Voting Process

Raul Persa, Lukas Vogel

13. Dezember 2015

Outline

- The voter enters the Wahllokal
- The Wahlhelfer hands the voter a valid token for the current Wahlkreis and election. He also enters into the system that the voter has been issued a token
  - If no token is remaining:
    - The Wahlhelfer generates a new batch of tokens using his credentials.
  - If the voter has already been issued a token (i.e. he already voted):
    - The Systems warns the Wahlhelfer. The voter is given a stern talking to and is sent away.
- The voter enters a free Wahlkabine and enters his token and votes.
  - If the sent data is invalid (invalid token, voting for a candidate not running in the current Wahlkreis or election, voting for a party not running in the election, generally trying to manipulate the system by sending his own POST-requests with manipulated data, HUMAN ERROR, VOTER):
    - The user is at fault: The System urges the user to check the entered data and retry
- The System registers the vote and invalidates the token in a single transaction.
  - If the commit fails (token was used by other voter between first consistency check of vote and commit of the voting-transaction, HUMAN ERROR, WAHLHELFER):
    - User is notified that an internal problem has happened and he should notify the Wahlhelfer
    - After checking the system state, the wahlhelfer issues the voter a new token

Implementation

The voting service is reachable at
`http://votingserver/wahl/electionID/wahlkreisID`.
In an actual production environment the server would not be reachable from the internet but only by VPN or something equally secure. The actual voting is done by sending a POST-REQUEST to
`http://votingserver/wahl/electionID/wahlkreisID/vote`.
The fields `token`, `erststimme` and `zweitstimme` have to be set. `erststimme` and `zweitstimme` are the IDs of a candidate / a party.

On the actual voting machine a stripped-down browser only pointing to `/wahl/eid/wkid` could be used as frontend. If this is deemed to insecure and / or resource-intensive an own fronted could be developed sending the right HTTP POST-Requests to the voting server.

Manipulation by sending rogue-POST-requests is impossible. Each issued token is internally associated with an election and a Wahlkreis. Thus a voter can't use his token to vote in another wahlkreis or election. The handling of a single vote is done in one transaction. This guarantees that the database cannot be in an inconsistent state (i.e. voted but token not invalidated).

New tokens can be generated by browsing to the location
`http://votingserver/wahl/tokens/electionID/wahlkreisID/tokenAmount/`.

In a production environment, this site wouldn't be exposed (obviously). It should rather be treated as a proof of concept.

SQL Injection

The server is hardened against SQL-Injections: All user input is escaped properly by using the proper query-building functions of the psqcopg2-python-module.

Protection against multiple voting

An issued token is only valid for exactly one vote (see above). A person can only vote more than once if the Wahlhelfer is issuing a voter more than one token (human error).

Unauthorized voting

Voting is only possible with a valid token. A token is valid for exactly one election and Wahlkreis. This way nobody can vote in a Wahlkreis he is not currently living in. Voting without a token is impossible.

To reduce the risk of MitM-Attacks (stealing tokens etc.), SSL is used. In a production environment VPN and strong encription would be another possibility.

Data protection

At no time in the process a vote is connected with identifying user information.