Rethinking web application access control

Joomla! 1.6 Security

...because open source matters
Sam Moffatt
[pasamio]
Development Co-ordinator, Joomla!
Systems Co-ordinator, USQ
Master of Computing Student, USQ

@Joomla: Joomla! Installer and Update Systems
@USQ: ePrints, VUFind

Master of Computing, topic:
→ Access control in semantic information systems
Joomla! 1.5 had a primitive ACL model with hard coded rules.

Joomla! 1.5 featured “view” permissions for “public”, “registered” and “special” with hard coded group assignments.

Joomla! 1.5 featured a set of groups that were again hard coded and limited.

The 1.5 ACL model could be best described as hard coded.
The 1.5 security system based primarily on phpGACL
1.5 is a mix of fully discretionary ACL (via phpGACL) and its own “view” permission system for items.
- Compatibility with 1.5
- Flexibility to define:
  - Groups
  - Rules
- Intuitive interface that is simple enough
- Has reasonable default settings
- Custom Groups
- Users can have multiple groups
- Two permission types retained
- View access levels
  - Controls who sees what item in the front-end
- Discretionary rules
  - Controls who can act upon a particular item (e.g. Create, Copy, Delete, Edit)
...because open source matters
View levels are definable and nameable
An item (article, category, menu item) may have only one view level
View levels set which groups are part of it
Groups that are a part of a view level for an item are able to see the items
Works with group hierarchy
...because open source matters
**User Manager: Edit Access Level**

**Level Details**

**Level Title**: Special

**User Groups Having Access**

- Public
- Manager
- Administrator
- Super Users
- Registered
- Author
- Editor
- Publisher
- Shop Suppliers
- Customer Group

...because open source matters
- Discretionary rules provide the ability to limit a set of actions against objects
- Discretionary rules inherit along the group chain
- however...

…because open source matters
By default you can’t do anything
  - Implicit deny.
Until you can allow actions
  - Explicit allow.
Or deny them
  - Explicit deny.
and deny always wins forever after!

...because open source matters
Example 1: Unset Permission

Global: Unset (Deny)
  - Component: Inherit (Deny)
    • Category: Inherit (Deny)
      - Article: Inherit (Deny)

All levels inherit the implied deny.
Example 2: Allow

Global: **Allow**
- Component: Inherit (**Allow**)
  - Category: Inherit (**Allow**)
    - Article: Inherit (**Allow**)

All levels inherit allow.
Example 3: Mixed unset/allow/deny

- Global: Unset (Deny)
  - Component: Allow
    - Category: Deny
      - Article: Allow (Deny)

- Globally denied (not allowed in the global context).
- Component is explicitly allowed.
- Category is explicitly denied.
- Article is denied regardless of the setting.

...because open source matters
Rule Levels

- Global
- Component
- Category
- Article/Item

…because open source matters
Basic Permissions

- Admin
- Site Login
- Admin Login
- Manage
- Create
- Delete
- Edit
- Edit State
- Rules inherit down from Global to Component to Category to Article
- Rules inherit down from the group parent to its siblings
- Admin at global level is like root
- Admin at component can change anything for that particular component
…because open source matters
- Login permissions control if people can log into particular aspects of a site.
- Site login controls front-end login
- Admin login controls back-end login
- Both permissions are independent
- One can be granted or denied without the other
• Grants access to the administrator part of a component.

• Global level: grants to all components

• Component: grants to just that one
because open source matters
- **Global**: Create content in any component
- **Component**: Create content in this component (any category)
- **Category**: Create subcategories or content in this category
- **Note**: Doesn't apply to articles, only to the container!
…because open source matters
- Global: Delete any content in any component.
- Component: Delete any content in this component.
- Category: Delete this category, sub-categories and content in this category.
- Article: Delete this article.
...because open source matters
- **Global**: Edit any content in any component.
- **Component**: Edit any content in this component.
- **Category**: Edit this category, sub-categories and content in this category.
- **Article**: Edit this article.
…because open source matters
Basic Permissions – Edit State

- **State** refers to publishing, trashing, ordering, etc.
- **Global**: Edit state of any content in any component.
- **Component**: Edit state of any content in this component.
- **Category**: Edit state of this category, sub-categories and content in this category.
- **Article**: Edit state of this article.

...because open source matters
Basic Permissions – Edit State

…because open source matters
Third party developers can use the API to create their own actions

Level of access control for third parties depends on their desire to implement

Joomla! handles “manage” to restrict backend access but nothing more from there

Tools to handle user interface supplied via public APIs
Demonstration

...because open source matters
Questions and answers

...because open source matters
Portions of these slides are drawn from Andrew Eddie's Joomla! 1.6 presentation.

Andrew's Presentation:

Andrew's video on 1.6 permissions:
http://vimeo.com/12900266

Andrew's article on 1.6 Permissions:
These slides available on conf.oss.my
Also available on USQ ePrints:
  - http://eprints.usq.edu.au/8330
My other papers/presentations:
  - http://eprints.usq.edu.au/profile/404