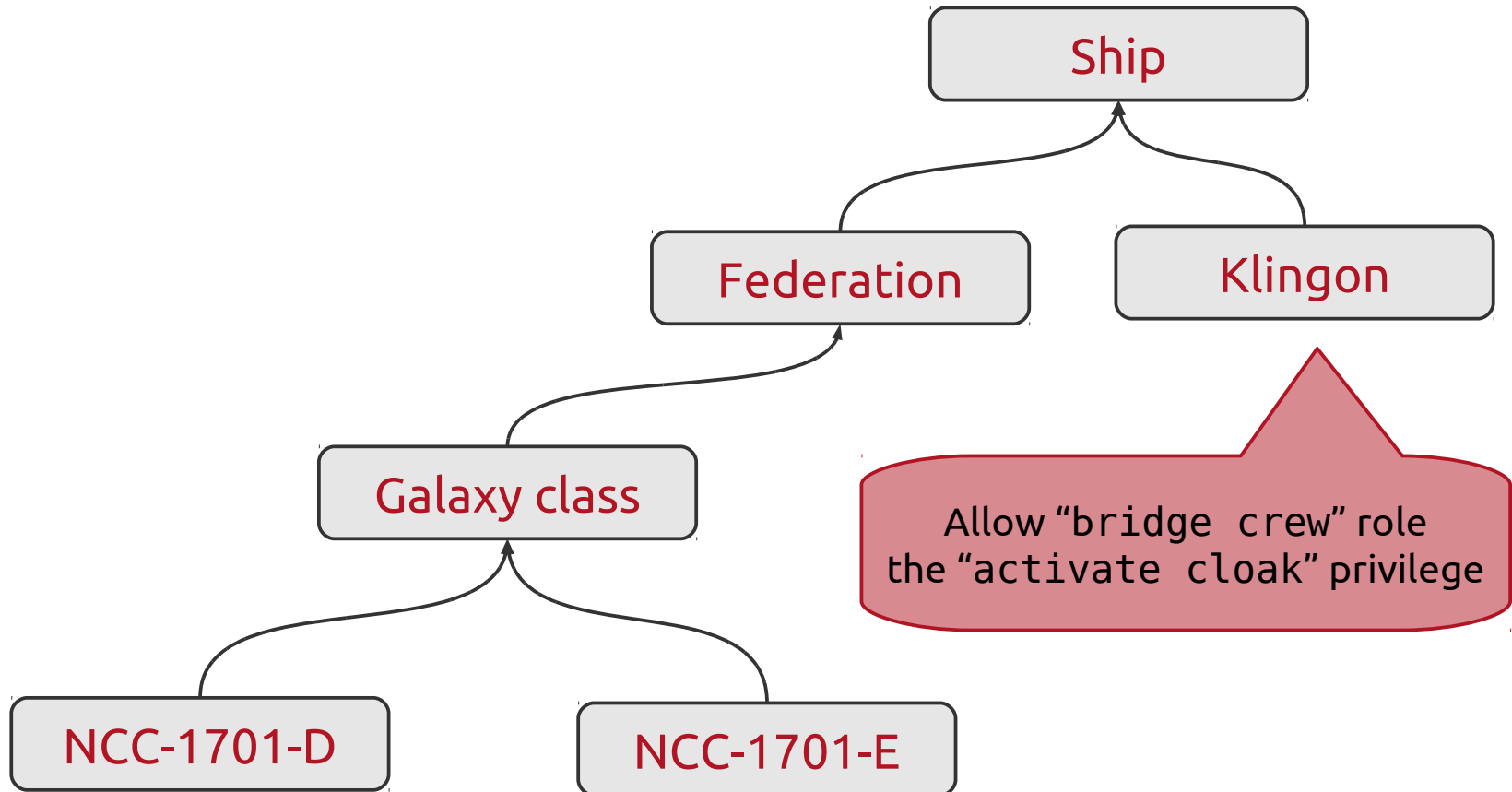
- A user is a leaf node



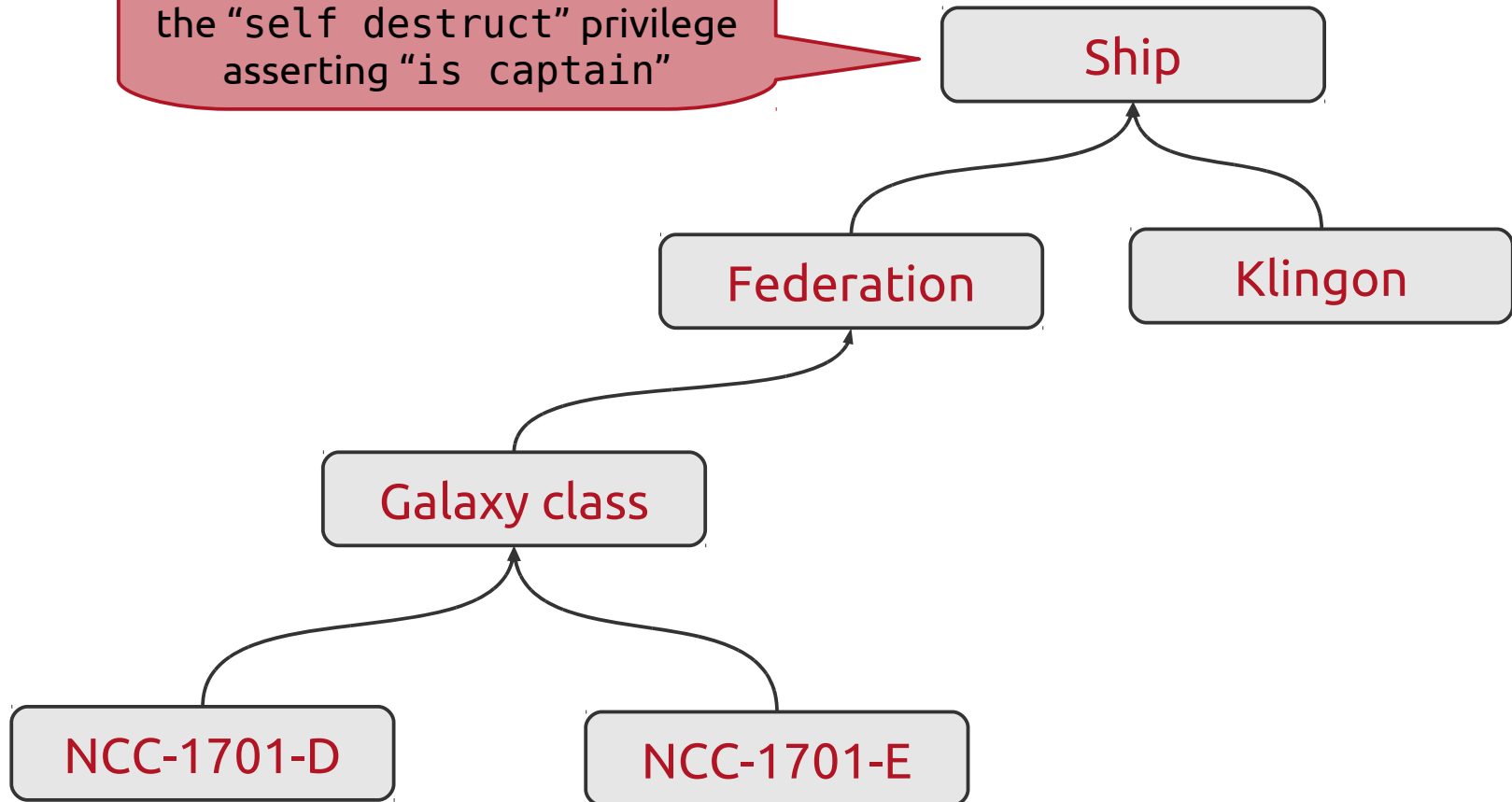
- Use inheritance sparingly
- Avoid circular dependencies
- Over-complicated relationships
- Difficult to configure

- Objects with which users can interact
- A resource may have one parent
- Build the tree from root to leaf





Allow "*" role
the "self destruct" privilege
asserting "is captain"



- Simple – just strings
- Qualifies the operation a role may perform against a resource
- Shared vocabulary: CRUD

- An arbitrary condition attached to the ACL returning true or false
- Has access to the role, resource, privilege and ACL
- Power and flexibility open to abuse

- "user" can "view" a "group photo" if "user is a member of the group"
- "user" can "create" an "comment" if "the user has submitted less than 5 comments in the last hour"
- "job scheduler" may "schedule" a "task" if "no instances of the task are running"

- "user" can "view" a "group photo" if "user is a member of the group"
- "user" can "create" an "comment" if "the user has submitted comments in the last hour"
 - Direct relationship between the role and the resource.
 - All dependencies are passed in.
 - "Visibility" is a good concept to keep in the ACL
- "job scheduler" may "schedule" a "task" if "no instances of the task are running"

- "user" can "view" a "group photo" if "user is a member of the group"
- "user" can "create" an "comment" if "the user has submitted less than 5 comments in the last hour"
- "job scheduler" may "schedule" a "task" if "no instances of the task are running"

Border-line – most dependencies contained
Does the time-based system state count as
"authorisation"?

- "user" can "view" a "group photo" if "user is a member of the group"
- "user" can "create" an "comment" if "the user has submitted less than 5 comments in the last hour"
Advanced dependencies definitely outside the scope
This is a "pre-add" check for the model
- "job scheduler" may "schedule" a "task" if "no instances of the task are running"

Let's see some code

`Zend_Acl`

`Zend_Acl_Role_Interface`

`Zend_Acl_Resource_Interface`

`Zend_Acl_Assert_Interface`

Simple, Static ACL

```
$acl = new Zend_Acl();  
$eng = new Zend_Acl_Role('engineering');  
$scotty = new Zend_Acl_Role('scotty');  
$kirk = new Zend_Acl_Role('kirk');  
$dilCrys = new Zend_Acl_Resource('dilithium crystals');  
  
$acl->addRole($eng);  
$acl->addRole($scotty, $eng);  
$acl->addRole($kirk);  
$acl->addResource($dilCrys);  
  
$acl->allow($eng, $dilCrys);
```

Simple, Static ACL

```
echo "Can Scotty replace the dilithium crystals?\n";  
echo ($acl->isAllowed('scotty', 'dilithium crystals', 'replace')) ?  
"Can do\n" : "Cannae do\n";  
echo "Can Kirk seduce the dilithium crystals?\n";  
echo ($acl->isAllowed('kirk', 'dilithium crystals', 'seduce')) ?  
"Really can\n" : "Obviously not\n";
```

```
rowan@swordbean:~$ php test01.php
```

```
Can Scotty replace the dilithium crystals?
```

```
Can do
```

```
Can Kirk seduce the dilithium crystals?
```

```
Obviously not
```

Any entity in your system:

- Controllers
- Models
- Users
- Files
- Processes

Implementing Resource

```
class Ship implements Zend_Acl_Resource_Interface
{
    public $captain;

    public $registry;

    public function getResourceId()
    {
        return $this->registry;
    }
}

$acl = new Zend_Acl();
$kirk = new Zend_Acl_Role('kirk');
$acl->addRole($kirk);

$ship = new Ship();
$ship->captain = 'kirk';
$ship->registry = 'ncc-1701';
$acl->addResource($ship);
```

Adding an Assertion

```
class IsCaptainOf implements Zend_Acl_Assert_Interface
{
    public function assert(Zend_Acl $acl, Zend_Acl_Role_Interface $role = null,
        Zend_Acl_Resource_Interface $resource = null, $privilege = null) {
        if ( !($resource instanceof Ship) ) {
            throw new Zend_Acl_Exception(
                'IsCaptainOf assertion only valid on Ships' );
        }
        return ($role->getRoleId() == $resource->captain);
    }
}

$assert = new IsCaptainOf();
$acl->allow('kirk', 'ncc-1701', 'destruct', $assert);

echo "Can Kirk order self-destruct?\n";
echo ($acl->isAllowed('kirk', 'ncc-1701', 'destruct')) ?
    "Star Trek III: The Search for Spock\n" : "No\n";
```

Adding an Assertion

```
class IsCaptainOf implements Zend_Acl_Assert_Interface
{
    public function assert(Zend_Acl $acl, Zend_Acl_Role_Interface $role = null,
        Zend_Acl_Resource_Interface $resource = null, $privilege = null) {
        if ( !($resource instanceof Ship) ) {
            throw new Zend_Acl_Exception(
                'IsCaptainOf assertion only valid on Ships' );
        }
        return ($role->getRoleId() == $resource->captain);
    }
}
```

Increasing complexity
Introducing extra points of failure

```
$assert = new IsCaptainOf();
$acl->allow('kirk', 'ncc-1701', 'destruct', $assert);

echo "Can Kirk order self-destruct?\n";
echo ($acl->isAllowed('kirk', 'ncc-1701', 'destruct')) ?
    "Star Trek III: The Search for Spock\n" : "No\n";
```

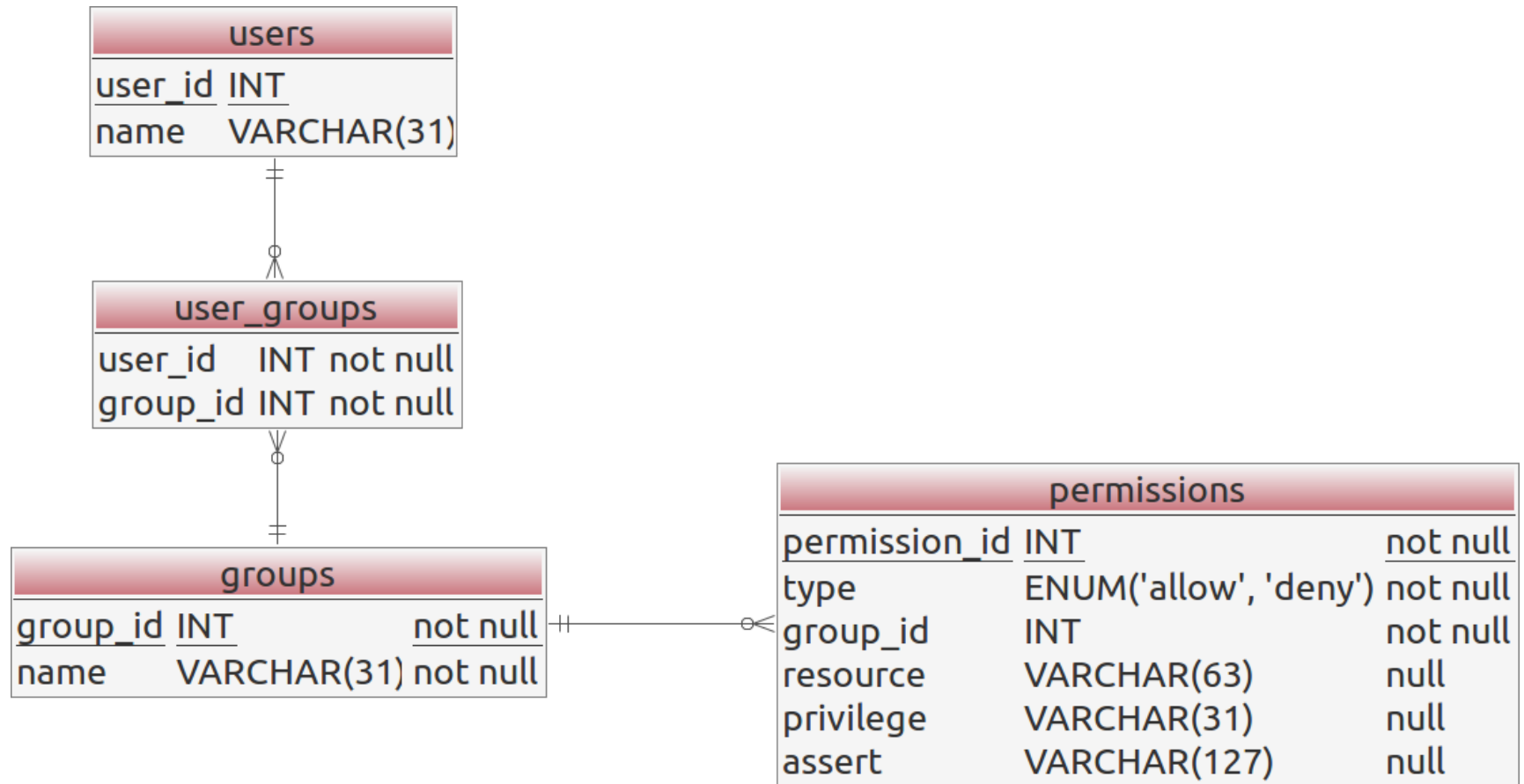
Dynamic ACL

Store config. in the DB

Build on the fly

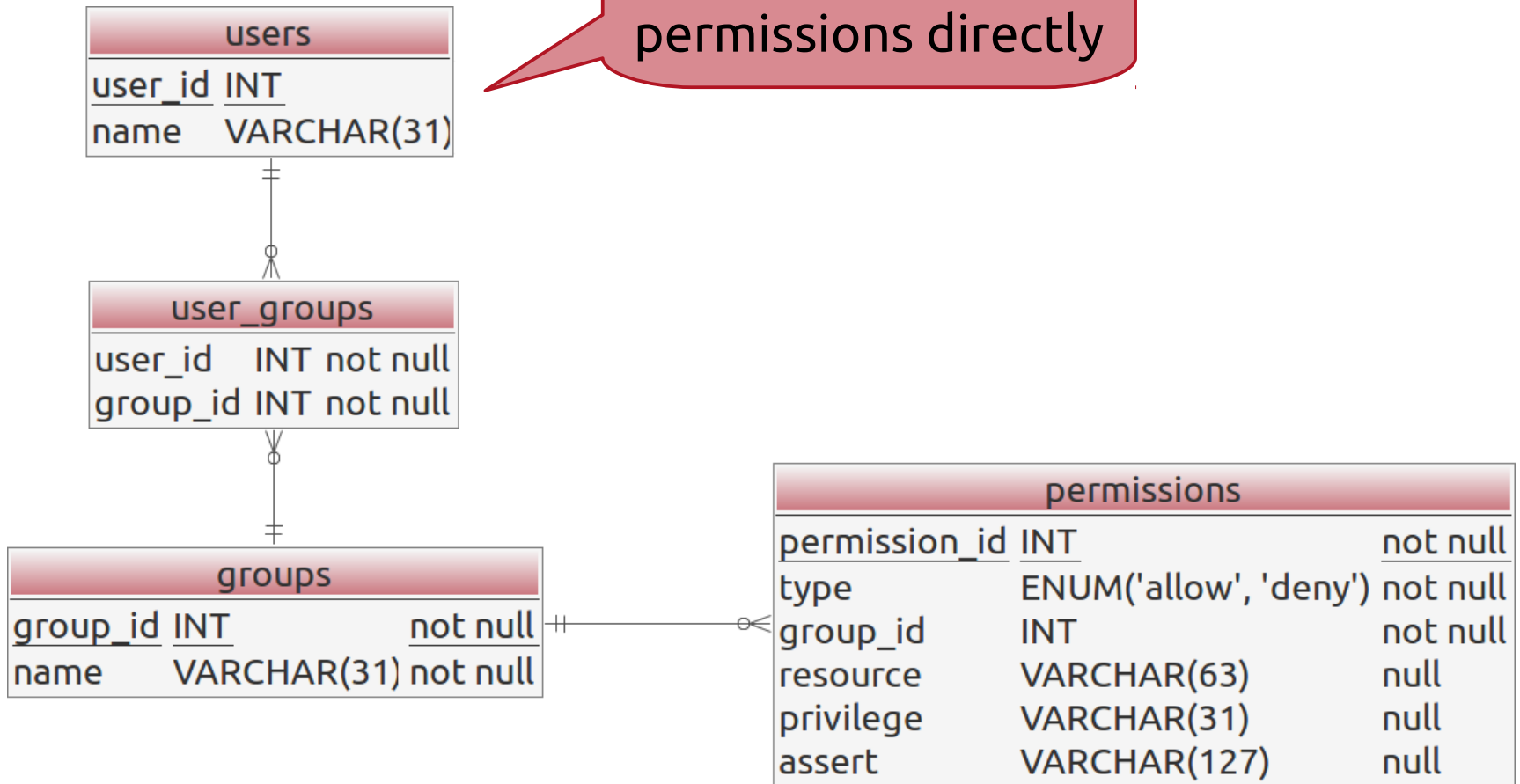
One size **does not** fit all

Database structure

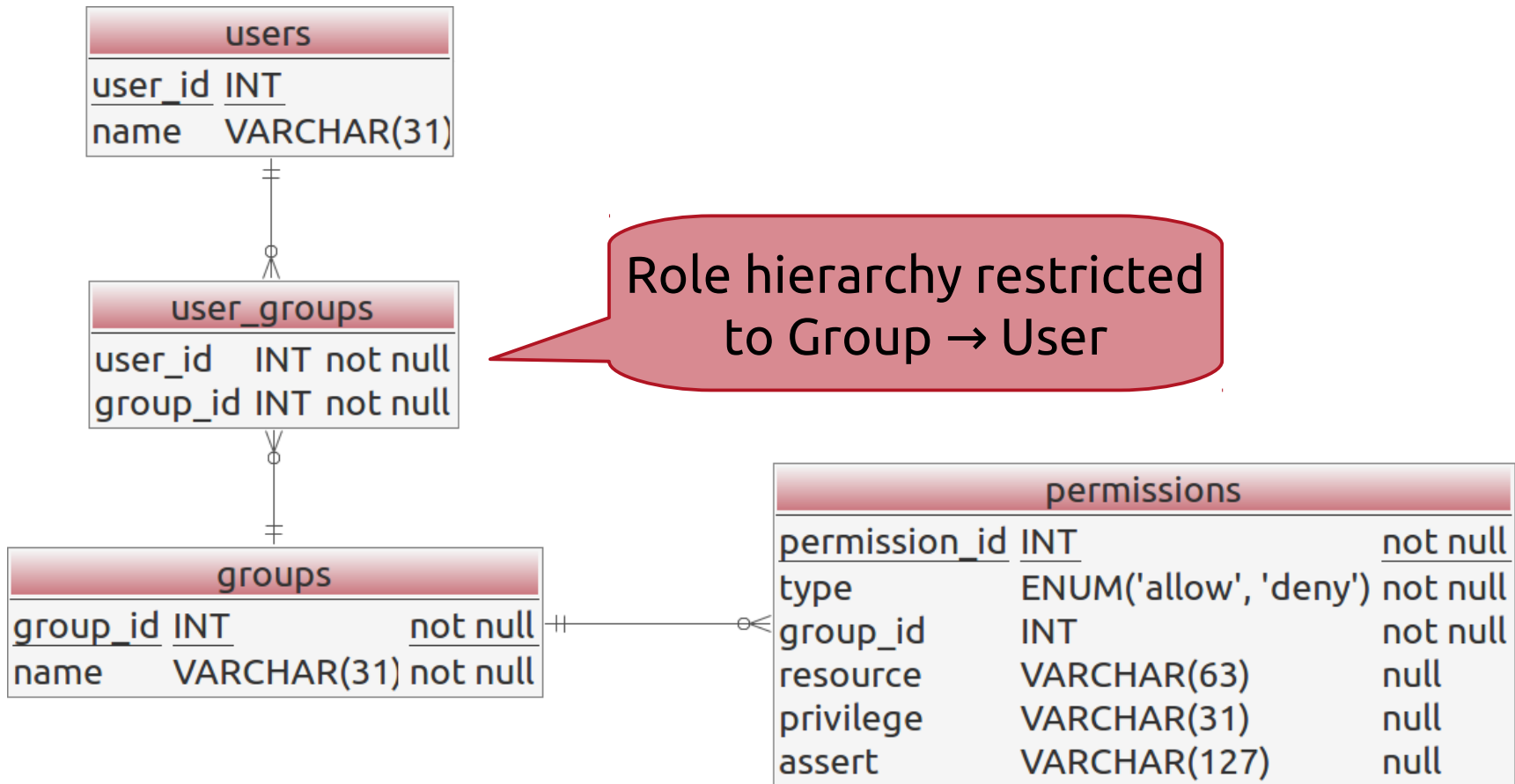


Database structure

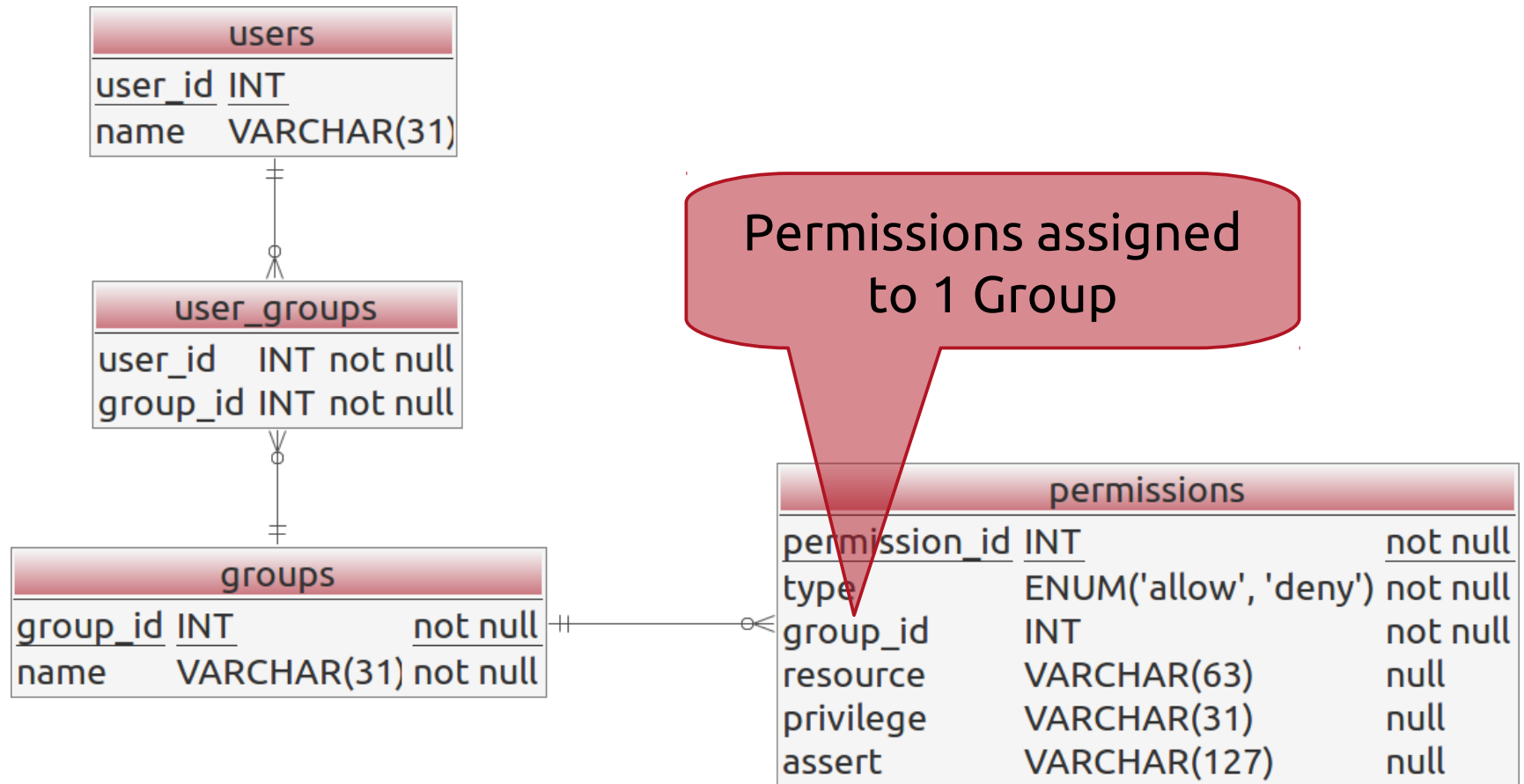
Users not given permissions directly



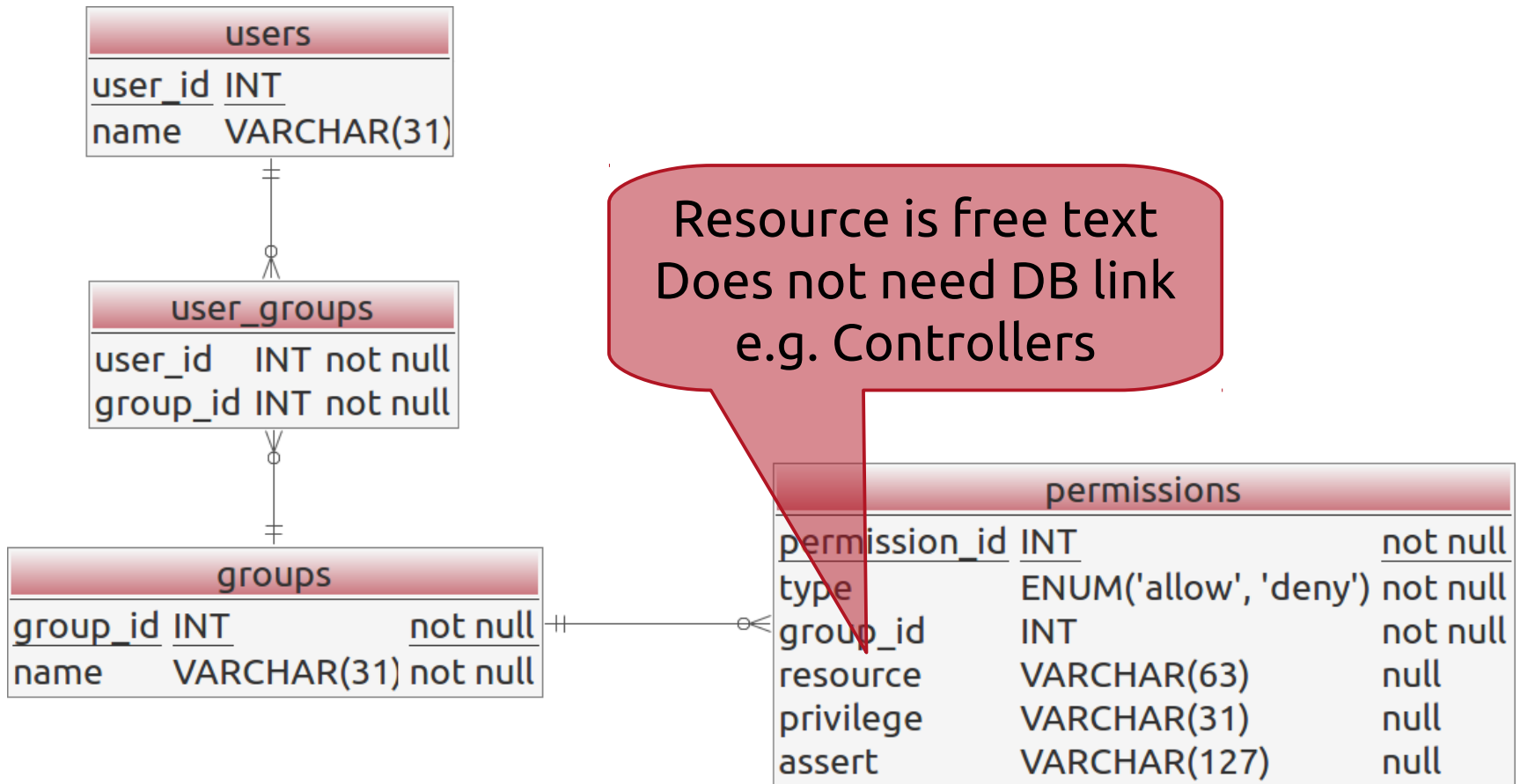
Database structure



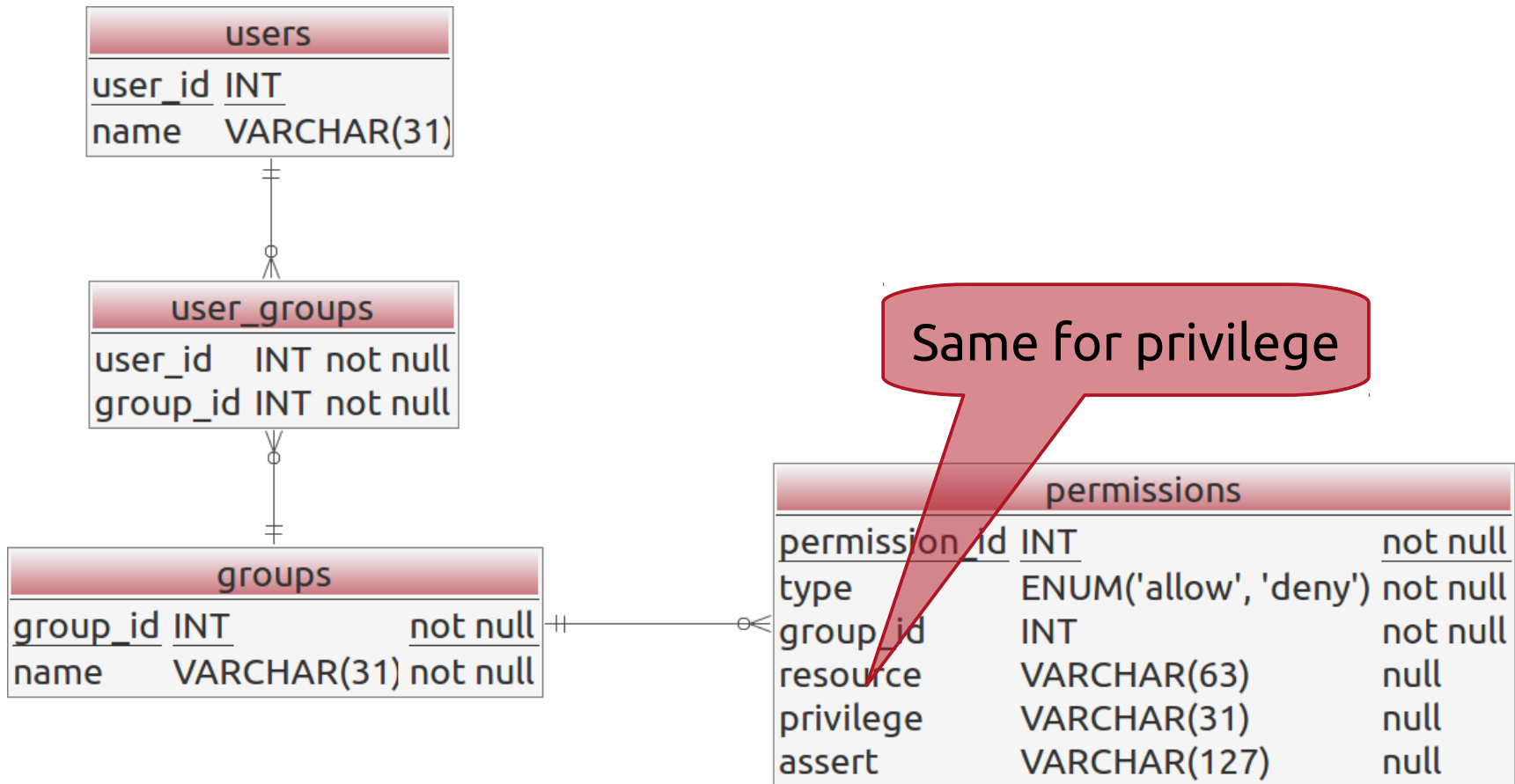
Database structure



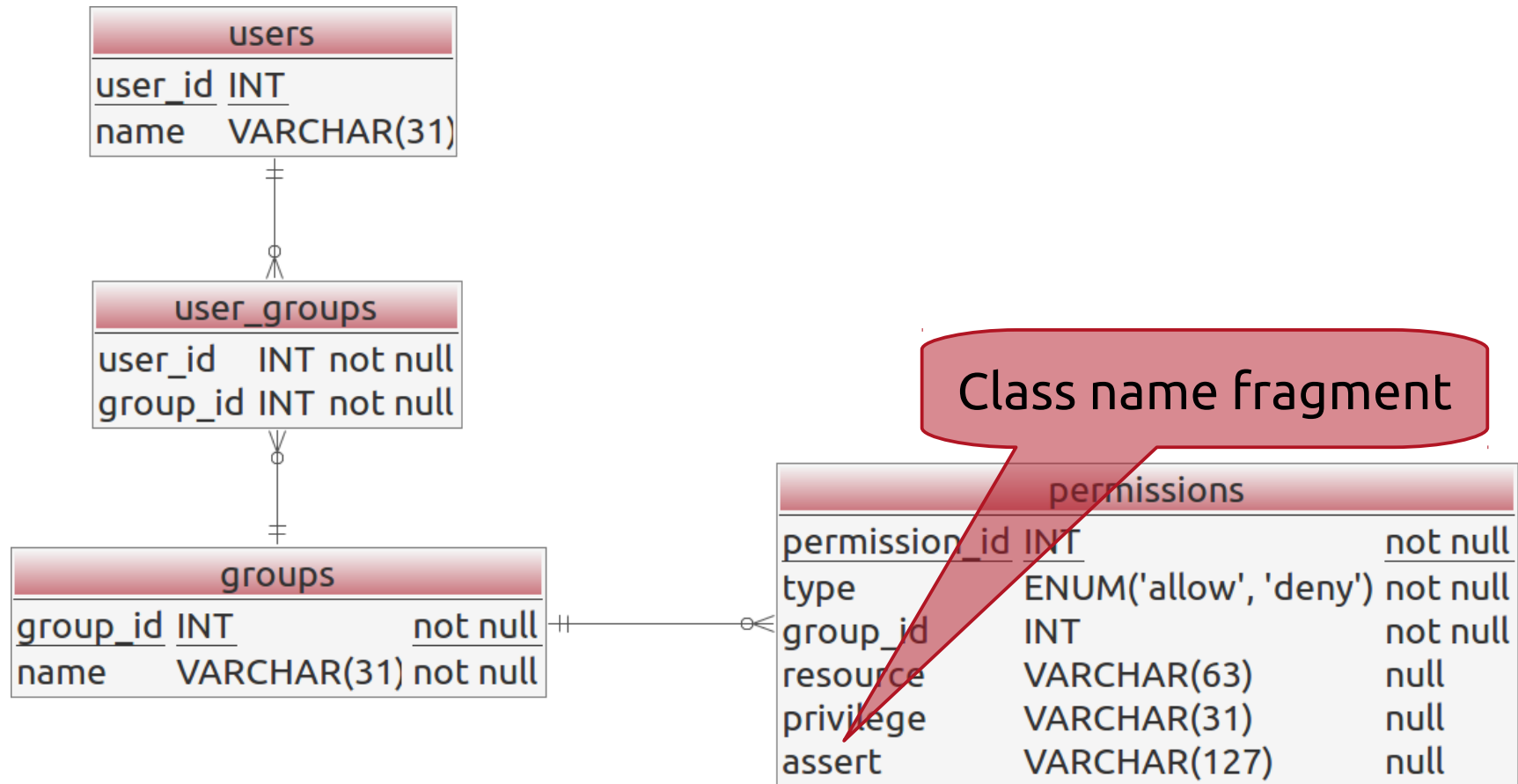
Database structure



Database structure



Database structure



DB Classes

```
class Users extends Zend_Db_Table_Abstract
{
    protected $_name = 'users';
    protected $_primary = 'user_id';
    protected $_dependentTables = array('UserGroups');
    protected $_rowClass = 'User';
}
```

```
class Groups extends Zend_Db_Table_Abstract
{
    protected $_name = 'groups';
    protected $_primary = 'group_id';
    protected $_dependentTables = array('UserGroups');
    protected $_rowClass = 'Group';
}
```

DB Classes

```
class UserGroups extends Zend_Db_Table_Abstract
{
    protected $_name = 'user_groups';
    protected $_primary = array('user_id', 'group_id');
    protected $_referenceMap = array(
        'User' => array(
            'columns'           => array('user_id'),
            'refTableClass'     => 'Users',
            'refColumns'        => array('user_id'),
        ),
        'Group' => array(
            'columns'           => array('group_id'),
            'refTableClass'     => 'Groups',
            'refColumns'        => array('group_id'),
        ),
    );
}
```

```
class Permissions extends Zend_Db_Table_Abstract
{
    protected $_name = 'permissions';
    protected $_primary = 'permission_id';
    protected $_referenceMap = array(
        'Group' => array(
            'columns'           => array('group_id'),
            'refTableClass'    => 'Groups',
            'refColumns'       => array('group_id'),
        ),
    );
}
```

Implementing Role

```
interface Role_Interface extends Zend_Acl_Role_Interface {
    public function getType();
}

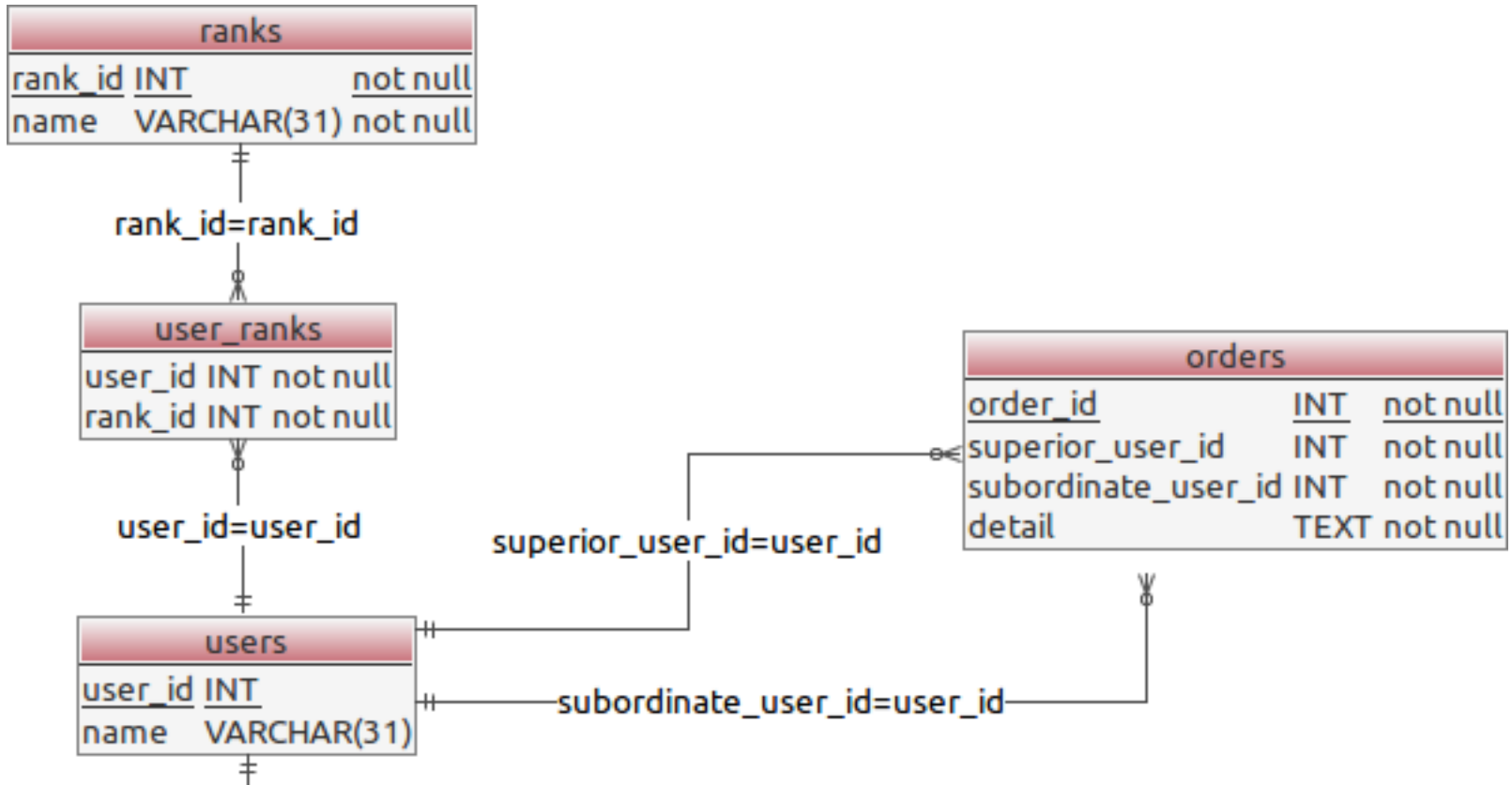
class User extends Zend_Db_Table_Row_Abstract implements Role_Interface {
    public function getType() {
        return 'User';
    }
    public function getRoleId() {
        return $this->getType().':'.$this->user_id;
    }
}

class Group extends Zend_Db_Table_Row_Abstract implements Role_Interface {
    public function getType() {
        return 'Group';
    }
    public function getRoleId() {
        return $this->getType().':'.$this->group_id;
    }
}
```

What do we want to enforce?

Users in the “command” group
may issue orders to users
subordinate to them

Implementing Resource



Implementing Resource

```
interface Resource_Interface extends Zend_Acl_Role_Interface {
    public function getType();
}

class Order extends Zend_Db_Table_Row_Abstract implements Resource_Interface
{
    public function getType() {
        return 'Order';
    }

    public function getResourceId() {
        $id = $this->getType();
        if ($this->order_id) {
            $id .= ':'.$this->order_id;
        }
        return $id;
    }
}
```

Populate the DB

```
mysql> select * from groups;
```

```
+-----+-----+
| group_id | name      |
+-----+-----+
|         1 | command  |
|         2 | bridge crew |
+-----+-----+
```

```
mysql> select * from permissions;
```

```
+-----+-----+-----+-----+-----+-----+
| permission_id | type  | group_id | resource | privilege | assert  |
+-----+-----+-----+-----+-----+-----+
|             1 | allow |         1 | Order    | read      | NULL    |
|             2 | allow |         1 | Order    | create    | IsSuperior |
|             3 | allow |        NULL | Order    | belay     | IsIssuer  |
|             4 | allow |         2 | Order    | read      | NULL    |
+-----+-----+-----+-----+-----+-----+
```

Populate the DB

```
mysql> select u.name, g.name as `group`, r.name as `rank` from users u
        inner join user_groups ug on u.user_id = ug.user_id
        inner join groups g on ug.group_id = g.group_id
        inner join user_ranks ur on u.user_id = ur.user_id
        inner join ranks r on ur.rank_id = r.rank_id;
```

```
+-----+-----+-----+
| name | group      | rank  |
+-----+-----+-----+
| kirk | command    | captain |
| rand | bridge crew | yeoman  |
+-----+-----+-----+
```

Issuing an order

```
$issuer = Zend_Auth::getInstance()->getIdentity();
$u = new Users();
$subord = $u->find(2)->current();

$order = new Order();
$order->superior_user_id = $issuer->user_id;
$order->subordinate_user_id = $subord->user_id;
$order->detail = "Get your red shirt, it's time for an away mission.";

$acl = new AclWrapper();
if (!$acl->isAllowed($issuer, $order, 'create')) {
    throw new Zend_Controller_Action_Exception(
        'Not allowed to create order!' , 403);
}
$order->save();
```

Issuing an order

```
$issuer = Zend_Auth::getInstance()->getIdentity();  
$u = new Users();  
$subord = $u->find(2)->current();  
  
$order = new Order();  
$order->superior_user_id = $issuer->user_id;  
$order->subordinate_user_id = $subord->user_id;  
$order->detail = "Get your red shirt, it's a way mission."  
  
$acl = new AclWrapper();  
if (!$acl->isAllowed($issuer, $order, 'create')) {  
    throw new Zend_Controller_Action_Exception(  
        'Not allowed to create order!' , 403);  
}  
$order->save();
```

You could move this check
onto the model

Building the ACL

```
class AclWrapper
```

```
{  
    public function isAllowed(User $role = null,  
        Resource_Interface $resource = null, $privilege = null) {  
        $acl = new Zend_Acl();  
        $groups = $user->findGroups();  
        foreach ($groups as $group) {  
            $acl->addRole($group);  
        }  
        $acl->addRole($user, $groups);  
        if (strpos($resource->getResourceId(), ':')) {  
            $parent = new Zend_Acl_Resource($resource->getType());  
            $acl->addResource($parent);  
            $acl->addResource($resource, $parent);  
        } else {  
            $acl->addResource($resource);  
        }  
    }  
}
```

```
[...]
```

Building the ACL

```
class AclWrapper
```

```
{
```

```
    public function isAllowed(User $role = null,  
        Resource_Interface $resource = null, $privilege = null) {  
        $acl = new Zend_Acl();  
        $groups = $user->findGroups();  
        foreach ($groups as $group) {  
            $acl->addRole($group);  
        }  
        $acl->addRole($user, $groups);  
        if (strpos($resource->getResourceId(), ':')) {  
            $parent = new Zend_Acl_Resource($resource->getType());  
            $acl->addResource($parent);  
            $acl->addResource($resource, $parent);  
        } else {  
            $acl->addResource($resource);  
        }  
    }
```



Add Group roles
Add the User role

```
[...]
```

Building the ACL

```
class AclWrapper
```

```
{
```

```
    public function isAllowed(User $role = null,  
        Resource_Interface $resource = null, $privilege = null) {  
        $acl = new Zend_Acl();  
        $groups = $user->findGroups();  
        foreach ($groups as $group) {  
            $acl->addRole($group);  
        }  
        $acl->addRole($user, $groups);  
        if (strpos($resource->getResourceId(), ':') {  
            $parent = new Zend_Acl_Resource($resource->getType());  
            $acl->addResource($parent);  
            $acl->addResource($resource, $parent);  
        } else {  
            $acl->addResource($resource);  
        }  
    }
```

'.' means adding an instance
and its parent

```
[...]
```

Building the ACL

```
foreach ($groups as $group) {
    foreach ($groups->findPermissions as $permission) {
        $assert = null;
        $classname = $permission->assert;
        if (
            $classname && class_exists($classname)
            && is_subclass_of($classname, 'Zend_Acl_Assert_Interface')
        ) {
            $assert = new $classname();
        }

        $op = ($permission->type == 'allow') ? 'allow' : 'deny';
        $acl->$op($group, $resource, $permission->privilege, $assert);
    }
}

return $acl->isAllowed($role, $resource, $privilege);
}
```

Building the ACL

```
foreach ($groups as $group) {
    foreach ($groups->findPermissions as $permission) {
        $assert = null;
        $classname = $permission->assert;
        if (
            $classname && class_exists($classname)
            && is_subclass_of($classname, 'Zend_Acl_Assert_Interface')
        ) {
            $assert = new $classname();
        }

        $op = ($permission->type == 'allow') ? 'allow' : 'deny';
        $acl->$op($group, $resource, $permission->privilege, $assert);
    }
}

return $acl->isAllowed($role, $resource, $privilege);
}
```

Validate as much as possible!

Asserting Superiority

```
class IsSuperior implements Zend_Acl_Assert_Interface
{
    public function assert(
        Zend_Acl $acl, Zend_Acl_Role_Interface $role = null,
        Zend_Acl_Resource_Interface $resource = null, $privilege = null)
    {
        if (!$role instanceof User) {
            throw new Zend_Acl_Exception('Assertion only applies to Users');
        }
        if (!$resource instanceof Order) {
            throw new Zend_Acl_Exception('Assertion only applies to Orders');
        }
        $supRank = $role->findRanks()->current();
        $subRank = $resource->findUsersBySubordinate()->current();
        return ($supRank->rank_id > $subRank->rank_id);
    }
}
```

Issuing an order

```
$issuer = Zend_Auth::getInstance()->getIdentity();
$u = new Users();
$subord = $u->find(2)->current();

$order = new Order();
$order->superior_user_id = $issuer->user_id;
$order->subordinate_user_id = $subord->user_id;
$order->detail = "Get your red shirt, it's time for an away mission.";

$acl = new AclWrapper();
if (!$acl->isAllowed($issuer, $order, 'create')) {
    throw new Zend_Controller_Action_Exception(
        'Not allowed to create order!' , 403);
}
$order->save();
```

Conclusions

Is this the right way?

- Controller and Action?
- Model and Method?
- Business object and Action?
- All of the above?

- Pass the Zend_Acl into your wrapper
- Use a factory

- Build everything for the User?
- Build everything for the Resource?
- Build everything for everything!

- Think about what you want to protect
- Test your solution with realistic data
- Assume that you are wrong



Questions?

<http://joind.in/2054>

“...even a well presented talk by a charismatic speaker upset me.”

Credits

<http://www.flickr.com/photos/innoksiuss/2824204305/>

<http://www.flickr.com/photos/carianoff/2849384997/>