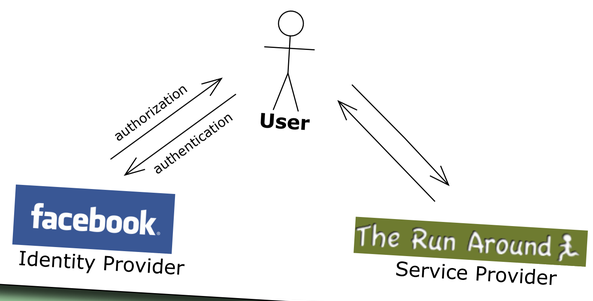
SOFSEM 2011 STUDENT RESEARCH FORUM

FORMAL ANALYSIS OF FACEBOOK CONNECT SINGLE SIGN-ON AUTHENTICATION PROTOCOL

Caterina Urban
Prof. Marino Miculan

Dept. of Mathematics and Computer Science, University of Udine, Italy

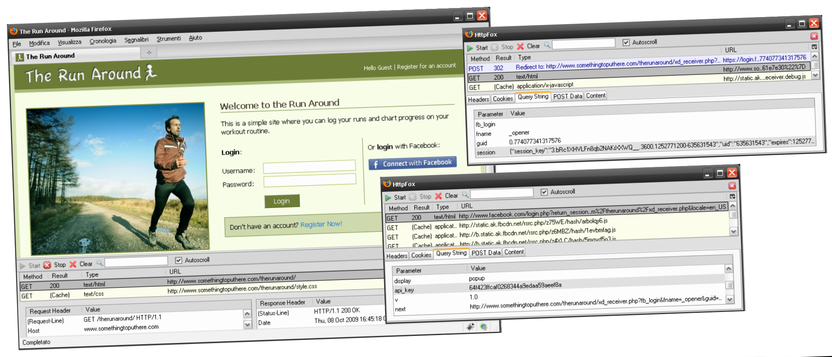
Facebook Connect is a Single Sign-On service integrated in the Facebook Platform (until few month ago)



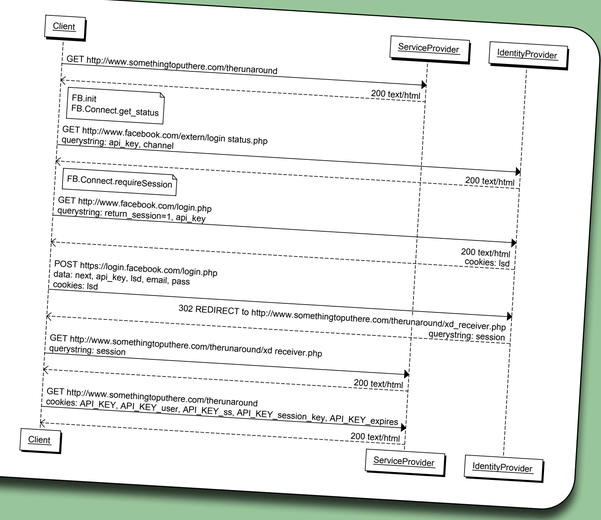
Single Sign-On (SSO) is an access control mechanism whereby a single action of user authentication and authorization can permit a user to access all related but independent systems where he has access permission, without being prompted to login again at each of them.

Is Facebook Connect secure?

Since Facebook Connect is proprietary, a detailed protocol description has not been officially provided. To overcome this problem, we have analyzed all incoming and outgoing HTTP traffic between the browser, the Facebook login server and an application server during the authentication process.



The authentication flow consists of six HTTP requests-responses pairs, one of which uses a secure channel. The data involved in each HTTP request-response transaction are request and response headers, sent and received cookies, query string parameters, POST data and response body. Cookies play a particularly important role, since they carry the information about the login status of the user.



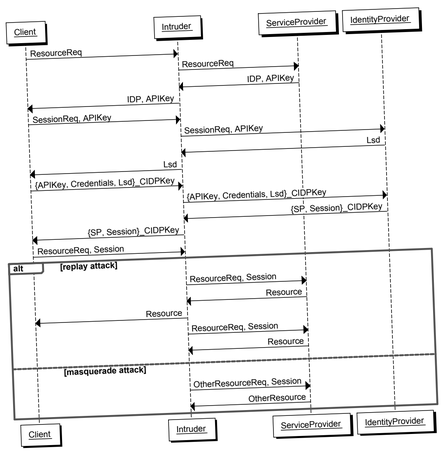
We have translated the protocol in Alice-Bob notation into HLPSL. In particular, we have formally specified the security properties to be verified.

```
role serviceProvider ( ... )
  transition ...
  2. State=1 /\ RCV(resourcereq.Key'.Uid'.Expires'.Ss'.
        Hash(Expires'.Ss'.Key'.Uid'.APISecret)) =|>
     State':=2 /\ SND(resource)
     /\ request(SP,IDP,sp_idp_sig,Hash(Expires'.Ss'.Key'.Uid'.APISecret))

role identityProvider ( ... )
  transition ...
  2. State=1 /\ in(CIDPKey',SSLKey)
     /\ RCV({APIKey.credentials.Lsd}_CIDPKey') =|>
     State':=2 /\ SSLKey':=delete(CIDPKey',SSLKey)
     /\ Key':=new() /\ Expires':=new() /\ Ss':=new()
     /\ Sig':=Hash(Expires'.Ss'.Key'.Uid.APISecret)
     /\ Session':=(Key'.Uid.Expires'.Ss'.Sig')
     /\ SND({SP.Session'}_CIDPKey')
     /\ witness(IDP,SP,sp_idp_sig,Sig')

role serviceProvider ( ... )
  transition ...
  3. State=1 /\ RCV(otherresourcereq.Key'.Uid'.Expires'.Ss'.
        Hash(Expires'.Ss'.Key'.Uid'.APISecret)) =|>
     State':=2 /\ SND(otherresource)
     /\ secret(otherresource,otherresourceid,{SP})
     /\ request(SP,IDP,sp_idp_sig,Hash(Expires'.Ss'.Key'.Uid'.APISecret))
```

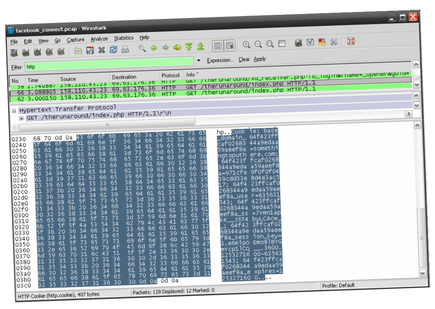
From HTTP traffic analysis we can define a protocol formalization in Alice-Bob notation by abstracting from implementation-level

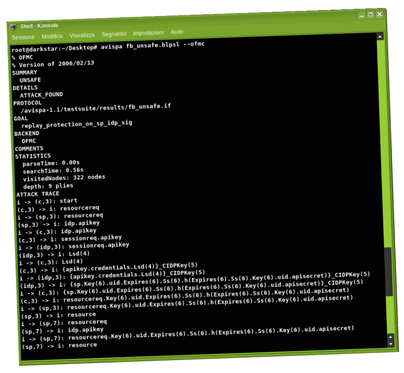
1. $C \to SP : ResourceReq$
2. $SP \to C : IDP, APIKey$
3. $C \to IDP : SessionReq, APIKey$
4. $IDP \to C : Lsd$
5. $C \to IDP : \{APIKey, Credentials, Lsd\}_{CIDPKey}$
6. $IDP \to C : \{SP, Session\}_{CIDPKey}$
7. $C \to SP : ResourceReq, Session$
8. $SP \to C : Resource$

We analyzed the HLPSL formalization using AVISPA (and in particular the OFMC model checker). This analysis has shown that Facebook Connect authentication protocol is subject to a replay attack and a masquerade attack.

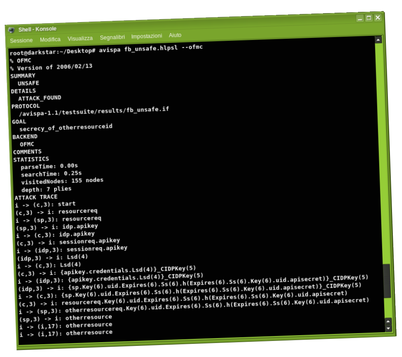


The replay attack can be effectively carried out since the HTTP traffic between the Client and the Service Provider is not encrypted: since HTTP (without SSL) is basically stateless, an intruder can always intercept a packet containing an HTTP request (e.g. using a packet-sniffer like Wireshark), and submit it again.





But also a more serious masquerade attack is possible: an intruder can acquire the HTTP cookies sent together with a legitimate query. Hence, he can send again them to the Service Provider to be authenticated as the Client to obtain illegitimately other resources (e.g., a user's mailbox, even if it has not been accessed before by the owner).



The masquerade attack can be prevented without establishing an encrypted channel. To this end, we propose a small correction to the protocol which adds the authentication of resource requests by means of a Diffie-Hellman session key.

1. $C \to SP : ResourceReq$
2. $SP \to C : IDP, APIKey, \boldsymbol{\{A, Ysp\}_{IDPKey}}$
3. $C \to IDP : SessionReq, APIKey$
4. $IDP \to C : Lsd$
5. $C \to IDP : \{APIKey, Credentials, Lsd, \boldsymbol{\{A, Ysp\}_{IDPKey}}\}_{CIDPKey}$
6. $IDP \to C : \{SP, Session, \boldsymbol{A, Ysp}\}_{CIDPKey}$
7. $C \to SP : ResourceReq, Session, \boldsymbol{Yc, Hash(ResourceReq.CSPKey)}$
8. $SP \to C : Resource$

Again, this protocol has been formalized in HLPSL and formal verification has been carried out using the OFMC model checker. The model checker has found no attacks.

http://users.dimi.uniud.it/~marino.miculan/data/downloads/SOFSEM11.pdf