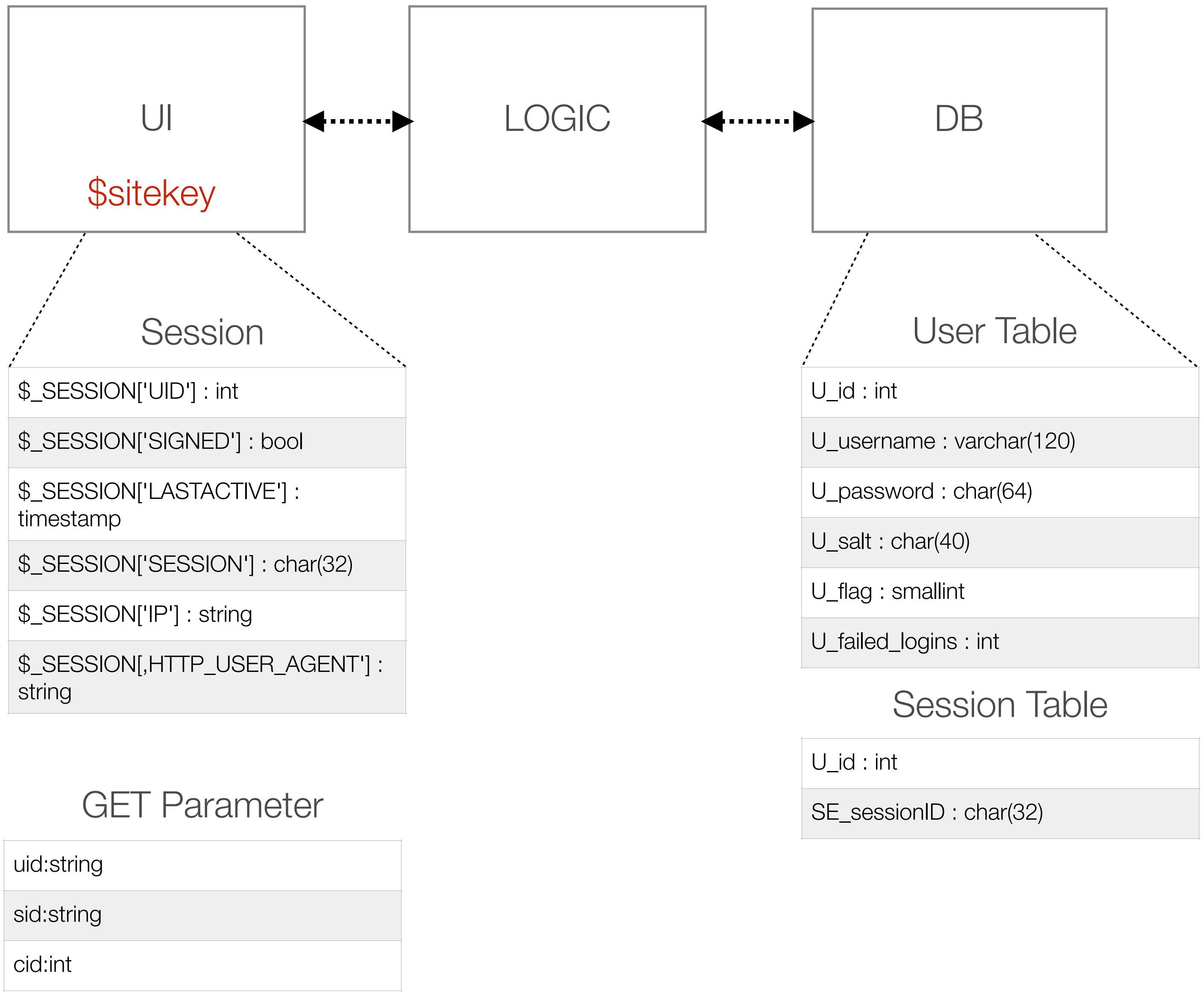
Security System

Documentation

19.03.2013

Security Data in all 3 layers



Session

$_SESSION['UID'] : int
$_SESSION['SIGNED'] : bool
$_SESSION['LASTACTIVE'] : timestamp
$_SESSION['SESSION'] : char(32)
$_SESSION['IP'] : string
$_SESSION[,HTTP_USER_AGENT'] : string

GET Parameter

uid:string
sid:string
cid:int

User Table

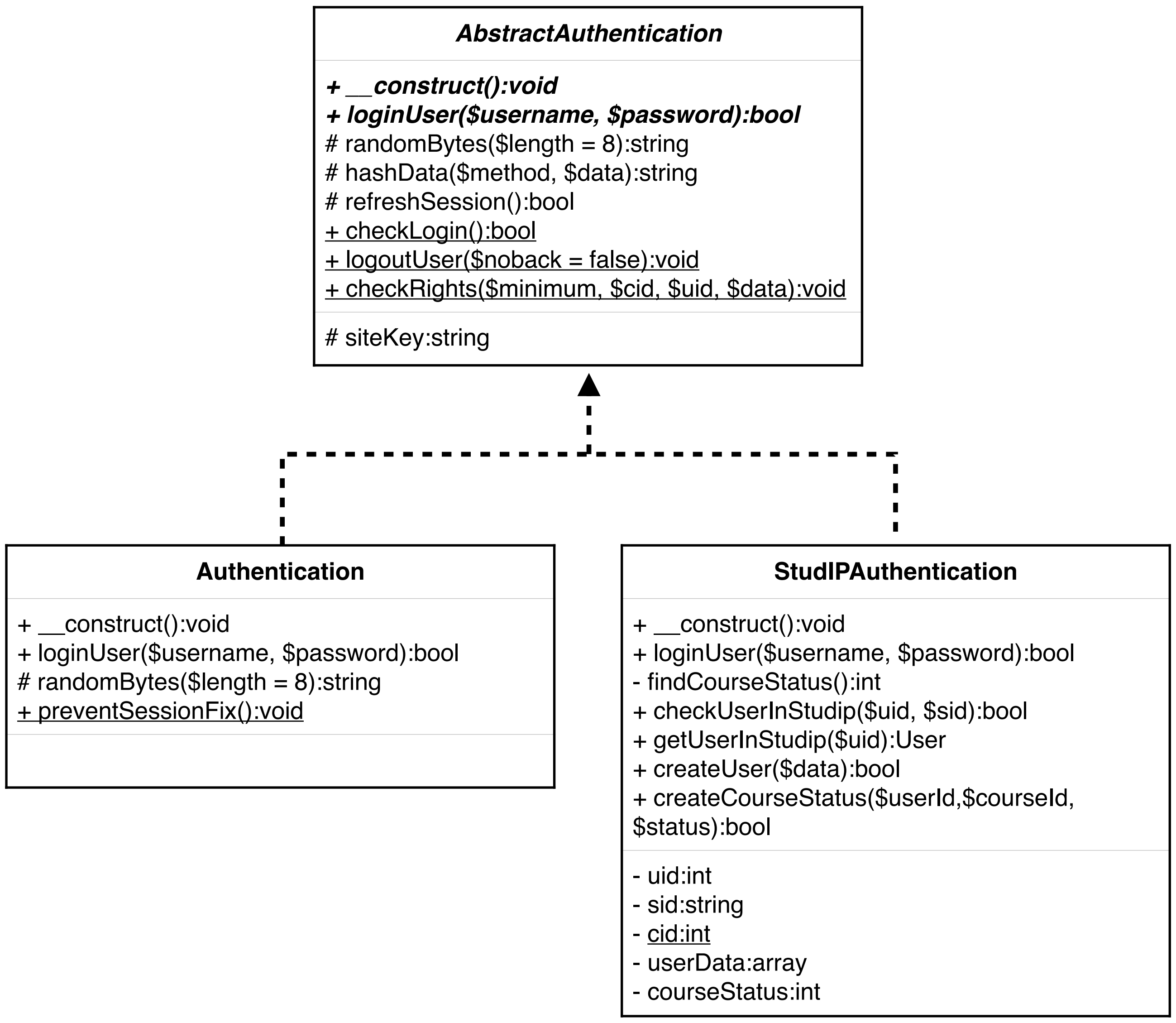
U_id : int
U_username : varchar(120)
U_password : char(64)
U_salt : char(40)
U_flag : smallint
U_failed_logins : int

Session Table

U_id : int
SE_sessionID : char(32)

All important variables for the security system are shown in the diagram above.
Note:
Course related data is not depicted. For more information about the course structure, please read the database documentation.

Class Diagram


AbstractAuthentication
+ __construct():void
+ loginUser($username, $password):bool
# randomBytes($length = 8):string
# hashData($method, $data):string
# refreshSession():bool
+ checkLogin():bool
+ logoutUser($noback = false):void
+ checkRights($minimum, $cid, $uid, $data):void
# siteKey:string
Authentication
+ __construct():void
+ loginUser($username, $password):bool
# randomBytes($length = 8):string
+ preventSessionFix():void
StudIPAuthentication
+ __construct():void
+ loginUser($username, $password):bool
- findCourseStatus():int
+ checkUserInStudip($uid, $sid):bool
+ getUserInStudip($uid):User
+ createUser($data):bool
+ createCourseStatus($userId,$courseId, $status):bool
- uid:int
- sid:string
- cid:int
- userData:array
- courseStatus:int

key: + public, **-** private, **#** protected, **bold**: abstract method, **underlined**: static

Concept

The Concept is modular based like all other components. That means, that additional login systems like ILEAS can be added in the future. To hook up a system simply inherit a class from ***AbstractAuthentication*** and implement __construct() and loginUser($username, $password). Look in UI/include/Authorization.php to see how the authentication classes are used.

All 3 layers are secured by this system. At the beginning a user have to log in by the given methods defined in our classes (for now there are independent login and studIP login). After that, a session is created on the UI layer and then created in DB layer. Only if both goes right, the user will be logged in correctly.

All REST API calls from UI layer redirects User (which is equal to $_SESSION['UID']), Session (which is a SessionID equal to $_SESSION['SESSION']) and DATE (which is equal to $_SESSION['LASTACTIVE']) in the header for authorization. Therefore unauthorized REST access is prevented. These header fields are redirected in all logic components. The DB layer checks the header every time a DB request is invoked over the DBControl. Date have to be within a 10 minute interval and Session have to be linked to User in DB.

AbstractAuthentication

AbstractAuthentication is the base class for all login systems like the regular one or studIP login.

Attributes

siteKey

The siteKey is a global security key for the whole platform. It is not allowed to change this, after the system is already running. Otherwise, all hashes would be wrong.

Methods

randomBytes method

This method generates a random string.

hashData method

This method hashes a string with a given hash method.

refreshSession method

This method creates all necessary session data fields like $_SESSION['UID'] etc. It also creates the Session in the DB for securing the REST API. It will return true if session is created in DB and UI correctly.

checkLogin method

This method is static and checks if an user is successfully logged in in our system. Therefore, a session have to be created. A user is logged in if the following conditions are met:

- SIGNED flag is true
- LASTACTIVE timestamp is in the valid 10 minutes interval
- HTTP_USER_AGENT isn't changed after initial login
- IP isn't changed after initial login

After these checks the method returns true and LASTACTIVE is updated.

logoutUser method

This method deletes the session in UI layer and DB. If $noback is set to true, this method will save the requested page in the URL as a GET parameter. Therefore, a user is redirected to the right page, saved in back, after a valid login.

checkRights method

Checks the Rights of a User. If the user does not have the rights required to view the page, it will redirect to index.php with error message 403. Right management is only implemented in UI layer. So keep in mind to check it in further development.

Authentication

The authentication class provides the normal login functionality by entering username and password in Login.php. This class have to be implemented. All the other login systems, which are hooked up, have to be instantiated after an Authentication instance (see Authorization.php) and are optional.

Methods

constructor method

The constructor method starts the session in php. That is the reason why it have to be instantiated before all other login systems. It forces to use cookies for the session and forbids transmitting SID over URL.

preventSessionFix method

Prevents possible session fixation attack by checking if the given session id comes from our platform. If it is randomly inserted in a cookie, regenerate a session id.

loginUser method

At first, it checks if a user is stored in the DB layer. If the answer is not a 404 status and a user object is returned, a salt will be read from this object. This salt is suffixed to the plaintext password (comes form the login form). After that, the new string is hashed with SHA256 and is compared with the password hash from the user object. If they are equal, the $_SESSION['UID'] is stored with the given U_id in the user object. Afterwards, the refreshSession method is called.
If the login failed, a GET request to the component /user/:userid/IncFailedLogin will be invoked. In this way, the field U_failed_logins for the user :userid in DB is increased. Therefore, brute-force attacks are prevented.

StudIPAuthentication

The primary function of this class is to bridge all users from StudIP to our system. It used the same API calls like the old “uebungsplattform 1.0“ which is an in-house production. The use of the official StudIP API is supported in the near future.

Attributes

uid

The UserID from studIP.

sid

The SessionID from studIP. It is not equal to our platform session.

cid

The CourseID from our platform.

userData

The user data which is gotten from the database.

courseStatus

The course status from the given user.

Methods

constructor method

The constructor method logs the user in if uid, cid and sid is given in GET Parameters.

findCourseStatus method

Finds the correct course status for given cid in the userData attribute.

checkUserInStudip method

Checks if user is logged in in StudIP. It uses the old studIP API check_user.

getUserInStudip method

Gives user data from StudIP. It uses the old studIP API get_user. It returns a new user object structure compatible with our DB.

createUser method

Creates User in DB.

createCourseStatus method

Adds course to an user.

loginUser method

It calls checkUserInStudip method to check if user is logged in in studIP. If the user already exists in our platform, the UID is stored in our Session and refreshSession method is called. After that, the course status is searched by invoking the findCourseStatus method. If there is no status, a new one will be created with the createCourseStatus method. The user will be added as student. If there is no user for the given studIP user, a new user is added to our system by invoking createUser and loginUser is called again.
This method only will return true if the process above is going right.

Triggers

- A session only will be created if U_flag is equal to 1.
- If U_flag unequal to 1, session will be deleted from DB.
- If U_failed_logins >= 10, account will be disabled (U_flag=2).
- If session is created in DB, U_failed_logins will be reset.
- If session already exists in DB, it will be overwritten. That means that there is always just one or no session for a user.

By the use of these triggers which are implemented in DB layer, a brute-force attack is prevented and the possibility that two persons are logged in with the same username.