M183 Applikationssicherheit Implementieren # 11

By Jürg Nietlispach

Roadmap


Pre-Auth Sessions
Authentication
Session Management
Access Control
Session finalization

Sessions – what’s next?

Recap: Sessions provide a solution that subsequent requests can be identified as «trusted» after one single successful login procedure.

Consider that a user has successfully logged in and has received a session:

- How can we now guarantee, that an authenticated user accesses data only meant for him/her?
- How can we now guarantee, that an authenticated user performs actions only he is allowed to?
- How can we now guarantee, that …

What do we need additionally?

Authorization

«Authorization is the process of deciding whether an **actor** can perform a certain **action** (or not)»

Actor? Action?

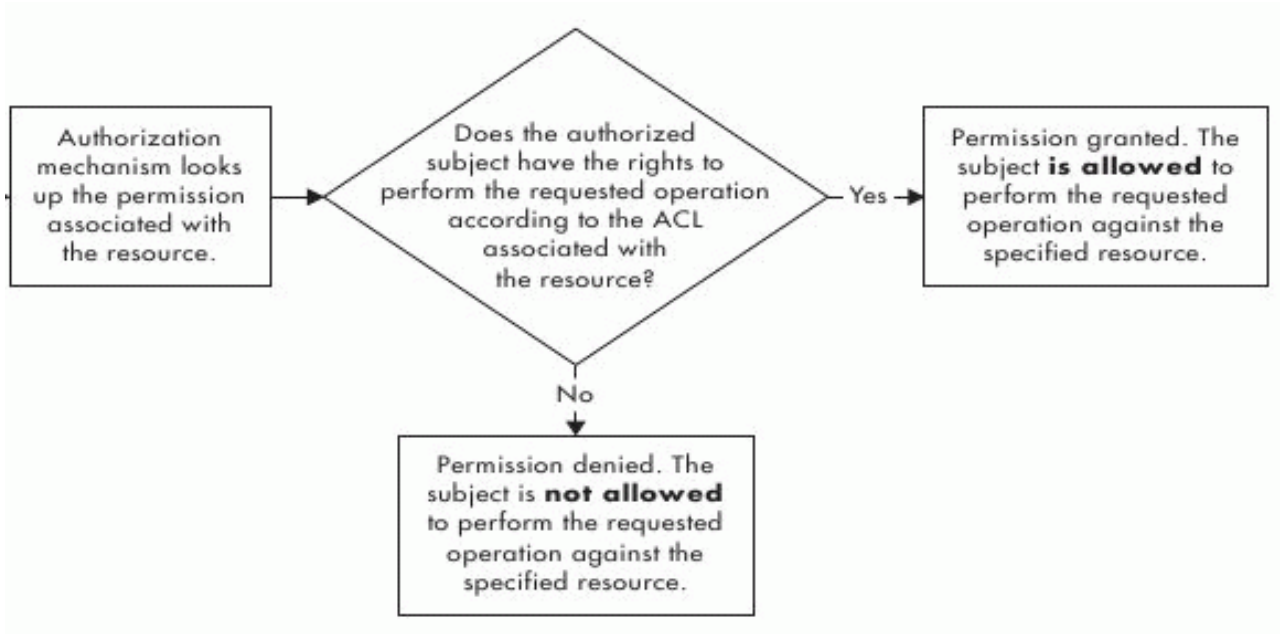
Authorization

Actors: User, Agent, Bot, Systems, APIs …

Actions (on Web-Ressources and Data):

- Access to Web-Resources,
- File Access, Data Access,
- File Manipulation,
- Data Manipulation
- …

Example Workflow



Consider a (multi role) intranet solution (User, Admin, Steering)

1. A user logs in and receives a Session
2. A user then wants to access i.e. a document from the intranet
3. Depending on the AC-Policies and permissions the user was granted he can access the file or not

- Either only documents are shown to the user that he can access
- Or a full list of documents is shown and the permission check is made upon file-access

Goals?

What must be achived?

What must be prevented?

Goals

- An actor performs actions only within his privilege level
- Access protected ressources only upon actors privilege level
- Prevent privilege escalation attacks

Access Control Models

Idea: deceiding whether an actor **can** perform an action

Based on Trusted Computer System Evaluation Criteria (see Wikipedia)

1. Discretionary Access Control (DAC) – Benutzerdefinierte bzw. Benutzerbestimmbare Zugriffskontrolle
Access control by means of restricting **access** to objects based on the **identity** of subjects and/or groups to which they belong to.

Example: Document Access Control defined by the owner of a Document

2. Mandatory Access Control (MAC) – Zugriffskontrolle durch System bedingt bzw. auf Regeln basierend
Access control is determined by the system or the system administrators.

Exmple: System provides several Categories for Classified Documents (Public, Confidential, etc.)

Access Control Models 2

3. Role Based Access Control (RBAC) – Rollenbasierte Zugriffsrechte
Similar to MAC – but instead of using rules, roles are used.

Example: Every user has assosiated role(s). Depending on the role, access can be granted

4. Hybrid Systems - Combinations of the above models

Permission Models

So far: using authorization, we can control the access to a ressource or data of a system.
Now: deciding whether an actor can perform a **certain action.**

Read
«Is a user allowed to see this data?»
-> Matters for data-ressources, not for ressources that provide functionality.

Write
«Is a user allowed to change this data?»
-> Matters for data-ressources, not for ressources that provide functionality.

Execute
«Is the user allowed to make this action?»
-> Matters primarily for ressources that provide functionality.

Custom Authorization – 3x3 Matrix

What

- Entities, that is making a request to on a ressource (User, Subjects, Systems)
- Functionality an entity has
- Data, ressources that are managed by a web application

When

- Timeslots, when authorization / permission checks have to be done

Who/How

- User Roles that can trigger / enable these actions / permissions and how

This is a very general concept, used also in non-IT-contexts!

Example 3x3 Matrix

Authorization Matrix

Key

A: Authorized to perform task
R: Responsible for task being performed

Abbreviations:

- LM: Laboratory Manager
- QO: Quality Officer
- BO: Biosafety Officer
- EO: Equipment Officer
- LT: Laboratory Technologist
- IA: Internal Auditor
- Secr: Secretariat
- OH: Occupational Health

Task	LM	QO	BO	EO	LT	Secr.	Other
General							
Implementing and maintaining the quality management system, content wise	R	A					
Implementing and maintaining the quality management system, design and assurance	A	R					
Formulating and changing quality policy	R/A						
Implementing the quality policy	R	A	A	A			
Evaluating quality policy (audits, management review, etc.)	R	A					IA: A
Translating quality targets in action plans	R	A					
Executing action plans originating from quality targets	R	A	A	A	A		
Archiving and follow-up of action plans originating from quality targets	R	A					
Providing information through quarterly reports, needed for the management review	R	A					
Performing and recording management review	R	A					
Archiving of management review and quarterly reports	R						
Formulating actions following management review in action plans	R	A					
Executing action plans originating from management review	R	A	A	A	A		
Archiving and follow-up of action plans originating from management review		R	A	A	A		
Drafting quality year plan	R/A						
Execution of selection and application procedure new personnel	R				A		
Ensuring that research methods and equipment comply with present scientific knowledge and technical	R			A	A		

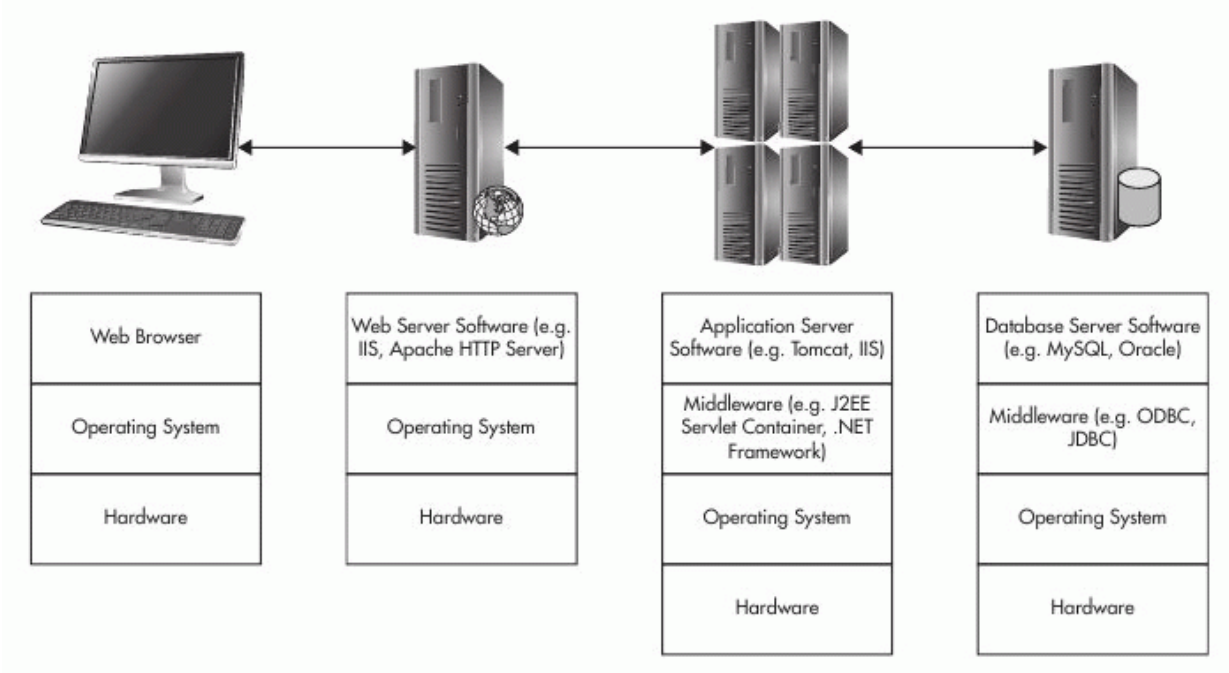
Per Layer Authorization

Similar to the Session-Concepts, Authorization can be achieved on multiple levels / multiple layers as well.

Application Layer
- IP Blacklisting / Whitelisting on a Webserver
- Database-Server using stored procedures and using different database users for different permission levels
- ? …

Network Layer
- Firewalls, which block / reject certain data packets
- ? …

More Layers …?



Authorization Best Practices

In addition to all common security principles

- Keep Accounts Unique (Why?)
- Check Authorization and Permissions at every request (Why?)
- Centralizing Authorization Mechanisms (Why?)
- Mistrust everybody (Why?)
- Do not store Authorization Tokens on the client – use only serverside authorization (Why?)
- …

Authorization Attacks

1. Forceful Browsing / URL-Tampering

Consider a role-based system where the role description is part of the url

https://www.example.com/admin/dashboard
https://www.example.com/user/dashboard

- An attacker can guess the url of a page with different permission level.

Prevention?

Authorization Attacks

1. Forceful Browsing / URL-Tampering

Check the roles / permissions at every request!

Authorization Attacks 2

2. Parameter-Tampering

Consider a role-based system where the role description is part of a hidden form element

<input type=«hidden» name=«managerlevel» value=«1» />
<input type=«hidden» name=«permissionlevel» value=«1» />

- An attacker may now change these values inside the browser and just look what happens ☺

Prevention?

Authorization Attacks 2

2. Parameter-Tampering

Check the roles / permissions at every request!

Authorization Attacks 3

3. Cross Site Request Forgery (XSRF)

Consider a role-based system where the role description is part of a url

https://www.example.com/admin/dashboard
https://www.example.com/user/dashboard

- An attacker that might use a XSS-Attack vulnerability of the system to get the browsing-history of a user along with the session id.

Prevention?

Authorization Attacks 3

3. Cross Site Request Forgery (XSRF)

- When accessing a url / ressource check the client-ip and the browser-footprint in addition to the session id
- Classic XSS prevention with input filtering / validation

Lab – Role-Based Authorization

Basic Idea: Create a small and simple Role-Based Authorization Application with URL-Tampering Attack prevention.

1. Create a Login with Credentials for two usertypes
2. Create a Redirect according to the Role of the user: /User/Dashboard, /Admin/Dashboard
3. Strengthen Application with simple Role-Checks for every Request.

$\Rightarrow$ Have a look at the tutorial.

$\Rightarrow$ Info: The .NET Entity-Framework does a good job for this. But it is database driven -> we will look into that.