UNIVERSITY OF Southampton

Authorization: Implementation, Attacks and Threats

By Dr. Nawfal Fadhel

Contributors
Jim Manico

GCHQ Academic Centre of Excellence EPSRC

Question

- Have you ever been hacked? (Insight)
- Can you give a bad example of Authorization?

Where is Authorization in Access Control?

- Authentication, Authorization, Accounting (AAA)
- Authorization A property where access is granted to resources based on access rights
- Authorization rules make sure users can only preform what are permitted to do.
- DAC for desktops
- MAC for Networks
- RBAC for application

Authorization Types

File permission	Share Permission
Access files locally	Access files over the network
Files are stored on local machine and Disk type is readable by OS	Files are stored on remote machine and Disk type dose not have to match the connected machine.
Managed by operating system	Managed by sharing protocol

Combining Shared Folder and NTFS Permissions

- When you combine NTFS permissions and share permissions the **most restrictive** effective permission applies.
  - For example, if you share a folder and assign the *share permission* READ to EVERYONE and assign FULL CONTROL *NTFS permissions* to Everyone, users connecting through the network will have Read permissions.
- When accessing a file locally, only NTFS permissions apply

Calculating Effective Permissions

- Both Share and NTFS Permissions are Cumulative
  - Cumulative permissions:
    - Permissions are combined when a user is **not explicitly denied** access
    - A user's effective permissions for a resource are the sum of the NTFS permissions that you assign to the individual user account and to all of the groups to which the user belongs.

Example

If a user has Read permissions for a folder

and

is a member of a group with write permissions for the same folder,

the user's cumulative permissions are both Read and Write.

Calculating Effective Permissions

To calculate effective permissions when combining share permissions and NTFS

1. Determine the *effective* NTFS permissions
2. Determine the *effective* share permissions
3. Take the most restrictive of the two.

Access Control Attacks

- Insider threat
  - A user with malicious intent accessing administrator account.
- Identity theft
  - User with the same role, accessing another person private information. (Spoofing, Phishing)
- Vulnerability in implementation
  - Logic error (Logic Bomb)
- Design flaw
  - Bad workflow design (Error in policy)

That lead to

Access Controls Threats - AAA

With the assumption that authorization is compromised

I. Authorization (Elevation of privileges) For example: using a user account, change it to service and then to admin.

II. Accounting (Clearing tracks) for example: deleting all logs during cyber attack to hid all tracks.

Access Controls Threats - CIA

I. Confidentiality (Disclosure of confidential data) for example: Compromising admin-level accounts often results in access to user's confidential data

II. Integrity (Data tampering) for example: Privilege levels do not distinguish users who can only view data and users permitted to modify data.

III. Availability (Data destruction) for example: deleting all sensitive information inside a system.

Designing Access Control - Bad Practices

- Hard-coded role checks in application code
- Lack of centralized access control logic
- Untrusted data driving access control decisions
- Access control that is “open by default”
- Lack of addressing compromised accounts on access control in a standardized way.
- Access control logic that needs to be manually added to every endpoint in code.

Designing Access Control – Good Practices

- Code to the activity, not the role
- Centralize access control logic
- Design access control as a filter
- Deny by default, fail securely
- Build centralized access control mechanism
- Apply same core logic to presentation and server-side access control decisions
- Server-side trusted data should drive access control

Question

- What is hard coding?

Hard Coded Roles

- Makes "proving" the policy of an application difficult for audit or Q/A purposes
- Any time access control policy needs to change, new code need to be pushed
- Fragile, easy to make mistakes

Hard Coded Roles - Example

```
if (user.isManager()         ||
    user.isAdministrator() ||
    user.isEditor()          ||
    user.isUser()) {
//  execute action
}
```

Faulty process flow

- Makes “protecting” the content of an application difficult for privacy purposes
- Access control policy dose not protect content behind authentication
- Vulnerable to path crawling

Faulty process flow - Example

```
http://example.com/buy?action=chooseDataPackage
http://example.com/buy?action=customizePackage
http://example.com/buy?action=makePayment
http://example.com/buy?action=downloadData
```

Can an attacker control the sequence?

Can an attacker abuse this with concurrency?

Trusted Data

- Never trust user data for access control decisions
- Never make access control decisions in JavaScript
- Never make authorization decisions based solely on
  - hidden fields
  - cookie values
  - form parameters
  - URL parameters
  - anything else from the request
- Never depend on the order of values sent from the client

No data restriction– open by default

- Many administrative interfaces require only a password for authentication
- Shared accounts combined with a lack of auditing and logging make it extremely difficult to differentiate between malicious and honest administrators
- Administrative interfaces are often not designed as "secure" as user-level interfaces given the assumption that administrators are trusted users
- Authorization/Access Control relies on client-side information (e.g., hidden fields)

```
<input type="text" name="fname"
  value="Derek">
<input type="text" name="lname"
  value="Jeter">
<input type="hidden" name="usertype"
  value="admin">
```

No planning for compromised accounts

- Many applications utilize an "all or nothing" approach
  - Once authenticated all users have equal privilege levels
- Authorization logic often relies on Security Through Obscurity (STO) by assuming:
  - Users won't find unlinked or "hidden" paths/functionality.
  - Users will not find and tamper with "obscured" client side parameters (i.e. "hidden" form fields, cookies, etc)
- Applications with multiple permission levels/roles often increases the possibility of conflicting permission sets resulting in unanticipated privileges

Best Practice: Code to the Activity

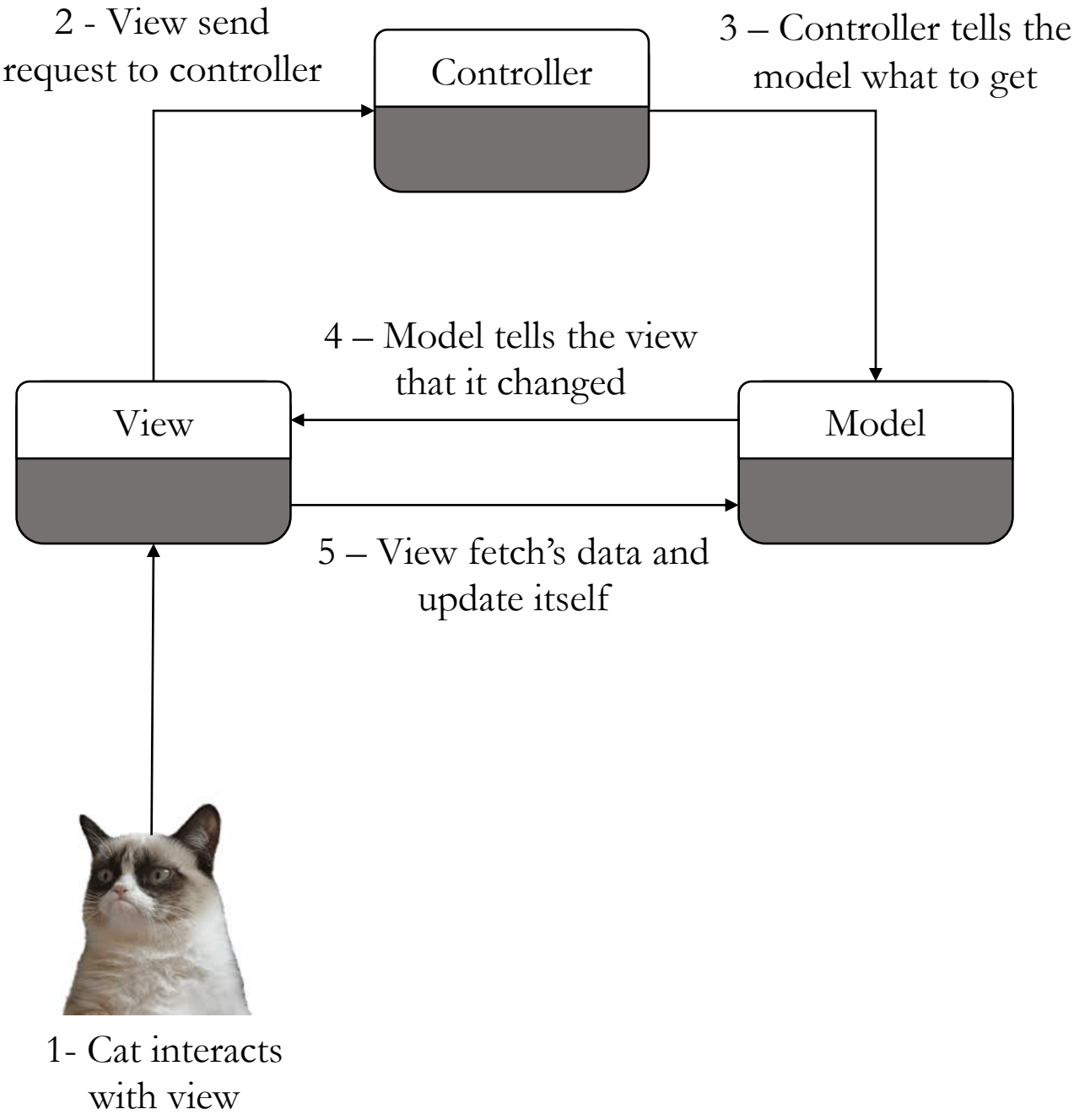
```
if (AC.hasAccess(ARTICLE_EDIT)) {
   //execute activity
}
```

- Code it once, never needs to change again
- Implies policy is persisted/centralized in some way
- Requires more design/work up front to get right

Best Practice: Centralized ACL Controller

- Define a centralized access controller
  - ACLService.isAuthorized(ACTION_CONSTANT)
  - ACLService.assertAuthorized(ACTION_CONSTANT)
- Access control decisions go through these simple API's
- Centralized logic to drive policy behavior and persistence
- May contain data-driven access control policy information

Building Access Control - MVC



Using a Centralized Access Controller

In View Layer

```
if (isAuthorized(VIEW_LOG_PANEL))
{
  <h2>Here are the logs</h2>
  <%=getLogs();%/>
}
```

In Controller

```
try (assertAuthorized(DELETE_USER))
{
  deleteUser();
}
```

Best Practice: Verifying policy server-side

- Keep user identity verification in session
- Load entitlements server side from trusted sources
- Force authorization checks on ALL requests
  - JS file, image, AJAX and FLASH requests as well!
  - Force this check using a filter if possible

SQL Integrated Access Control

Example Feature

http://mail.example.com/viewMessage?msgid=2356342

This SQL would be vulnerable to tampering

select * from messages where messageid = 2356342

Ensure the owner is referenced in the query!

select * from messages where messageid = 2356342 AND messages.message_owner = <userid_from_session>

Data Contextual Access Control

Data Contextual / Horizontal Access Control API examples

- ACLService.isAuthorized(EDIT_ORG, 142)
- ACLService.assertAuthorized(VIEW_ORG, 900)

Long form

- isAuthorized(user, EDIT_ORG, Organization.class, 14)

- Essentially checking if the user has the right role in the context of a specific object
- Protecting data at the lowest level!

Data Contextual Access Control

User	
User ID	User Name

Role/Activity	
Role/Activity ID	Role/Activity Name

Entitlement / Privilege			
User ID	Role/Activity ID	Data Type ID	Data Instance Id

Data Type	
Data ID	Data Name

Coding Practices to Defend Against Access Control Attacks

- Implement role based access control to assign permissions to application users for vertical access control requirements
- Implement data-contextual access control to assign permissions to application users in the context of specific data items for horizontal access control requirements
- Avoid assigning permissions on a per-user basis
- Perform consistent authorization checking routines on all application pages
- Where applicable, apply DENY privileges last, issue ALLOW privileges on a case-by-case basis

System Polices to Defend Against Access Control Attacks

- Where possible restrict administrator access to machines located on the local area network (i.e. it's best to avoid remote administrator access from public facing access points)
- Log all failed access authorization requests to a secure location for review by administrators
- Perform reviews of failed login attempts on a periodic basis
- Utilise the strengths and functionality provided by the SSO solution you chose, e.g. google SSO

Testing for Broken Access Control

- Attempt to access administrative components or functions as an anonymous or regular user
  - Scour HTML source for "interesting" hidden form fields
  - Test web accessible directory structure for names like admin, administrator, manager, etc (i.e. attempt to directly browse to "restricted" areas)
- Determine how administrators are authenticated. Ensure that adequate authentication is used and enforced
- For each user role, ensure that only the appropriate pages or components are accessible for that role
- If able to compromise administrator-level account, test for all other common web application vulnerabilities (poor input validation, privileged database access, etc)

Summary

- What is authorization?
- Authorization types
- Authorization threats
- Authorization Design
- Authorization Implementation
- Authorization Testing

Question

Share Permissions of PublicApps

•Everyone Change

NTFS Permissions of PublicApps

John: Full Control

Sales: Read

You share a folder on your computer and you assign the share permission Change to Everyone. John, a user from the Sales Department, has been granted Full Control NTFS permissions to the folder.

John is a member of the Sales Group, which has been assigned the READ NTFS permission.

What are John's effective permissions when connecting to the share from across the network?

Question

Share Permissions of PublicApps

•Everyone Change

NTFS Permissions of PublicApps
John: Full Control
Sales: Read

John's Effective NTFS Permissions: Full Control
John's Effective Share Permissions: Change
Most Restrictive of the two: Change

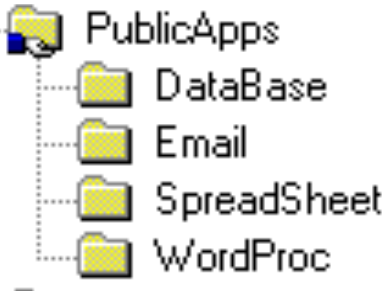
Extended Share Permissions

Rules to Remember

- If you or a group you belong to is on both the share permissions access control list (ACL) and the NTFS ACL, you can browse into the share
- If you or a group you belong to is on only the share ACL, you cannot browse in but, if you have rights to folders beneath the shared folder you can access them using a UNC path.
- If you or a group you belong to are only on the NTFS ACL, you cannot browse into the share and you cannot access any folders beneath the share, even if you have rights to them.

A Suggested Security Assignment for PUBLIC APPLICATION FOLDERS

Permissions assigned here assume that all users in the domain should be able to run programs that exist in any of the share's subfolders.



Share Permissions

•Everyone Full Control

- **NTFS Permissions**
- PublicApps: Administrators Full Control
  Users Read & Execute; List Folder Contents; Read

If the PublicApps folder is created at the root of the drive and Microsoft's default NTFS permissions haven't been changed at the root, you can use the default NTFS permissions.

A Suggested Security Assignment for PUBLIC APPLICATION FOLDERS

Permissions assigned here assume that all users in the domain should be able to run programs that exist in any of the share's subfolders.

Share Permissions

- Users Read
- Administrators Full Control

NTFS Permissions

PublicApps: Administrators Full Control
Users: Read and Execute
List Folder Contents
Read

If the PublicApps folder is created at the root of the drive and Microsoft's default NTFS permissions haven't been changed at the root, you can use the default NTFS permissions.

A Suggested Security Assignment for PUBLIC APPLICATION FOLDERS

Permissions assigned here assume that all users are able to add to, delete from and change the contents of files in the shared folder area. Users should not however be able to change permissions on a file or folder nor should they be able to take ownership of a file or folder.

Share Permissions

•Everyone Full Control

NTFS Permissions

- PublicData: Administrators Full Control
  Users everything *but* Full Control

A Suggested Security Assignment for PUBLIC APPLICATION FOLDERS

Permissions assigned here assume that all users are able to add to, delete from and change the contents of files in the shared folder area. Users should not however be able to change permissions on a file or folder nor should they be able to take ownership of a file or folder.

Share Permissions

- Administrators Full Control
- Users Change

NTFS Permissions

- PublicData: Administrators Full Control
  Users everything *but* Full Control

A Suggested Security Assignment for PUBLIC APPLICATION FOLDERS

Permissions assigned here assume that users in each department should only have access to their department's applications. (i.e., Accounting can only access Accounting; Sales can only access Sales, etc.)

Share Permissions

Everyone Full Control

- **NTFS Permissions**
- PrivateApps: Administrators Full Control
  - Remove Inheritance from above (do not allow inheritable permissions from this object's parent) After removing the inheritance make sure Administrators have full control applied to This folder, subfolders and files.
- Each subfolder
  - Administrators should already be assigned full control because of inheritance
  - Assign each group the following permissions to their department's respective folder (i.e., Sales group to the Sales folder; Marketing group to the Marketing folder, etc.) (users in each department will have to access their respective folder via the UNC path)
    - Read and Execute,
    - List Folder Contents
    - Read

A Suggested Security Assignment for PUBLIC APPLICATION FOLDERS

Permissions assigned here assume that users in each department should only have access to their department's applications. (i.e., Accounting can only access Accounting; Sales can only access Sales, etc.)

Share Permissions

Everyone Full Control

- **NTFS Permissions**
- PrivateApps: Administrators Full Control
  Users Read and Execute, List Folder Contents, Read
  - If the PrivateApps folder is created at the root of the drive and Microsoft's default NTFS permissions haven't been changed at the root, you can use the default NTFS permissions.
- Each subfolder
  - Remove Inheritance from above (do not allow inheritable permissions from this object's parent) After removing the inheritance make sure Administrators have full control applied to This folder, subfolders and files.
  - Assign each group the following permissions to their department's respective folder (i.e., Sales group to the Sales folder; Marketing group to the Marketing folder, etc.)
    - Read and Execute,
    - List Folder Contents
    - Read

A Suggested Security Assignment for PUBLIC APPLICATION FOLDERS

Permissions assigned here assume that users in each department should only have access to their department's data. Users in each department should be able to add to, delete from and change the contents of files in their department's folder.

Share Permissions

•Everyone Full Control

NTFS Permissions

- PrivateData: Administrators Full Control
  - Remove Inheritance from above (do not allow inheritable permissions from this object's parent) After removing the inheritance make sure Administrators have full control applied to This folder, subfolders and files.
- Each subfolder
  - Administrators should already be assigned full control because of inheritance
  - Assign each group everything *but* Full Control to their respective folder (i.e., Sales group to the Sales folder; Marketing group to the Marketing folder, etc.) (users in each department will have to access their respective folder via the UNC path)

A Suggested Security Assignment for PUBLIC APPLICATION FOLDERS

Permissions assigned here assume that users in each department should only have access to their department's data. Users in each department should be able to add to, delete from and change the contents of files in their department's folder.

Share Permissions

•Everyone Full Control

- **NTFS Permissions**
- PrivateData: Administrators Full Control
  Users Read and Execute, List Folder Contents, Read
  - If the PrivateData folder is created at the root of the drive and Microsoft's default NTFS permissions haven't been changed at the root, you can use the default NTFS permissions.
- Each subfolder
  - Remove Inheritance from above (do not allow inheritable permissions from this object's parent) After removing the inheritance make sure Administrators have full control applied to This folder, subfolders and files.
  - Assign each group everything *but* Full Control to their department's respective folder (i.e., Sales group to the Sales folder; Marketing group to the Marketing folder, etc.)