

## **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



## 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 Run Around - Hozilla Firefox         Ele Modifica Visualizza Cronologia Segnalibri Strumenti Aluto         The Run Around	Hello Guest   Re	egister for an account
The Run Around A	Welcome to the Run Around         This is a simple site where you can log your runs and chart pr         workout routine.         Login:         Username:         Password:         Login         Don't have an account? Register Now!	POST 302 Headree 00 met//mm.son.org.gr/ http://www.so51e/e3022224/0  GET 200 text/html GET (Cache) application/k-javascript Headers Cookies Query String POST Data Content Parameter Value
Start Stop       Clear       URL         Method       Result       Type       URL         GET       200       text/html       http://www.sometil         GET       (Cache)       text/css       http://www.sometil         Request Header       Value       (Request-Line)       GET / therunaround/ HTTP/1.1         Host       www.somethingtoputhere.com       Completato	ingtoputhere.com/therunaround/ ingtoputhere.com/therunaround/style.css Response Header Value (Status-Line) HTTP/1.1 200 OK Date Thu, 08 Oct 2009 16:45:18 0	arameter Value popup play popup key 64/423ffcaf0268344a9edaa59aeef8a 1.0 xt http://www.somethingtoputhere.com/therunaround/xd_receiver.php?fb_login&fname=_opener&guid=

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.

<u>Client</u>	
	ServiceProvider IdentityProvider
GET http://www.somethingtoputhere.com/therunaround	
FB.init FB.Connect.get_status	<u>xt/html</u>
GET http://www.facebook.com/extern/login status.php querystring: api_key, channel	
FB.Connect.requireSession	200 text/html
GET http://www.facebook.com/login.php querystring: return_session=1, api_key	
←POST https://login.facebook.com/login.php data: next, api_key, lsd, email, pass cookies: lsd	200 text/html cookies: lsd
302 REDIRECT to http://www.somethingtoputhere.com/th	Perunaround/xd receiver a h
GET http://www.somethingtoputhere.com/therunaround/xd receiver.php querystring: session	guerystring: session
GET http://www.somethingtoputhere.com/therunaround cookies: API_KEY, API_KEY_user, API_KEY_ss, API_KEY_session_key, API_KEY_expires	
int 200 text/htm Servic	eProvider <u>IdentityProvider</u>

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')

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

1.  $C \rightarrow SP : ResourceReq$ 2.  $SP \rightarrow C : IDP, APIKey$ 3.  $C \rightarrow IDP : SessionReq, APIKey$ 4.  $IDP \rightarrow C : Lsd$ 5.  $C \rightarrow IDP : \{APIKey, Credentials, Lsd\}_{CIDPKey}$ 

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))

6.  $IDP \rightarrow C : \{SP, Session\}_{CIDPKey}$ 7.  $C \rightarrow SP$  : ResourceReq, Session

8.  $SP \rightarrow C : Resource$ 



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 masquerade attack can be prevented without establishing an encrypted channel. To this end, we propose a small legitimate query. Hence, he can send again them to the correction to the protocol which adds the authentication of Service Provider to be authenticated as the Client to obtain resource requests by means of a Diffie-Hellman session key. illegitimately other resources (e.g., a user's mailbox, even if it has not been accessed before by the owner). 1.  $C \rightarrow SP : ResourceReq$ 2.  $SP \rightarrow C : IDP, APIKey, \{A, Ysp\}_{IDPKey}$ 3.  $C \rightarrow IDP : SessionReq, APIKey$ 4.  $IDP \rightarrow C : Lsd$ 5.  $C \rightarrow IDP : \{APIKey, Credentials, Lsd, \{A, Ysp\}_{IDPKey}\}_{CIDPKey}$ 6.  $IDP \rightarrow C : \{SP, Session, A, Ysp\}_{CIDPKey}$ Again, this protocol has been formalized in HLPSL and 7.  $C \rightarrow SP$  : ResourceReq, Session, Yc, Hash(ResourceReq.CSPKey) formal verification has been carried out using the OFMC model checker. The model checker has found no attacks. 8.  $SP \rightarrow C : Resource$ http://users.dimi.uniud.it/~marino.miculan/data/downloads/SOFSEM11.pdf

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.





🖉 Shell - Konsole	
Sessione Modifica Visualizza Segnalibri Impostazioni Aiuto	
belance (beskton# avispa fb unsafe.hlpslofmc	
root@darkstar:~/uesktop# ut_p# tt	
% UPRC 0 of 2005/02/13	
INSAFE	
DETAILS	
ATTACK_FOUND	
PROTOCOL the second to state if	
/avispa-1.1/testsuite/results/ib_unsults_	
GOAL	
replay_protection_vn_sp_tars	
BACKENU	
COMPENTS	
STATISTICS	
parseTime: 0.00s	
searchTime: 0.56s	
visitedNodes: 322 nodes	
depth: 9 plies	
ATTACK TRACE	
$i \rightarrow (c,3)$ : start	
(c, 3) -> 1: lestateree	
$(on \beta) \rightarrow i$ ; idp.apikey	
$i \rightarrow (c,3)$ ; idp. apikey	
(c,3) -> i: sessionreq.apikey	
i -> (idp,3): sessionreq.apikey	
$(idp, 3) \rightarrow i: Lsd(4)$	
i -> (c,3): Lsd(4)	
$(c,3) \rightarrow 1; \{aplKey, credentials.Lsd(4)\}$ _CIDPKey(5)	(5)
1 -> (10p,5); (apike)(0.111, Expires(6).Ss(6).h(Expires(6).Ss(6).key(0).ulu.apisecret)) CIDPKey(5)	5)
(190,5) (190,10) (190	
(c,3) -> i: resourcereq.Key(6).uid.Expires(6).Ss(6).n(Expires(6).Ss(6).Key(6).uid.apisecret)	
i -> (sp,3): resourcereq.Key(6).uid.Expires(6).Ss(0).ntexpires(6).text	
(sp,3) -> i: resource	
i -> (sp,7): resourcereq	
<pre>(sp,7) -&gt; i: idp.aplKey (6).uid.Expires(6).Ss(6).h(Expires(6).Ss(6).Key(6).uid.aplsecret)</pre>	
i -> (sp,7): resourcereq.key(s) talatere	
(sp,7) -> 1: resource	

