**The Webb-site login system and mailvote schema**

The login system is part of a schema called mailvote, which resides on the Webb-site server. The mailvote schema also includes storage for the Webb-site opinion polling system.

For obvious reasons, we are only dumping the structure of mailvote and not its data. The relevant table is livelist, which we use to drive our newsletter system. A user can login, change their email address or password, and turn the newsletter on/off. They can also maintain a list of stocks which they follow. They can also volunteer to edit the database, in which case they must choose a username. See the separate file [NAME] for that.

**Creating a Webb-site account**

User is identified by the mailAddr column in livelist and a salted hashed password. We don’t collect any personal data on the user – they can use an obscure gmail account (as many do).

|  |  |
| --- | --- |
| salt | A 16-byte binary created with SELECT MD5(RAND()). That produces a 32-character hex string which is stored in the binary salt column as UNHEX(salt) |
| hash | A 32-byte (256-bit) binary using SHA-256 and appending the salt to the password so that 2 users with the same password will not have the same hash. Also, a dictionary attack with passwords will not be easy. Created with:  UNHEX(SHA2(CONCAT(‘apos(pwd)&”’,salt),256) |
| URLencode | A MySQL function in mailvote which takes a binary blob and converts it into a BASE64 string. BASE64 has 64 possible values in 6 bits [A-Z][a-z][0-9][+/]. Therefore, a 16-byte (128 bit) MD5 becomes 22 characters. In URL coding, “+” is replaced with “-“ and “/” is replaced with “\_” otherwise it would cause problems with addresses. |
| token | Used to verify email addresses. Generated with MySQL function in mailvote genToken():  LEFT(URLencode(UNHEX(MD5(RAND()))),22) |
| etokHash | A 32-byte binary column which is the SHA-256 hash of the token, so:  UNHEX(SHA2(token,256)) |
| etokTime | The timestamp on the token, so that we can invalidate the tokens after a set time (72 hours = 4320 minutes in our script) using TIMESTAMPDIFF(MINUTE,etokTime,NOW())>=4320 |
| eVerified | Boolean column, whether the address has been verified. |

When a new user creates an account, an email is sent with the token in a link to our verify.asp page:

[https://webb-site.com/webbmail/verify.asp?t="&token](https://webb-site.com/webbmail/verify.asp?t=%22&token)

The verify page then receives the token, searches for its SHA-256 hash in the user-table, and activates the account by setting eVerified=True if the token is not expired.

**Logging in**

The password is verified as follows:

Hash=UNHEX(SHA2(CONCAT(‘password’,LOWER(HEX(salt))),256)

The password is temporarily stored in Session(“pwd”), so that it can be used when accessing MySQL via conMaster (direct, rather than via a wsrole) for DavidOnline and Cynthia.

If the user selects a “keep me logged in” (for up to 30 days) then we generate a token for their browser using genToken and store a hash of it in our “persist” table:

|  |  |
| --- | --- |
| tokID | Primary key, integer autoincrement (removed 2024-05-02, using tokhash as Primary Key instead) |
| tokhash | 32-byte binary, SHA-256 hash of the token |
| tokTime | Timestamp for expiry of token |
| userID | The userID to which the token belongs |

We put the token in a cookie collection called “keep”:

Response.Cookies(“keep”)(“token”)=token

We set Response.Cookies(“keep”).Expires to the time at which the cookies expire, and require SSL:

Response.Cookies(“keep”).Secure=True

When a user revisits the site, cookiechk.asp takes a HASH of the token and compares it with the stored hashed tokens in mailvote.persist.

For dbexec users (DavidOnline, Cynthia), we need to recover their password from the cookie/token, without including the password in the cookie, as that would be readable.

Use the token as a keystring and store AES\_ENCRYPT (pwd,keystring) in mailvote.persist column cred as a BLOB. Then when the token comes back, recover password with:

CAST(AES\_DECRYPT(cred,token) AS CHAR)

**Clearing dead tokens**

Every time someone hits the login.asp page, it runs a delete query to purge dead tokens for all users:

mailDB.Execute "DELETE FROM mailvote.persist WHERE tokTime<Now()"

Note: tokhash has a very low chance of collisions on INSERT as there are about 10^15 possible values of RAND (based on the precision we see).

There could be a speed impact if the index was huge as the column is 32 bytes (256-bit hash) but that seems unlikely.

**Checking that user is logged in**

Each user page starts by calling Sub login() in functions1.asp. If the user isn’t logged in then the page redirects to login.asp.

If the session(“e”) (the email address) is empty then it calls cookiechk, which is in a file (templates/cookiechk.asp), which is included in cotop.asp.

Cotop.asp includes the file and calls cookiechk on every page, so that the top bar can display the email address if it is logged in.

Cookiechk looks for Session(“ID”) (the userID) and if missing then it looks for “token” in the “keep” cookie collection. If it finds that then it checks whether the hash of the token matches the stored value and that the current time has not exceeded tokTime. If good, then it puts:

Session(“e”)=mailAddr

Session (“ID”)=userID

Session.Timeout=60 (minutes)

Session(“username”)

Session(“master”) Boolean (David)

Session(“editor”) Boolean – the user has live editor rights in at least one role.

It also updates the “lastlogin” field of livelist with the current time (technically, the latest session).

**Logging out**

The user clicks on the top-left “Log out” link which goes to

<https://webb-site.com/webbmail/login.asp?b=1>

It then deletes all “persist” records with that userID and sets session(“ID”) and session(“e”) to empty string, and all the Boolean Session variables to False. That means that the user is logged out on all devices, not just this one.

**The mailvote.keys table**

This table includes secret values that the Webb-site server needs, such as the login details of your mail server for sending mail, and your Captcha keys for using Google captcha. We have included a CSV file, mailvote.keys.CSV, with the names (but not the values) and description of each value.