

oauth: flaws and proposal

2012





inb4

bit.ly/homakov_rails_security

- any questions?

bit.ly/homakov_oauth_demo

- these slides



сейчас

фреймворк - концепция для предоставления Клиенту доступ к ресурсам Юзера у Провайдера. отличается от oauth1 никаких сигнатур, https based не идеален, в разработке. тем не менее уже используется в прод.



workflow

Client

Provider

Resources

User

Device(User-Agent, Desktop app..)



used things

```
access token
refresh token
code(to obtain access token)
redirect uri
client id/client secret(Client credentials)
scope(привелегии, доступы)
response type(which way to auth)
```



response_type=(code|token)

token - Implicit/code - Authorization code

Implicit is insecure. direct obtaining of access_token

"Authorization code" flaw is more secure BUT



desktop apps

login/password exposure to get access_token and refresh_token.



authcode - just upgrade

если на ресурсе есть XSS то получение access_token не составляет труда - response_type=token, подставляется нужный redirect_uri (можно с пассивным xss) и вытаскивается из URI fragment (недоступного в реферерах) токен.



authcode leaks via referer..?

To obtain access_token the Client MUST provide 'redirect_uri' used to get 'code'. Thus даже используя redirect_uri позволяющий узнать code вы не сможете его использовать т.к. код был выпущен для левого redirect_uri



wait for it...



CSRF

C..WTF? which benefit?

- 1. на первом этапе <iframe src=site. com/fb_connect></iframe> - no
- 2. на этапе пермита <iframe src=facebook. com/hacker_params></iframe> нет, можно только менять redirect_uri a это пресекается или response_type



???? PROFIT



hacker doesn't use callback -.
store it in img src or iframe and CSRF the
User

* really easy to use. scriptkiddies, c'mon!

Your FB is gonna be attached to User's account.



```
a very few showcases:
pinterest
digg
soundcloud
snip.it
bit.ly
stumbleupon
```



also all Ruby on rails + omniauth websites

inurl:"/auth/facebook"

https://www.google.ru/search?sourceid=chrome&ie=UTF-8&q=inurl%3A%22%2Fauth%2Ffacebook%22&qscrl=1#q=inurl:% 22/auth/facebook%22&hl=ru&newwindow=1&qscrl=1&prmd=imvns&ei=GCzbT5PhOMKj4gT2ttnTCg&start=80&sa=N&bav=on.2, or.r_gc.r_pw.r_cp.r_qf.,cf.osb&fp=b08787c6ce15cd4d&biw=1536&bih=748

Результатов: примерно 27 900

PoC?



'state' MUST be used to prevent CSRF(not just to carry "popup_page").

OPTIONAL = not used

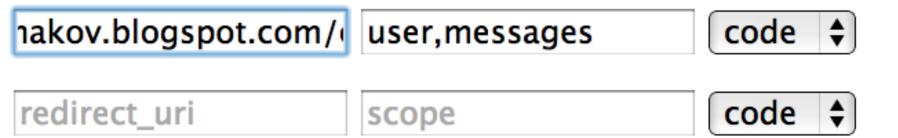


flaws

- 1. response_type is an option
- 2. redirect_uri must be hardcoded. only 1 domain is allowed. we need site.com/cb site. net/cb site.local/cb
- 3. state is optional. tons of vulns
- 4. scope is an option. user can remove some scopes
- 5. User cannot adjust expire time



multiple redirect_uri-s



Add callback



proposal 1.0

```
code is generated by Client
"code=#{session[:code] = random string()}"
param.
code IS NOT returned back, callback don't
get any params
to obtain access token redirect uri=...
&code=session[:code]&client credentials...
BUT exploitable, session fixation
```



proposal 2.0

tie 'state' as well as 'code' and send nothing to callback. then obtain access_token with code+state+client_creds redirect_uri is not needed anymore CSRF is fixed by design, redirect_uris, scopes and response_type are very agile and secure.

thoughts?