OAuth
securing the insecure

khash kiani
khash@thinksec.com
roadmap

‣ OAuth flow

‣ malicious sample applications
  ★ mobile OAuth google app
  ★ web-based OAuth facebook app

‣ insecure implementation
  ★ flawed session management
  ★ password management
  ★ insecure storage of secrets

‣ summary
what’s OAuth?
user-centric scheme

user controls authorization

user

mT
mint

FM Token
AIG Token

AIG

Freddie Mac

Twitter

Twitpic

share photos on twitter

Twitter Token

Twitter Token

Twitter Token

Twitter Token

4
actors:
resource owner (user)
resource consumer (client)
resource provider (server)

tokens:
consumer credentials
request token
access token
authorization flow

1. client app authentication
2. get request token: POST oauth/request_token
3. authenticate user: GET oauth/authorize
4. get access token: POST oauth/access_token
building malicious OAuth clients
(native and web apps)
password theft with Google client
(a native iOS mobile app)
OAuthSampleTouch mobile Google app

- download
- compile
- run

- edit controller
modify the UIWebViewDelegate's:

webView:shouldStartLoadWithRequest:navigationType

callback method to intercept the login page prior to sending the post request
OAuth process with an embedded view

user authenticates and grants permission
output the Google credentials
“but it looked so official!”

OAuth provides the user with a false sense of safety in the authentication workflow
recommendations  
(mobile apps)

- **client application developers**: keep authentication outside the app and inside the browser

- **users**: do not trust clients that do not use a trusted neutral application such as safari to manage server auth

- **protocol designers**: stricter policies around authenticating clients to server. better browser API support
fortune telling facebook app
(a browser-based web application)

a social engineering oauth application to establish user trust
lure the victim to use your app

domain apps.facebook.com is trustworthy!

---

Welcome to the Social-Engineer Toolkit (SET). Your one stop shop for all of your social-engineering needs.

Select from the menu on what you would like to do:

1. Spear-Phishing Attack Vectors
2. Website Attack Vectors
3. Infectious USB/CD/DVD Generator
4. Update the Metasploit Framework
5. Update the Social-Engineer Toolkit
6. Create a Payload and Listener
7. Mass Mailing Attack
8. Help, Credits, and About
9. Exit the Social-Engineer Toolkit

Enter your choice: 

---

Hi Victim -

Your life is currently full of mishaps, and nothing is going the way you wanted it to. Ever wonder if your fortune will change? Look no further, the Red Devil will have your answer!

Click here or visit https://apps.facebook.com/RedDevilFortune/.

Good luck and see you on the dark side!
https://apps.facebook.com/redevilfortune/
70% * source: core impact client-side phishing campaign
query private user messages

```sql
$fql = "SELECT body, thread_id FROM message WHERE thread_id=222222;"
```

(read the inbox messages)

I like to bank at Hack-Muh-Bank
build the trap to aid exploitation

**link to execute ajax post and carry our CSRF**

```
<script>
  function jsinit_load()
  {
    var http = new XMLHttpRequest();
    var url = "/auth/post.php";
    var params = "account=9999999999&amount=300000";
    http.open("POST", url, true);

    //Send the proper header information along with the request
    http.setRequestHeader("Content-type", "application/x-www-form-urlencoded");
    http.setRequestHeader("Content-length", params.length);
    http.setRequestHeader("Connection", "close");

    http.onreadystatechange = function()
    {
      if(http.readyState == 4 & http.status == 200)
      {
        alert(http.responseText);
      }
    }
    http.send(params);
  }
</script>
```
“but it looked so official!”

OAuth provides the user with a false sense of safety in the authentication workflow
Dear Facebook,
what is the business need for a web application to read my private messages?
Insecure Implementation
flawed session management
Avon selects twitterfeed to publish something
- Avon is redirected to twitter’s authorization endpoint
- Avon enters his twitter credentials and grants access
- Avon is redirected back to complete the feed
- Avon signs out of twitterfeed and walks away
what about his twitter session?
risks

- unattended session
- no session timeout
- user remains logged in
what can go wrong?
We wish @joe biden the best of luck as our new President of the United States. In such a time of madness, there’s light at the end of tunnel

2 hours ago

BREAKING NEWS: President @BarackObama assassinated, 2 gunshot wounds have proved too much. It’s a sad 4th for #america. #obamadead RIP

2 hours ago

#ObamaDead, it’s a sad 4th of July. RT to support the late president’s family, and RIP. The shooter will be found

2 hours ago

@BarackObama shot twice at a Ross’ restaurant in Iowa while campaigning. RIP Obama, best regards to the Obama family.

2 hours ago
I give myself to Lucifer every day for it to arrive as quickly as possible. Glory to Satan!

britneyspears
Britney Spears

I hope that the new world order will arrive as soon as possible! -Britney

britneyspears
Britney Spears
problem, meet solution

- invalidate server session
- short-lived access token
- no auto-processing
a better approach

You're logged out of Flickr. Sign in again?

Log out of the Yahoo! network too?
can you really change your password?
Avon_Barksdale_'s settings

Account  Password  Mobile  Notifications  Profile  Design  Applications

Current Password: *********
   Forgot your password?

New Password: ********

Verify New Password: ********

Change
change password = old password still works!
You've allowed the following applications to access your account

**twitterfeed** by twitterfeed
- feed your blog to twitter - twitterfeed lets you post any RSS or Atom feed to twitter automatically
- read and write access • Approved: Sun Feb 27 23:04:15 2011

**Twitpic** by Twitpic Inc
- Share photos on Twitter with Twitpic
- read and write access • Approved: Fri Feb 25 12:39:04 2011
solution

- ensure compromised credentials cannot be used
- revoke tokens upon password changes
  - results from Facebook access token leakage to 3rd party apps
insecure storage of secrets
(consumer credentials)
1. public class TwitterClient {
2.
3.   private static String key = "qSkJuxxxxxxxxxx76A";
4.   private static String secret = "Bs738xxxxxxxxxxxxxZe9EhXw";
server-side

- isolate the credentials
- protect the integrity
native clients

- native mobile app
- desktop apps
“... if twitter uses the client secret in installed applications for anything other than gathering statistics, well, they should reconsider.”

“So forget about using the consumer credentials for anything other than somewhat reliable statistics.”

- e. hammer lahav
how about these use cases:

› fulfill specific business requirements  
  - server must keep track of all clients

› prevent phishing attacks
popular implementations

(native apps)

1. omit the client credentials entirely
2. embed in the client app itself
threat

(with embedded client credentials)

- compromised credentials
open source clients

- source code
- resource bundle
the not so secret consumer secrets

```python
import appuifw
appuifw.app.directional_pad = False
appuifw.app.body = appuifw.Text(u'Please update your feed')
appuifw.app.title = u'ff60'
appuifw.app.screen = 'normal'

import sys
import e32
import e32dbm

import friendfeed
import re

SIS_VERSION = "0.2"

oauth_consumer_key = u'039f2ee0fe942be9ca9ccdd3455a98c'
oauth_consumer_secret = u'6cde18c375644d4a5619aa5b42c81d85cb4116dd4a84a4948f2740659ff096ea0'
ff_num_per_page = 25

class Main:
    def __init__(self):
        # отключаем экранную клавиатуру
        self.db = e32dbm.open(u'c:\ff60.db', 'c')
        self.data = None
        self.lb = None
        self.links_list = appuifw.Listbox([u'Links list'], self.open_link)
        self.page = 0
        self.ff = None
```
DEBUG = False
TEMPLATE_DEBUG = DEBUG
FRONTEND_URL =

OAUTH_CONSUMER_KEY = '3471c80c5d0146a2#f8b560d14c21ca8d' '#02fb15e494e89c3c'
OAUTH_CONSUMER_SECRET = 'fzBNIZDG#Wr07GUR' '#1N8D21k'
OAUTH_GENERAL_PURPOSE_KEY = 'Gjs2HVZjPF6JH8A8' '#9BdSpFvsaOzJz3tz' '#VPiGwNzejA5Z16HE'
OAUTH_GENERAL_PURPOSE_SECRET = 'nq8LCCZTGWKaeSio' '#jZHLZe0BuTO41kG' '#DzEuo8GFESsp0F2'
DATABASE_ENGINE = 'mysql'
DATABASE_NAME = 'db85894_motion'
DATABASE_USER = 'db85894'
DATABASE_PASSWORD = 'w4yn#ePW'
DATABASE_HOST = 'internal-db.s85894.gridserver.com'
DATABASE_PORT =

{$REGION 'SysConst'}
C_RN = #13#10;
C_MN = '#0D#0A';
C_BR = '<br>,'
C_HR = '<HR>,'
C_AS = '<b>%s</b>'
C_KB = 'KB';
C_MB = 'MB';
C_VS = '%s';
C_VD = '%d';
C_Dtseconds = 1 / SecsPerDay;
C_DblClickTime = 0.6 * C_Dtseconds;
C_WM_APPBAR = WM_USER + 1;
X_Twitter_OAuth_Consumer_Key = 'L2k1KZBCDXAAS79jEBdOJg';
X_Twitter_OAuth_Consumer_Secret = 'uKWhm36A2ZpaGnmSNKQh0hT2rD656xRwTPYJ6Kg';
{$SENDREGION}

{$REGION 'FilesConst'}
closed source clients

- binary extraction on android oauth client:
  - astro file mgr to copy the client app
  - poke around
  - classes.dex
  - “dexdump classes.dex”
compromised credentials

impact:

- key rotation and kill switch
- not meeting business requirements
- anonymous publication by competition
- susceptible to phishing attacks
alternative mitigation

- a deviated approach with automated provisioning
alternate flow
(mobile)

- authenticate user to client’s web server
- call home to get device id
- store device id locally
- proceed with oauth flow to get request token
- validate device id to authenticate client
- proceed with the flow to grant access token
conclusion

- defeating password anti-pattern
- trusting native mobile apps
  - don’t trust the logo
  - don’t trust the domain
- session & pswd management
- client authentication
  - consumer credentials
- implementation, not protocol
take-away:
use it when it makes sense!

I DON'T ALWAYS USE OAUTH

BUT WHEN I DO, I DON'T SCREW AROUND
please turn in your completed feedback form at the registration desk

THANK YOU!

khash@thinksec.com