Contents

[Why Are Hashes Irreversible? 2](#_Toc433281113)

[Password Salting 2](#_Toc433281114)

[Scenarios 5](#_Toc433281115)

[**Web server applications** 5](#_Toc433281116)

[Token expiration 6](#_Toc433281117)

[Personal notes: 6](#_Toc433281118)

[Friday, September 04, 2015 6](#_Toc433281119)

[DATABSE CONNECTION: CHAPTERS TO BUILD BUDGET MANAGEMENT DATABASE 9](#_Toc433281120)

[Thursday, September 17, 2015 9](#_Toc433281121)

[**Sunday, October 11, 2015 : Php value to jquery:** 10](#_Toc433281122)

[Thursday, October 15, 2015 11](#_Toc433281123)

[SENDING MAIL FROM XAMPP 13](#_Toc433281124)

LOL JUST FANCY STUFF

STOP STARING

Vigneshvk

2015

Aug 28:Create login page: login.html

1. August 28: Sql database in xampp:

CREATE TABLE WebsiteUsers ( userID int(9) NOT NULL auto\_increment, fullname VARCHAR(50) NOT NULL, userName VARCHAR(40) NOT NULL, email VARCHAR(40) NOT NULL, pass VARCHAR(40) NOT NULL, PRIMARY KEY(userID) );

There are 4 methods of form action type: Get post delete put. Usually what is used is Get but post is a better way of implementing forms.

We are using POST method because it is secure. If we use GET method our data will not private. POST is commonly being used for these kind of webpages, due to its security. And when we are dealing with our ID and Password, so we should be alert from the hackers. Because the GET is being used in Phishing webpages, by which people are making fool the internet users to get their personal ID and Password.

Do not use the Google IDs returned by **getId()** or the user's profile information to communicate the currently signed in user to your backend server. Instead, [send ID tokens](https://developers.google.com/identity/sign-in/web/backend-auth), which can be securely validated on the server.(even mentioned in google sign in option instructions).

1. August 30:

Self observations: Submit type should be mentioned “submit” . Don’t use input id and input name both . Causes confusion. In some cases of php sql communication. Like in case of before (if) condition the sql die method should not be used as the query stops processor so there is no way to find out what did happen. Alternatively we should find a method to get query results displayed to the user.

September 1:

Changed password varchar to 64 to try sha 256 encryption method

Sha 256 encryption for password during registration ready.

//need to check for sql injection prevention

//need to check password matching

//need to show user if username is repeated or email is repeated.

Tried php password checking decided we are better off with password checking BEFORE sending to php. i.e using javascript

Scrapped the javascript idea. Instead usehtml 5 for creating password constraints. Now will try to use same for comparing passwords.

Ended up using a combination of both html constraints and javascript to check username and password issues.

Thursday, October 22, 2015

## [Why Are Hashes Irreversible?](http://learncryptography.com/why-are-hashes-irreversible/)

A common question many people have upon learning about one-way cryptographic hash functions is how the “one-way” part works. A certain amount of data goes through a process and a result is given – shouldn’t it be possible to do that same process in reverse, since we know what the process is in the first place?

Well there’s a very good reason – and it applies mostly the same to all hash functions, even though they are all implemented differently. The answer is the modulo operator. For a quick review, modulus is essentially the same as saying “the remainder of” (applying to division). A quick example would be:

16\,mod\,5 = 1

What’s happening here is that we’re dividing 16 by 5, and the result of the operation is whatever is left over – or the remainder. One more:

24\,mod \,7 = 3

Obviously if the number was evenly divisible my the modulo then it would result in zero, and if it was less than the number it would remain as the number itself. Now – back to the important part. Why is this so important in one-way hash functions? Because the modulo operation is not reversible. If the result of the modulo operation is 4 – that’s great, you know the result, but there are infinite possible number combinations that you could use to get that 4.

Another thing to consider is that a lot of data is discarded during the hash process. While the input string may be as big as it wants, the output must always be of a set size determined by the function.  Because of this, a lot of data is thrown out. It would be impossible to figure out the original data of the function with just the resulting hash – as not much of that data is left – the only workable method is to brute force every possible combination. If we could reverse a hash, we would be able to compress data of any size into a mere few bytes of data!

## [Password Salting](http://learncryptography.com/password-salting/)

Password salting is the process of securing password hashes from something called a Rainbow Table attack. The problem with non-salted passwords is that they do not have a property that is unique to themselves – that is, if someone had a precomputed rainbow table of common password hashes, they could easily compare them to a database and see who had used which common password.

So how do we make each hashed password in a database unique? We add something called a salt to the input to the hash function. Usually the salt is some random data stored with the user in the database, but it could be something else unique to the user such as their login name or registration date. Why is this effective?

Hash(password + salt)

Every user now has something that is unique to them, that is added on to their password before it is hashed and stored in the database. Now, if someone were to try the same rainbow table attack with a list of common password hashes – none of the hashes would match. The salts change the output of the hash function completely, so even the common passwords are safe. If an attacker wanted to crack the passwords now, they would have to add the salt for each individual user – increasing the amount of time required to brute force all of the passwords exponentially.

Thursday, October 22, 2015

$\_SESSION [ ] is a global variable in php . During login. From the users end to keep the user “logged in “ on every page he traverses , a cookie is set. And from the servers end, this global variable is used. Basically can be used as a flag that the user has logged in. once logged out this variable is made empty again

$row is also another global ARRAY variable. Its used to store sql query results in the form of an array

And some imp point I feel like he has told but I cant rmmbr right now. Anyway it stores the query

$row = mysql\_fetch\_array($query) using that. To authenticate user login further u check if $row is empty and then the login session happens.

Also u always need to session\_start(); for login and session end or destroy for lgout . this sets and unsets global variables.

These two are called as session variables. Kbye.

//SIGN IN WITH GOOGLE SHIZ

All applications follow a basic pattern when accessing a Google API using OAuth 2.0. At a high level, you follow four steps:

#### **1. Obtain OAuth 2.0 credentials from the Google Developers Console.**

Visit the [Google Developers Console](https://console.developers.google.com/) to obtain OAuth 2.0 credentials such as a client ID and client secret that are known to both Google and your application. The set of values varies based on what type of application you are building. For example, a JavaScript application does not require a secret, but a web server application does.

#### **2. Obtain an access token from the Google Authorization Server.**

Before your application can access private data using a Google API, it must obtain an access token that grants access to that API. A single access token can grant varying degrees of access to multiple APIs. A variable parameter called scope controls the set of resources and operations that an access token permits. During the access-token request, your application sends one or more values in the scope parameter.

There are several ways to make this request, and they vary based on the type of application you are building. For example, a JavaScript application might request an access token using a browser redirect to Google, while an application installed on a device that has no browser uses web service requests.

Some requests require an authentication step where the user logs in with their Google account. After logging in, the user is asked whether they are willing to grant the permissions that your application is requesting. This process is called user consent.

If the user grants the permission, the Google Authorization Server sends your application an access token (or an authorization code that your application can use to obtain an access token). If the user does not grant the permission, the server returns an error.

It is generally a best practice to request scopes incrementally, at the time access is required, rather than up front. For example, an app that wants to support purchases should not request Google Wallet access until the user presses the “buy” button; see [Incremental authorization](https://developers.google.com/accounts/docs/OAuth2WebServer#incrementalAuth).

#### **3. Send the access token to an API.**

After an application obtains an access token, it sends the token to a Google API in an HTTP authorization header. It is possible to send tokens as URI query-string parameters, but we don't recommend it, because URI parameters can end up in log files that are not completely secure. Also, it is good REST practice to avoid creating unnecessary URI parameter names.

Access tokens are valid only for the set of operations and resources described in the scope of the token request. For example, if an access token is issued for the Google+ API, it does not grant access to the Google Contacts API. You can, however, send that access token to the Google+ API multiple times for similar operations.

#### **4. Refresh the access token, if necessary.**

Access tokens have limited lifetimes. If your application needs access to a Google API beyond the lifetime of a single access token, it can obtain a refresh token. A refresh token allows your application to obtain new access tokens.

**Note:**Save refresh tokens in secure long-term storage and continue to use them as long as they remain valid. Limits apply to the number of refresh tokens that are issued per client-user combination, and per user across all clients, and these limits are different. If your application requests enough refresh tokens to go over one of the limits, older refresh tokens stop working.

## Scenarios

### **Web server applications**

The Google OAuth 2.0 endpoint supports web server applications that use languages and frameworks such as PHP, Java, Python, Ruby, and ASP.NET.

The authorization sequence begins when your application redirects a browser to a Google URL; the URL includes query parameters that indicate the type of access being requested. Google handles the user authentication, session selection, and user consent. The result is an authorization code, which the application can exchange for an access token and a refresh token.

The application should store the refresh token for future use and use the access token to access a Google API. Once the access token expires, the application uses the refresh token to obtain a new one.

Your application sends a token request to the Google Authorization Server, receives an authorization code,
exchanges the code for a token, and uses the token to call a Google API endpoint.

.

## Token expiration

You should write your code to anticipate the possibility that a granted token might no longer work. A token might stop working for one of these reasons:

* The user has revoked access.
* The token has not been used for six months.
* The user account has exceeded a certain number of token requests.

There is currently a 25-token limit per Google user account. If a user account has 25 valid tokens, the next authentication request succeeds, but quietly invalidates the oldest outstanding token without any user-visible warning.

If you need to authorize multiple programs, machines, or devices, one workaround is to limit the number of clients that you authorize per user account to 15 or 20. If you are a [Google Apps admin](https://support.google.com/a/), you can create additional admin users and use them to authorize some of the clients.

Personal notes: //without backend server authentication wala code.

* In google developers page make sure u use localhost:80 (mention also the port in both javascript origins and redirect page)
* <https://developers.google.com/identity/sign-in/g-normal.png> use this to get the “g” of the google. Site users use from /identity as google is mainsite in the google webpage.

Or make ur own shitty button.

# Friday, September 04, 2015

New table created

Timestamp section added

Especially for token purposes

CREATE TABLE `users` (

`id` *int*(**11**) NOT NULL AUTO\_INCREMENT,

`oauth\_provider` *varchar*(**255**) COLLATE utf8\_unicode\_ci NOT NULL,

`oauth\_uid` *varchar*(**255**) COLLATE utf8\_unicode\_ci NOT NULL,

`fname` *varchar*(**255**) COLLATE utf8\_unicode\_ci NOT NULL,

`lname` *varchar*(**255**) COLLATE utf8\_unicode\_ci NOT NULL,

userName VARCHAR(40) NOT NULL,

pass VARCHAR(40) NOT NULL

`email` *varchar*(**255**) COLLATE utf8\_unicode\_ci NOT NULL,

`gender` *varchar*(**10**) COLLATE utf8\_unicode\_ci NOT NULL,

`locale` *varchar*(**10**) COLLATE utf8\_unicode\_ci NOT NULL,

`gpluslink` *varchar*(**255**) COLLATE utf8\_unicode\_ci NOT NULL,

`picture` *varchar*(**255**) COLLATE utf8\_unicode\_ci NOT NULL,

`created` datetime NOT NULL,

`modified` datetime NOT NULL,

*PRIMARY KEY* (`id`)

) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8\_unicode\_ci;

Observations:

Backend servers require API’s from github. Each website google, twitter,linkedin,facebook have the repository in the website.

They need to be downloaded and used for the php files.

The database connection and query sent should contain values that are a possibility of being received from the server like first name last name email auth data etc. example

: $insert = mysqli\_query($this->connect,"INSERT INTO $this->tableName SET oauth\_provider = '".$oauth\_provider."', oauth\_uid = '".$oauth\_uid."', fname = '".$fname."', lname = '".$lname."', email = '".$email."', gender = '".$gender."', locale = '".$locale."', picture = '".$profile\_image\_url."', gpluslink = '".$link."', created = '".date("Y-m-d H:i:s")."', modified = '".date("Y-m-d H:i:s")."'") or die(mysqli\_error($this->connect));

}

Make sure u have your redirect url correctly mentioned. Redirect url at the developers console. So when the user is authenticated the php script in ur send will have a redirect url which should be mentioned in the developers console else it does not work properly.

Keep track with the table.

Make sure u revoke token access from logout.

For some reason the port of the xampp localhost also plays a role in mentioning in the developers console. (google)

Imma sleep now gnite.

Saturday, September 12, 2015

<https://accounts.google.com/o/oauth2/auth?response_type=code&redirect_uri=http%3A%2F%2Flocalhost%2Fmoneymap%2Fphp%2Flogin%2FGoogle&client_id=886700361912-cj68r4tdc73flrluhunjccrn85u1cqi8.apps.googleusercontent.com&scope=https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fuserinfo.profile+https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fuserinfo.email&access_type=offline&approval_prompt=force>

## DATABSE CONNECTION: CHAPTERS TO BUILD BUDGET MANAGEMENT DATABASE

# Thursday, September 17, 2015

So I found a slider in jquery

$(document).ready(function() {

$("#slider").slider({

animate: true,

value:1,

min: 0,

max: 100000,

step: 100,

slide: function(event, ui) {

update(1,ui.value); //changed

}

});

Document ready function checks the html form after its loaded.

On every change it updates the value of the variable.

$total = "$" + ($amount \* $duration/100);

$( "#total" ).val($total);

Now this statement . After jquery value is been calculated in $total. The #total is a form field in the html page which is hidden. After this is stored in the textbox area. When the submit button is pressed it can be sent to php and php can recieeve using $\_POST[xyz] method.

Side note to comment I figured this way out myself as in I thought about it myself. But I was stupid enough to not see that this method is already there in the slider template provided by the designer. But im quite proud I was thinking right so yay? Ok.

### **Sunday, October 11, 2015 : Php value to jquery:**

You have two or three options: if the Javascript is in the php file, you can

var phpVar = <?php echo $var; ?>;

Otherwise if the Javascript is anywhere at all, you can do:

<input type="hidden" id="phpVar" value="<?php echo $var; ?>">

and then access it as

$('#phpVar').val();

$month = date("m");

$query="SELECT \* FROM `income` WHERE `id`='$id' AND MONTH(date) = '$month'";

$result=mysql\_query($query) or die(mysql\_error());

$row = mysql\_fetch\_array($result) or die(mysql\_error());

$income= $row['income'];

echo "$income";

Month(date) is the sql thing for getting only month in the date.

And date(“m”) is to get current date in system.

To set budget in form variable. Separate one to show text.

$( "#budget" ).val($budget);

$( "#budget-label").text($budget);

$query2="UPDATE `budget` SET `budget`='$budget',`Saving`=655 WHERE `id`=3 AND month(date)='$month'";

FINALLY FINISHED THE BUDGET SETTING FRONT END BLOODY BULLSHIT HELL. Ashwins savings.html sucked ass jquery wasn’t linked well. Some characters were not seen. Butttt his common.html became useful as expected by mahhh advice and I was able to properly implement.. ☺

## Thursday, October 15, 2015

<?php

function sift($cat)

{

$catarray=array("Food","Travel","Party","Misc");

$catname=$catarray[$cat-1];

echo $catname;

}

?>

Separate function created for category frontend conversion

Added footer

Added dual date range

DATE function like MONTH is used to only consider date without time

$date1 =str\_replace("/","-",$\_POST['date1']);

$date1=str\_replace(":","",$date1);

$date1=str\_replace(" ","",$date1);

$month1=substr($date1,4,2);

echo $date1;

echo "-------";

$date2 =str\_replace("/","-",$\_POST['date2']);

$date2=str\_replace(":","",$date2);

$date2=str\_replace(" ","",$date2);

$month2=substr($date2,4,2);

echo $date2;

echo "-------";

$query="SELECT \* FROM `expense` WHERE `id`=3 AND DATE(transid) >= '$date1' AND DATE(transid) < '$date2'";

$result=mysql\_query($query)or die(mysql\_error());

$row=mysql\_fetch\_array($result);

$cat=$row['category'];

echo $cat;

sift($cat);

INSTEAD OF THIS SHIT

while($row=mysql\_fetch\_array($result))

{

echo "<tr>";

echo "<td>"; echo $row['id']; echo "</td>";

echo "<td>"; echo $row['date']; echo "</td>";

echo "<td>"; echo $row['category']; echo "</td>";

echo "<td>"; echo $row['expense']; echo "</td>";

echo "</tr>";

}

USE THIS VERY CLEVERLY MADE USE OF PHP’s dissassoiativeness

<?php

while($row = mysql\_fetch\_array($result)) {

?>

<tr>

<td><?php echo $row['date']?></td>

<td><?php echo $row['category']?></td>

</tr>

<?php

}

?>

## SENDING MAIL FROM XAMPP

You can send mail from localhost with sendmail package , sendmail package is inbuild in XAMPP. So if you are using XAMPP then you can easily send mail from localhost.

for example you can configure C:\xampp\php\php.ini and c:\xampp\sendmail\sendmail.ini for gmail to send mail.

in C:\xampp\php\php.ini find extension=php\_openssl.dll and remove the semicolon from the beginning of that line to make SSL working for gmail for localhost.

in php.ini file find [mail function] and change

ALSO DISABLE MAILTODISK

SMTP=smtp.gmail.com

smtp\_port=587

sendmail\_from = my-gmail-id@gmail.com

sendmail\_path = "\"C:\xampp\sendmail\sendmail.exe\" -t"

Now Open C:\xampp\sendmail\sendmail.ini. Replace all the existing code in sendmail.ini with following code

[sendmail]

smtp\_server=smtp.gmail.com

smtp\_port=587

error\_logfile=error.log

debug\_logfile=debug.log

auth\_username=my-gmail-id@gmail.com

auth\_password=my-gmail-password

force\_sender=my-gmail-id@gmail.com

Now you have done!! create php file with mail function and send mail from localhost.

PS: don't forgot to replace **my-gmail-id** and **my-gmail-password** in above code. Also, don't forget to remove duplicate keys if you copied settings from above. For example comment following line if there is another **sendmail\_path** : sendmail\_path="C:\xampp\mailtodisk\mailtodisk.exe" in the php.ini file

Also remember to restart the server using the XAMMP control panel so the changes take effect.

[sendmail]

; you must change mail.mydomain.com to your smtp server,

; or to IIS's "pickup" directory. (generally C:\Inetpub\mailroot\Pickup)

; emails delivered via IIS's pickup directory cause sendmail to

; run quicker, but you won't get error messages back to the calling

; application.

smtp\_server=smtp.gmail.com

smtp\_port=587

error\_logfile=error.log

debug\_logfile=debug.log

mail:tls=true

mail.smtp.auth=true

mail.smtp.starttls.enable=true

auth\_username=moneymaphelp@gmail.com

auth\_password=moneymap123

[force\_sender=moneymaphelp@gmail.com](mailto:force_sender=moneymaphelp@gmail.com)

[mail function]

; XAMPP: Comment out this if you want to work with an SMTP Server like Mercury

SMTP=smtp.gmail.com

sendmail\_path = "\"G:\xampp\sendmail\sendmail.exe\" -t"

; For Win32 only.

; http://php.net/sendmail-from

;sendmail\_from = postmaster@localhost

; XAMPP IMPORTANT NOTE (1): If XAMPP is installed in a base directory with spaces (e.g. c:\program filesG:\xampp) fakemail and mailtodisk do not work correctly.

; XAMPP IMPORTANT NOTE (2): In this case please copy the sendmail or mailtodisk folder in your root folder (e.g. C:\sendmail) and use this for sendmail\_path.

; XAMPP: Comment out this if you want to work with fakemail for forwarding to your mailbox (sendmail.exe in the sendmail folder)

;sendmail\_path = "\"G:\xampp\sendmail\sendmail.exe\" -t"

; XAMPP: Comment out this if you want to work with mailToDisk, It writes all mails in the G:\xampp\mailoutput folder

;sendmail\_path="G:\xampp\mailtodisk\mailtodisk.exe"

; Force the addition of the specified parameters to be passed as extra parameters

; to the sendmail binary. These parameters will always replace the value of

; the 5th parameter to mail(), even in safe mode.

;mail.force\_extra\_parameters =

; Add X-PHP-Originating-Script: that will include uid of the script followed by the filename

mail.add\_x\_header=Off

; Log all mail() calls including the full path of the script, line #, to address and headers

mail.log = "G:\xampp\php\logs\php\_mail.log"

<?php

$to = "destinationAddress@Anywhere.com";

$subject = "The mail subject goes here";

$content= "And this is the mail content!";

$headers = "From:Me@Mywebsite.Com\r\n";

mail($to,$subject,$content,$headers);

?>

# Email Verification function:

So theoretically what I found out was, U get an activation link and that’s emailed to u. when clicked ur email gets activated.

For this, the table has an added column “activation” of 40 varcahar.

Your email address is hashed using md5 function.

And the hashed value is stored in activation table.

The activation link will have following format:

<http://website.com/activate.php?email=admin@example.com&key=c47662ba2504508bcdd5cb75106110a6>

Here what happens is the ones after “php?” Are stored for php $\_GET variables. Post variables are encrypted whereas get variables are not. The email will check the current email and key will have hashed value of the key. So ok. Bye. Tired.

# $\_GET USAGE:

# $Message="Sorry You are already registered User" header("Location: index.php?Message=" . urlencode($Message));

And in the redirected page capture it using :

if (isset($\_GET['Message'])) {

print '<script type="text/javascript">alert("' . $\_GET['Message'] . '");</script>';

}

Or you could store php variable

Use bootstraps alert division to display the message

Enable Disable the division using the script:

<script type="text/javascript">

var msg=<?php echo $msg; ?>;

var div = document.getElementById('alerty');

//window.alert(msg);

if(msg=='0')

div.style.visibility = 'hidden';

</script>