UNIVERSITÄT SIEGEN

Lehstuhl IT-Sicherheitsmanagment
Prof. Dr. Kesdogan

Assignment 02

in brief:

1. parse the graph, find friends
2. introduce friends to your foaf
3. search the social network graph

1 Preparations

Create a website for your user profile. Optimally it should be created automatically from your public foaf profile. The webpage should allow barrier-free access.

2 Finding Friends through Knows

You are not nosy just because you want to know who is known by the people you know, aren't you? After all, browsing the network graph is one of the defining attributes of social networks as defined by Boyd and Ellison. It is no surprise to you, that your implementation will need the ability to download and parse cards and foaf-files of other persons.

While at it, it should be quite easy to include selected data on those persons within your triple store. Right along your own profile maybe?

This would at least increase the speed of authentication the next time any of your friend is visiting your site.

1. Implement downloading and parsing of foafs (using libraries if you wish).
2. Implement a storage for these downloaded profiles, especially selected parts of them, including public key material.
3. Design requirements for access control for those locally stored profiles.
4. Analyse conceptional security problems of your concept and implementation.

3 Why can't we be friends?

Implement the Open Social "Create a relationship" Interface[1] at your service. For a start ignore the `groupID` parameter. Your implementation should accept modifications to the profile it hosts only if the user is authenticated by the private key related to that profile.

Add a formular submit-button to the homepage of the profile that allows another user to add your Person as "known" into his profile. The button should automatically call the "Create a relationship"

[1] `http://opensocial-resources.googlecode.com/svn/spec/2.0.1/Social-API-Server.xml#People-Service-CreateRelationship`

Kontakt: `fischer@wiwi.uni-siegen.de`

if the visiting user is known, eg. by authentication. Otherwise the identity of the user should be requested before the data is send. Either by presenting a formular input field if the user is unknown or by presenting a second form with that input field. Nonetheless the page should be useable whether or not the visitor is known or anonymous.

Bonus: Use the extension by Eric Vitiello[2] to mark the type of relationship on your own page. Use the label of the relationship within an optional parameter `label` to transfer the label defined in the extension.

4 Bonus: Hold out your hand cos friends of friends will be friends

. . . right till the end (of the graph).

You already are able to parse the foaf of a soon-to-be friend. But what about his (or her) friends, and friend of friends? You want to find the closest friendship path to someone who is knowledgeable in "firebreathing" or is working at the next Chippy (for you are in desperate need of a free lunch). What else is there to do as a throughout breadth-first search (BFS)?

1. Include the friends of a given person into a local foaf-database. You might increase efficiency by only including the interesting bits "id" and "knows".
2. Help your program to read a sensible search request.
3. Check any new data against this search request (this might be done by a library, if, for example, your query language is something like SPARQL).
4. Return the first n finds. (Or continouosly print out any find until the user interrupts.)

[2] `http://www.perceive.net/schemas/20021119/relationship`