Webtechnologien 2

Markus Frohme

Lecture 7: 2023-05-26 – Authentication and Authorization

Schedule

Date	On Campus	Virtual	Video Release
2023-04-07	-	-	-
2023-04-14	Introduction	-	Maven
2023-04-21	Maven	-	JPA, Hibernate
2023-04-28	JPA, Hibernate	-	RESTful services
2023-05-05	RESTful services	-	-
2023-05-12	-	Backend-HelpDesk	Angular
2023-05-19	Angular	-	Angular
2023-05-26	Angular	-	**Auth & Auth**
2023-06-02	Auth & Auth	-	ALEX
2023-06-09	ALEX	-	-
2023-06-16	-	Frontend-HelpDesk	AAL
2023-06-23	AAL	-	Docker / CI / CD
2023-06-30	Docker / CI / CD	-	Security
2023-07-07	Security	-	-
2023-07-14	-	Exam-HelpDesk	-

Table of contents

Introduction

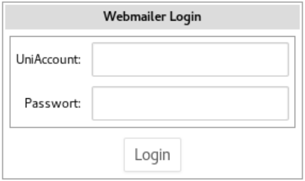


Figure: UniMail-Login

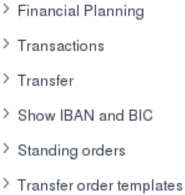


Figure: Online-Banking

```
frohme00@plutonium:~$ cat /etc/shadow
cat: /etc/shadow: Permission denied
```

Figure: “Hack” attempt

Terminology

Authentication and authorization are not exclusive to IT:

- Human-To-Human interaction (e.g., doctors, . . .)
- Human-To-Machine/Service interaction (e.g., banking terminal,. . . .)
- Machine/Service-To-Machine/Service interaction (e.g., NICs, . . .)

Generalized terminology:

- **Subject**: An entity requesting access to an object
- **Object**: An entity (document, action, . . .) being requested

Authentication and authorization...

. . . or *authentication* **vs.** *authorization*?

Identification

The decision which and the provision of information that are required for (possibly uniquely) identifying a subject

Authentication

The process of confirming the claimed identity (or a claimed property) of a subject

Authorization

The process of verifying that a subject is granted access to a requested object

Types of authentication

- Knowledge-based authentication (What do I know?)
- Property-based authentication (What do I have?)
- Identity-based authentication (What am I?)

Knowledge-based authentication

Authentication is achieved by checking the equality of a **shared secret**

- Passwords
- PINs
- Personal questions

Knowledge may be acquired by others:

- Direct attack (Cryptography)
- Sniffing, Man-in-the-middle attacks
- Social Engineering
- Brute-Force, Dictionary-Attacks

Property-based authentication

Authentication is achieved by checking the **possession of a property**

- Key(-cards)
- mobilePIN
- (Hardware-) Tokens

http://www.spiegel.de/netzwelt/web/...,
® RSA

Ownership of properties may be corrupted:

- may be lost/broken
- may be stolen
- may be duplicated

Identity-based authentication

Authentication is achieved by checking the **biometric data** of a subject

- Retina-Scanner
- Fingerprints
- Voice-/Face-Recognition
- Handwriting (forgery of documents, . . .)

General problems:

- Finding biometric data that is unique amongst subjects
- Correct extraction of biometric data

Two-factor authentication

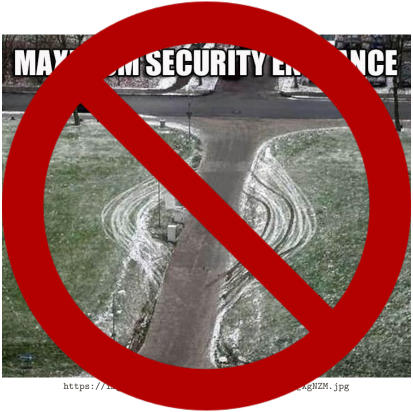
The combination of (usually two) **independent** authentication methods

- Property + Knowledge: e.g., SecurID, banking card, one-time tokens via mobile
- Identity + Property: e.g., Biometric passport

Generally a trade-off between comfort and security

Authorization

Principle of complete mediation



https://i [illegible] _XgNZM.jpg

Authorization...

...from a conceptual perspective:

- DAC (Discretionary Access Control)
- MAC (Mandatory Access Control)

...from an implementation perspective:

- RBAC (Role-based Access Control)
- PBAC (Permission-based Access control)
- ABAC (Attribute-based Access control)

Discretionary Access Control

- Access control solely based on the identity of subjects and objects
- Often objects are *owned* by a subject, that is responsible for its rights
- Can be formally defined as a relation (Access Control Matrix / Access Control List):

Object Subject	**File 1**	**Process 1**	**Printer 1**
User 1	read, write		write
User 2	read		
Process 1		block, resume	write

- Ex.: ACM(User 1, File 1) = $\{$read, write$\}$

Discretionary Access Control

Pros:

- Easy to implement / low complexity
- High configurability
- Checking access for an object is easy/fast
- Manipulating (adding, removing) access rights of objects is easy/fast

Cons:

- Fetching subject (object) rights is expensive
- Low scalability when managing changing subjects (objects)
- A lot of potential "corner-cases"

Mandatory Access Control

Access control is not discretionary but defined by system-controlled rules

There exist several access-control models that employ the mandatory paradigm:

- **Bell-LaPadula**
- Biba
- Compartment-/Lattice-Model
- . . .

Mandatory Access Control

Bell-LaPadula:

- Subjects and objects are assigned to a security classification (SC)
- top secret $>$ secret $>$ confidential $>$ unclassified
- no-read-up:
  $\forall s \in S, \forall o \in O : SC(s) < SC(o) \Rightarrow read \notin ACM(s,o)$
- no-write-down:
  $\forall s \in S, \forall o \in O : SC(o) < SC(s) \Rightarrow write \notin ACM(s,o)$
- Information may be passed to lower levels via *trusted subjects*

Mandatory Access Control

Pros:

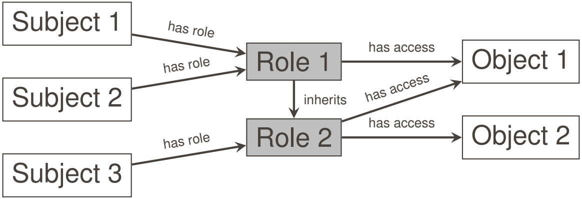
- Potential formality (provable properties)

Cons:

- Potential maintenance burden
- May be too rigorous (BLP: no write-checks possible)
- Covert channels

Role-based Access Control

Roles act as an intermediate layer between subjects and objects:



Role-based Access Control

Pros:

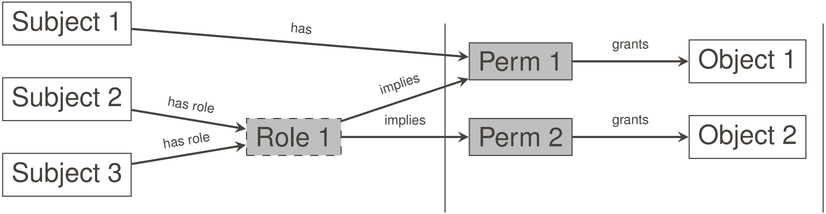
- High flexibility (1 role per subject (separation of duties), 1 role per task (principle of least privelege), role hierarchies, etc.)
- Suitable for DAC and MAC scenarios
- Eases subject management, as only roles need to be updated

Cons:

- Which roles are needed? (role engineering)
- When hard-coding roles, changes may become expensive

Permission-based Access Control

Idea: Fine-grained access control that may use other concepts to reduce its complexity



Authorization does not need to be role-aware

Attribute-based Access Control

Shifting the focus from subject-centric constraints (roles, permissions) to object-centric constraints (attributes). Allows for *dynamic* permissions.

Attribute-based access control may depend on:

- Attributes of the subject
- Attributes of the object
- Environmental variables
- Any relations between the above attributes

Access control is fully decided by the specified rules. Does this make ABAC DAC-like or MAC-like?

Apache Shiro » Terminology

- Subject:
  - Principals: data identifying the subject (name, e-mail, . . .)
  - Credentials: data verifying the subjects identity (password, biometric data, . . .)
- Realm:
  - The bridge between shiro and your application
  - e.g., LDAP-Realm if user information are stored in a LDAP directory
  - may define both authentication and authorization contracts
- Objects:
  - Can be secured by *roles* or *permissions*

Apache Shiro » Authentication

```
Subject currentUser = SecurityUtils.getSubject();

Session session = currentUser.getSession();
session.setAttribute("key", "value");

if (!currentUser.isAuthenticated()) {
        UsernamePasswordToken token =
                new UsernamePasswordToken("admin", "admin");
        try {
                currentUser.login(token);
        } catch (UnknownAccountException uae) {
                ...
        }
}

...

currentUser.logout();
```

REST advocates statelessness!

Lookup is defined in the realm

Apache Shiro » Authorization

```
if (currentUser.hasRole("role")) { /*#checkRole()*/
        //do role-specific stuff
}

if (currentUser.isPermitted("<permission>")) { /*#checkPermission()*/
        //do permission-specific stuff
}
```

Permission syntax: `domain:action:instance`

- `printer:print:lp7200`
- `printer:print,query:*`

Checks for logical implications:

- `printer:*:lp7200` implies `printer:print:lp7200`

Apache Shiro » Annotations

```
@RequiresAuthentication
@RequiresPermission("printer:print:*")
public void print() {...}

@RequiresAuthentication
@RequiresRole("admin")
public void manage() {...}

@RequiresGuest
public void printHelp() {...}
```

- For applications (on-/offline), bytecode-weaving (e.g., AspectJ,. . . .) is required – (may be buggy sometimes)
- Web-apps may configure access control based on URLs

Apache Shiro » Configuration via shiro.ini

```
[main]
wt2Realm = de.ls5.wt2.auth.WT2Realm
securityManager.realms = $wt2Realm

credMatcher = org.apache.shiro.authc.credential.Sha256CredentialsMatcher
wt2Realm.credentialsMatcher = $credMatcher

authc.loginUrl = /login.jsp
authc.successUrl = /
logout.redirectUrl = /
...
[user]
...
[roles]
...
[urls]
/rest/** = authcBasic, roles["admin"], perms["rest:*"]
/pages/** = authc
/logout = logout
/** = anon
```

Endpoint needs to be able to process POST requests

first match wins

Apache Shiro » Configuration via web.xml

```
<listener>
    <listener-class>
        org.apache.shiro.web.env.EnvironmentLoaderListener
    </listener-class>
</listener>
<filter>
    <filter-name>shiroFilter</filter-name>
    <filter-class>
        org.apache.shiro.web.servlet.ShiroFilter
    </filter-class>
</filter>
<filter-mapping>
    <filter-name>shiroFilter</filter-name>
    <url-pattern>/*</url-pattern>
    <dispatcher>REQUEST</dispatcher>
    <dispatcher>FORWARD</dispatcher>
    <dispatcher>INCLUDE</dispatcher>
    <dispatcher>ERROR</dispatcher>
</filter-mapping>
```

Ensures complete mediation for HTTP requests

Example

- Ex. 6 (cont.): A rudimentary shiro configuration, that authenticates users via either
  - sessions,
  - basic authentication or
  - JWTs

  and controls access to system resources by using
  - roles and
  - (attributed) permissions.

References

- DAC – B. W. Lampson: Protection
- MAC – D.E. Bell, and L.J. LaPadula: Secure Computer Systems
- RBAC – D. Ferraiolo, R. Kuhn: Role-Based Access Control
- RBAC + ABAC R. Kuhn, E. Coyne, T. Weil: Adding Attributes to Role-Based Access Control
- Apache Shiro – https://shiro.apache.org/