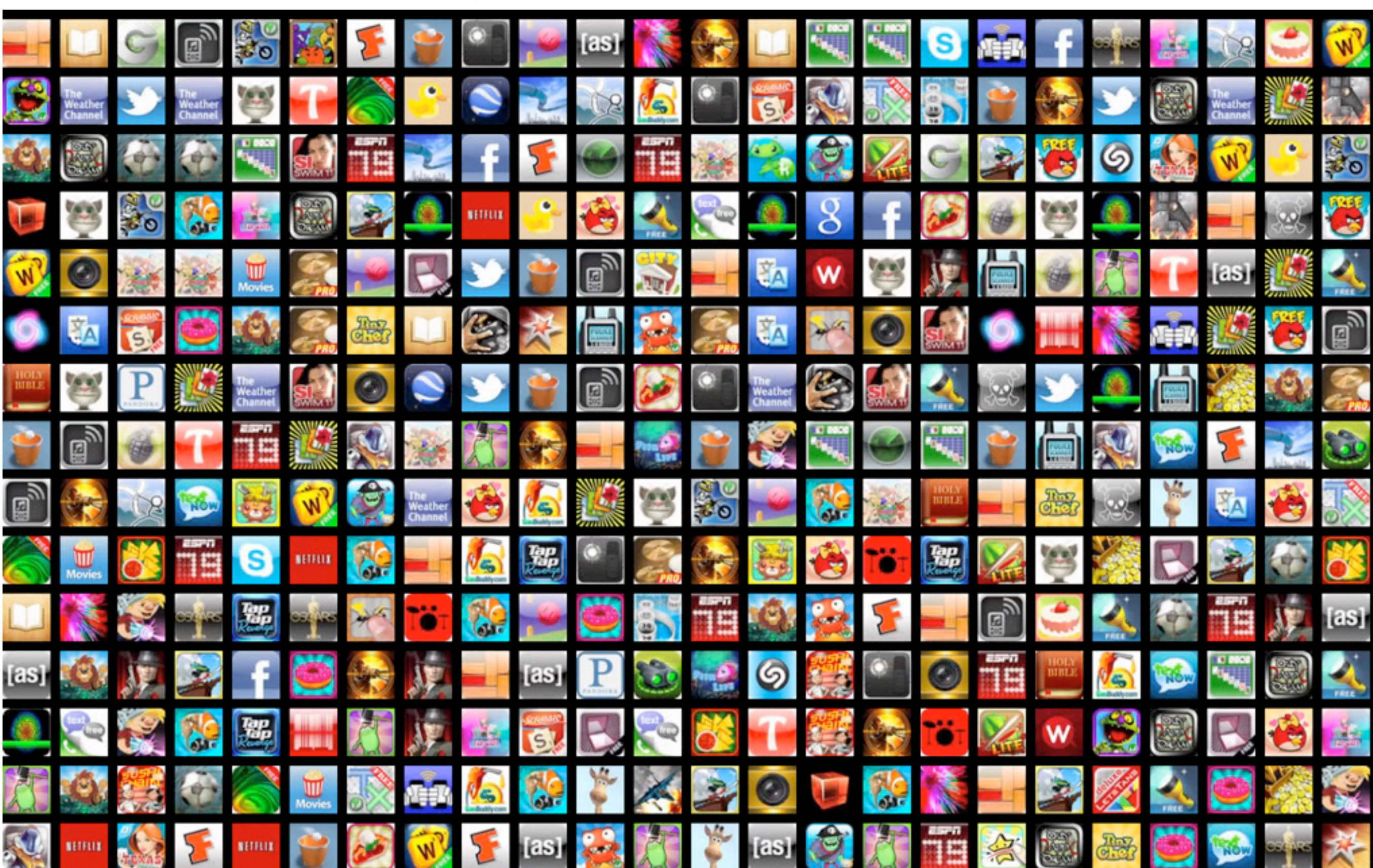


Hacking and Securing Next Generation iPhone and iPad Apps



10 BILLION APPS DOWNLOADED

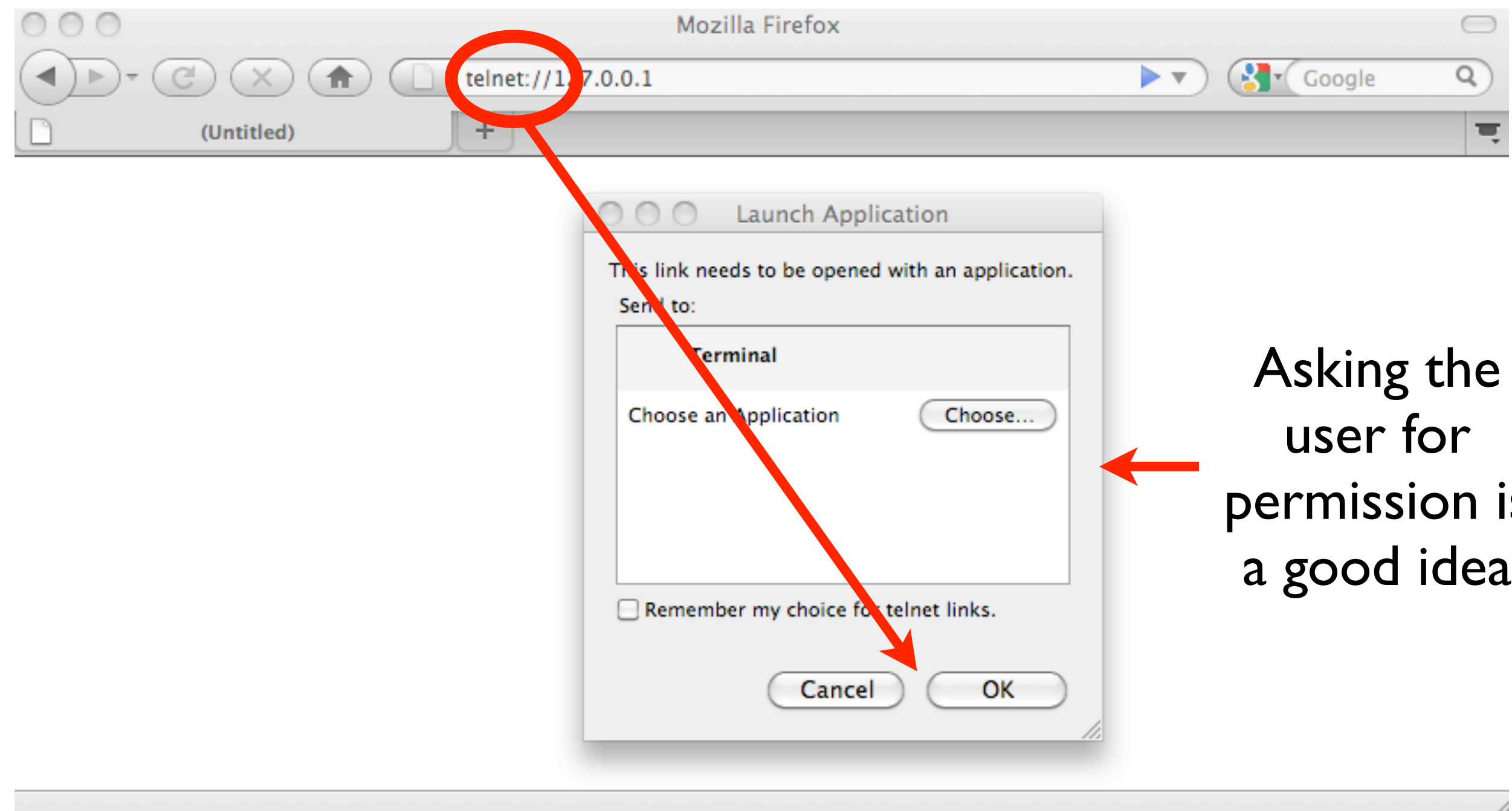


100 MILLION iPhones + 15 MILLION iPads SOLD

Focus

- ▶ Focus on the App layer
- ▶ Net-new attacks targeting iOS Apps
- ▶ URLScheme handling attacks
- ▶ UIWebView and UI Spoofing
- ▶ Apple Push Notifications: Use and Abuse
- ▶ A word on file system encryption and data protection
- ▶ Clear-text network pranks, privacy leakage, and DeCloaking attacks
- ▶ The implications of location-aware Apps
- ▶ Take home bonus check-list :-)

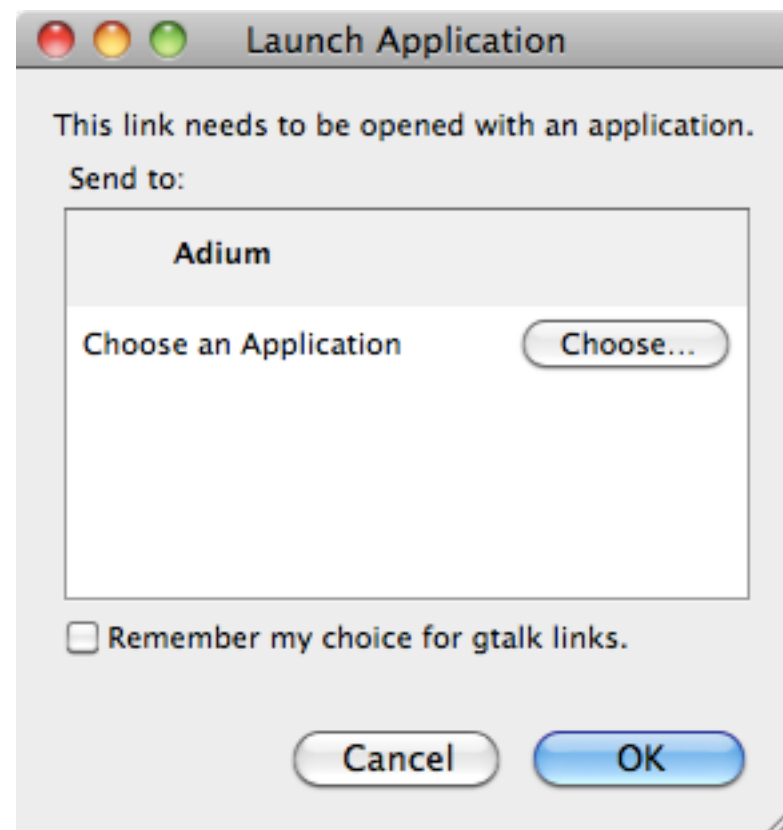
Protocol Handlers: Quick Recap



Protocol Handlers: Quick Recap

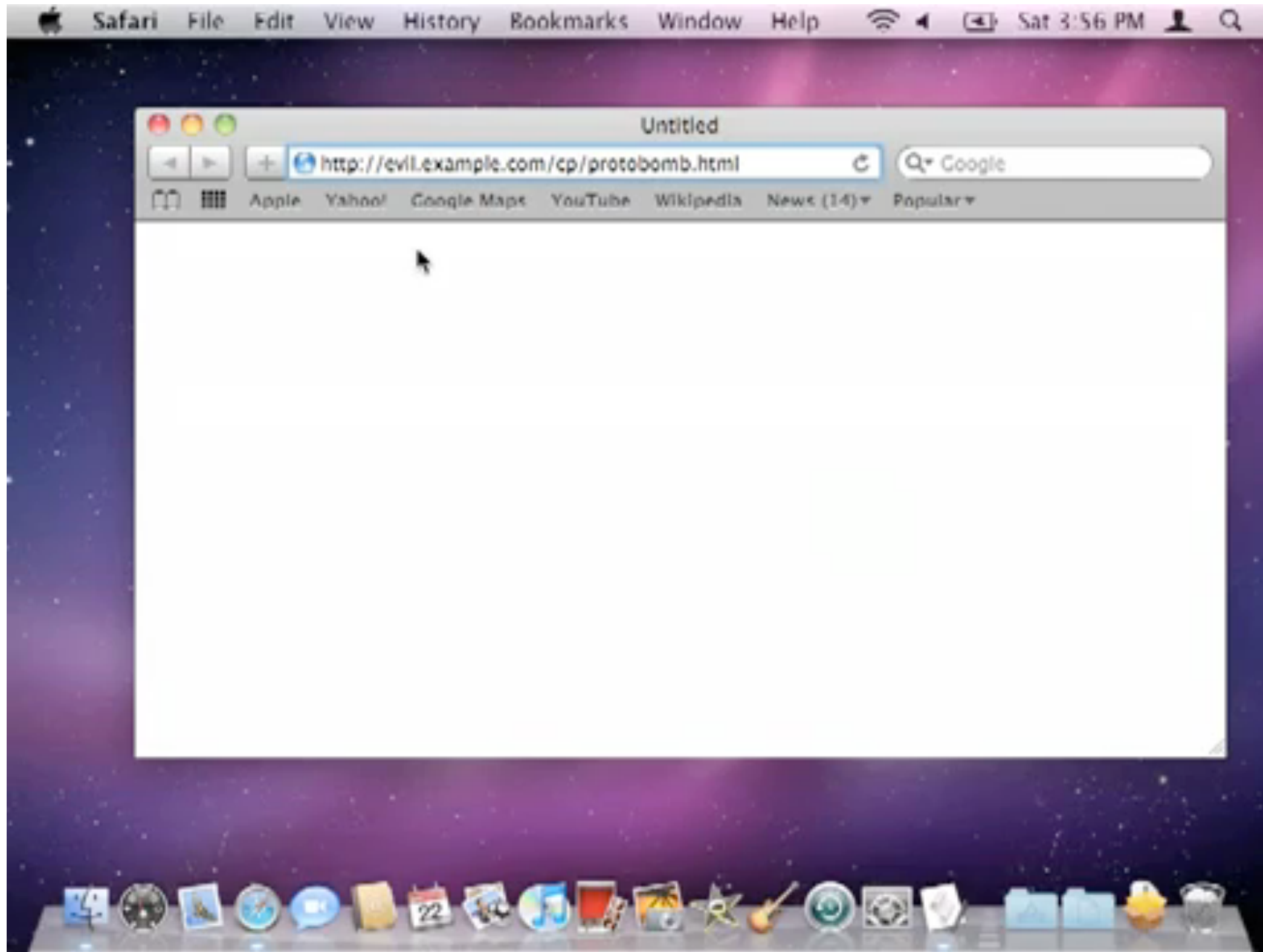


```
<iframe id="lulz" src="gtalk://justin_bieber"></iframe>
```



Asking for
permission before
launching an
executable or
connecting with
Justin Bieber is *also*
a good idea

Safari on OSX #FAIL



Does not ask user for permission before launching executables

Lots of low hanging fruit here :-)

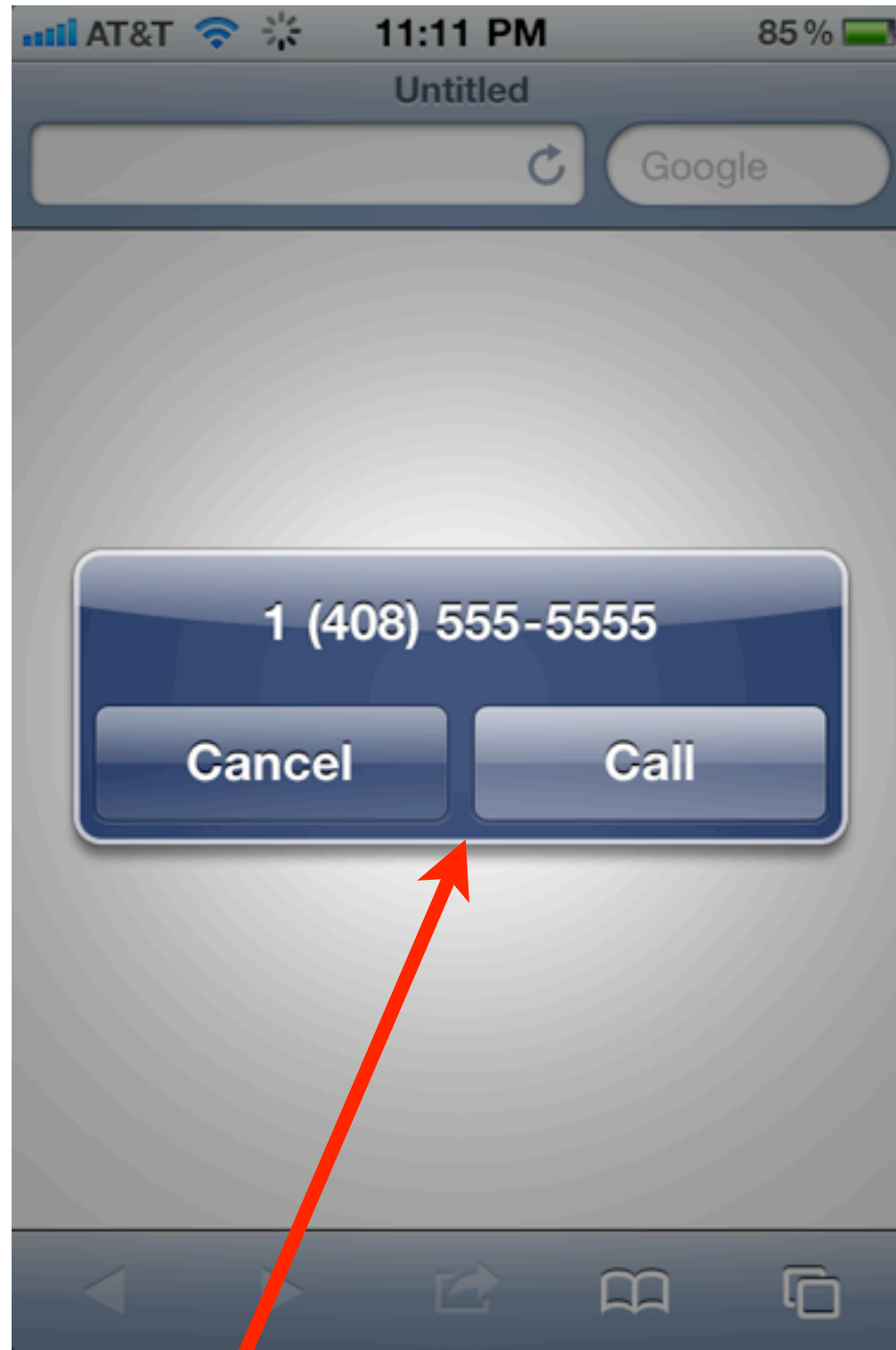
Malicious websites can DoS desktops (video)

<http://dhanjani.com/blog/2010/05/2-years-later-droppin-malware-on-your-osx-carpet-bomb-style-and-then-some.html>

OSX Lion

A large, thick red handwritten 'NO' is superimposed over the text 'OSX Lion'. The 'N' is formed by two strokes, and the 'O' is a single continuous loop. A small checkmark is drawn inside the 'O'.

tel: on Safari (iOS) Gets Special Treatment



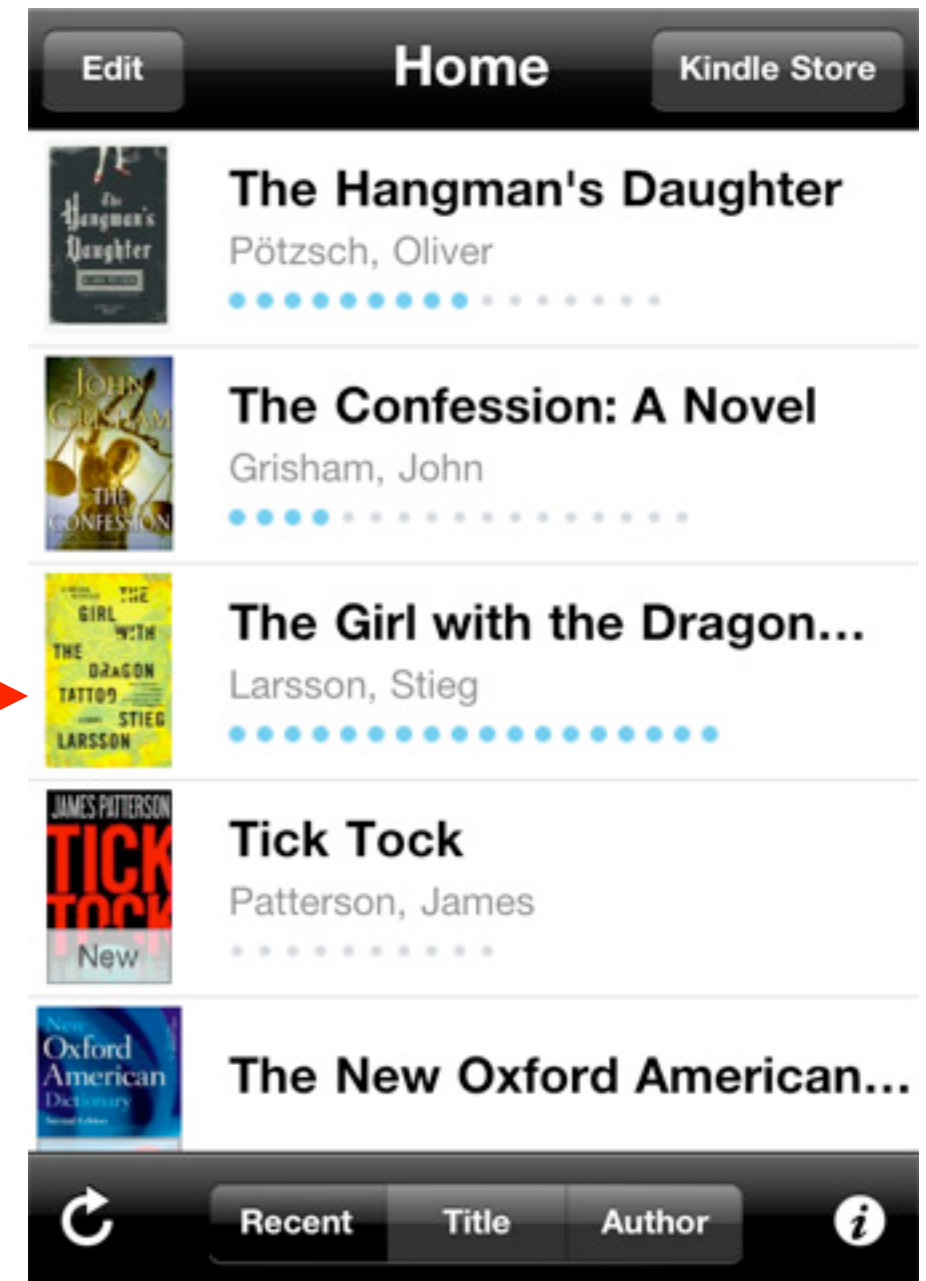
Asking the user
for permission
prior to
launching a
phone-call is a
good idea

```
<iframe src="tel:1-408-555-5555"></iframe>
```


Other URLSchemes on iOS *Yank* Straight into the App!

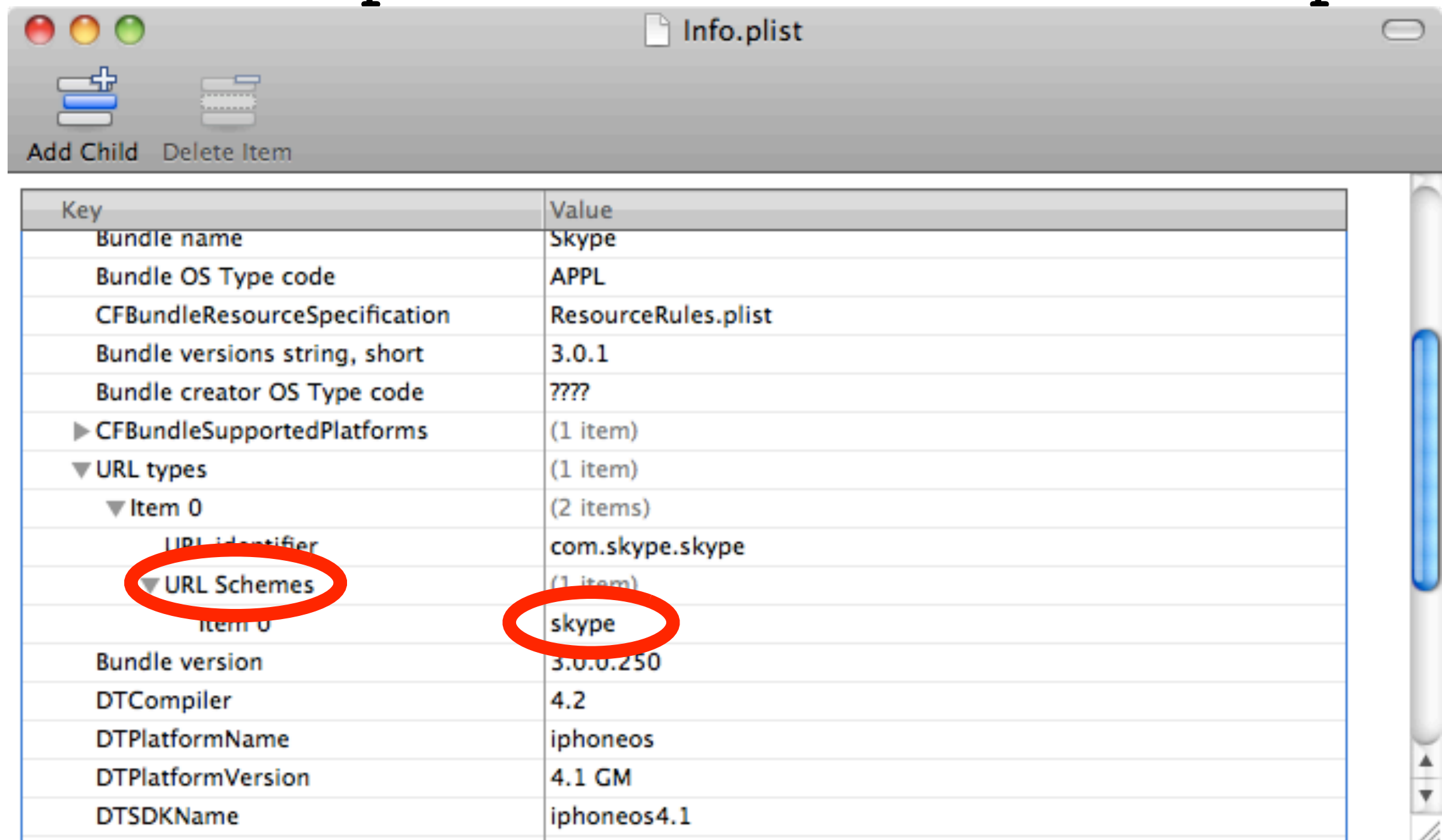


kindle://



Locating Exposed URLSchemes

- ▶ Goto `~/Music/iTunes/Mobile Applications`. Copy the `.ipa` file, rename to `.zip` and unzip.
- ▶ Locate `Info.plist` file. Open with “Property List Editor” or convert to XML: `plutil -convert xml1 Info.plist`



Malicious Websites Can Force Arbitrary skype: Calls



`<iframe src="skype:11408555555?call">`

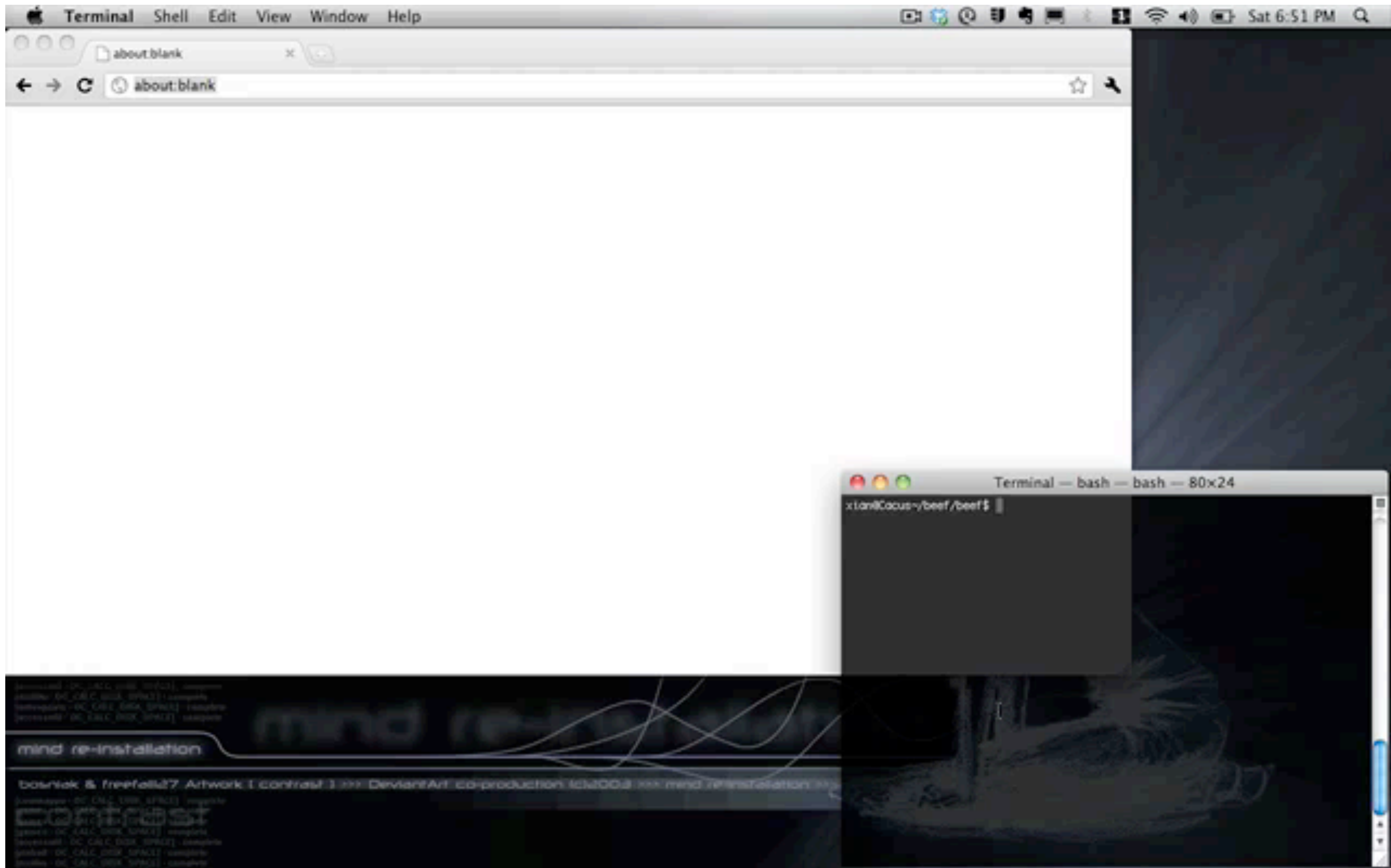
`<iframe src="skype://justin_bieber?call">`



IMPLICATIONS:

- ▶ Malicious websites can invoke arbitrary Skype calls
- ▶ Identity de-cloak
- ▶ Can happen much too quickly
- ▶ A forced call to Justin Bieber can be especially devastating

skype: Now Incorporated Into BeEF [video]



<http://www.youtube.com/watch?v=5SVu6VdLVgs>

Chronology

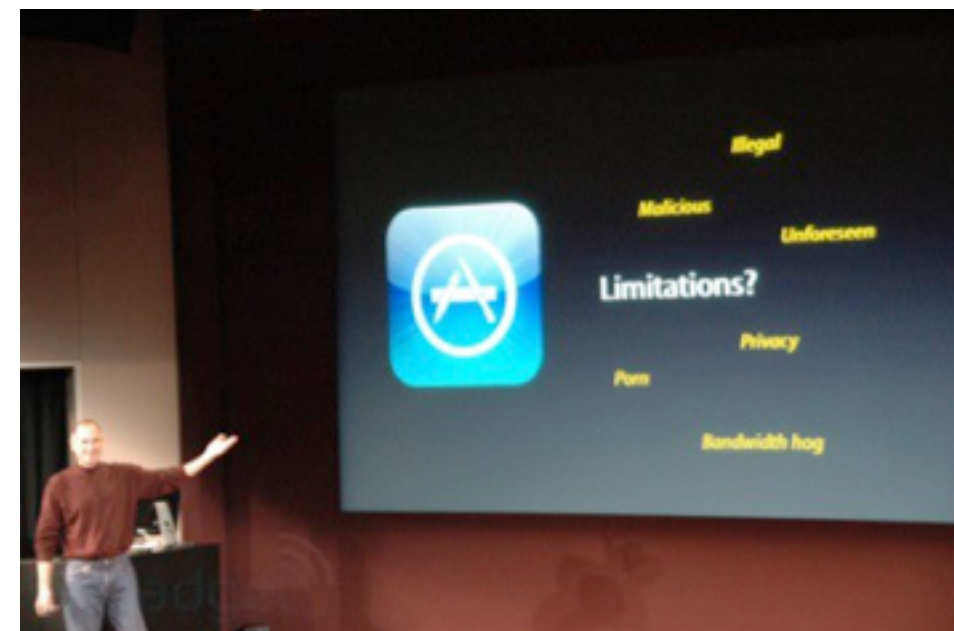
- ▶ October 9, 2010: Reported issue to Apple
- ▶ October 12, 2010: Response from Apple: Onus on receiving App
- ▶ October 12, 2010: Proposed another side effect to Apple
- ▶ October 12, 2010: Apple to research proposed side effect
- ▶ October 13, 2010: Contacted Skype Security team to report issue.

Never heard back

- ▶ December 20, 2010: Skype pushed out fix in Skype 3.0 for iOS
- ▶ February 8, 2011: Apple indicates next update of iOS will have some additional controls (no firm details available)

Open Questions

- ▶ Should Safari interface with Apps and ask for authorization before yanking the user into the App?
- ▶ Should Apple be expected to kick out existing Apps when vulnerabilities are reported?
- ▶ What exactly is Apple's methodology to vet if an App is secure before?
- ▶ Should a list of exposed URLSchemes be available to the advanced user / enterprise administrator?



Implications Beyond skype:

► Custom-developed Apps in the field:

utility_dashboard: //shutdown/
device_id=34

health_record: //close_case/
patient_id=993423

► URLSchemes often used to communicate between a suite of Apps

► Possible venue of back-door functionality

► *Undocumented* URLSchemes in many Apps

► There is a lot of low hanging fruit in this area

URLScheme	App
mailto:	Mail
itms-books[s]:	iBooks
comgoogleearth	Google Earth
itranslate:	iTranslate
maps:	Google Maps
tweetie:	Tweetie
twinkle:	Twinkle
twitterific:	Twitterific
itms-apps:	App Store
sms:	SMS
boxcar:	Boxcar
fb:	Facebook
portscan:	PortScan
twitter:	Twitter
yelp:	Yelp

Also See:

[http://wiki.akosma.com/
iPhone_URL_Schemes](http://wiki.akosma.com/iPhone_URL_Schemes)

+

<http://handleopenurl.com/scheme>

Undocumented URLSchemes in Facebook.App

- ▶ Apps in the App Store are encrypted
- ▶ See <http://dvlabs.tippingpoint.com/blog/2009/03/06/reverse-engineering-iphone-appstore-binaries> for details
- ▶ Or use “Crackulous”

```
$ strings Facebook.app/Facebook | grep 'fb:' | more
```

...

```
fb://online#offline
```

```
fb://birthdays/ (initWithMonth:) / (year:)
```

```
fb://place/ (initWithPageId:)
```

```
fb://places/ (initWithCheckinAtPlace:) / (byUser:)
```

```
fb://places/legalese/tagged/ (initWithTaggedAtPlace:
```

```
fb://publish/profile/ (initWithUID:)
```

```
fb://publish/post/ (initWithPostId:)
```

```
fb://publish/photo/ (initWithUID:) / (aid:) / (pid:)
```

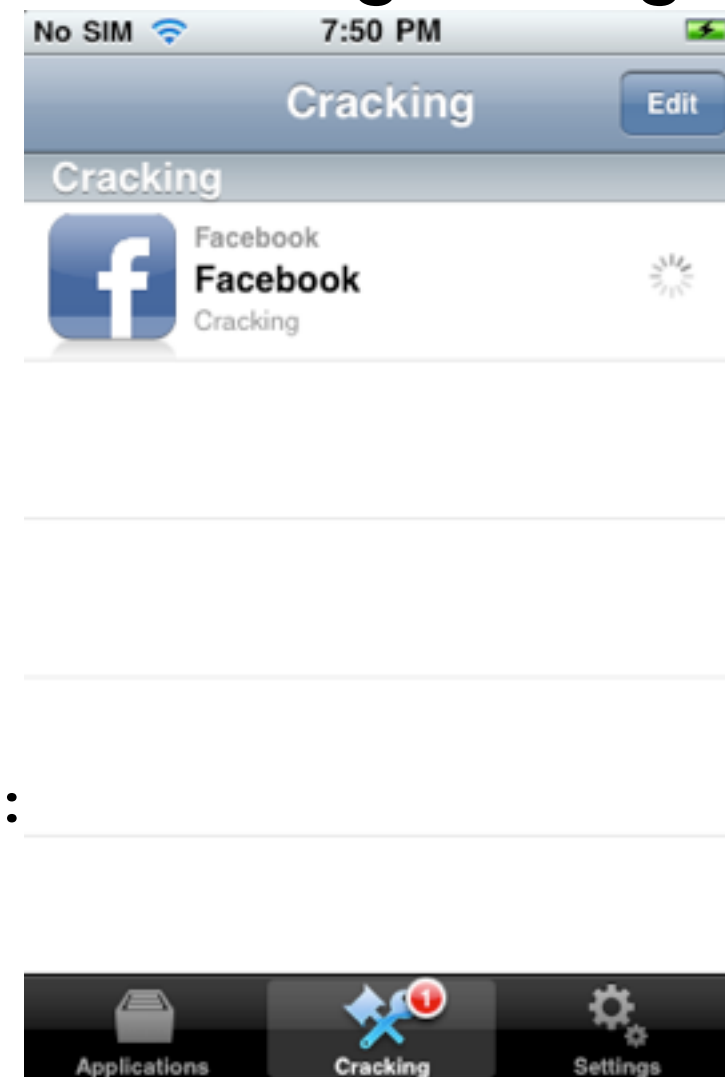
```
fb://publish/mailbox/ (initWithFolder:) / (tid:)
```

```
fb://upload/checkin/ (showUploadMenuWithCheckinID:)
```

```
fb://upload/album/ (showUploadMenuWithUID:) / (aid:)
```

```
fb://upload/actions/resume
```

...



Securely Implementing URLSchemes

► Handle the event:

```
(BOOL) application:(UIApplication *) application  
handleOpenURL:(NSURL *) url  
{  
    // Parse url <-- Careful, do thorough input  
    validation  
    // Ask for authorization  
    // Perform transaction  
}
```

► `application:handleOpenURL` is deprecated as of iOS 4.2.

► **Use** `application:openURL:sourceApplication:annotation:`
which is more secure because you get the invoker's BundleID
(SourceApplication) and a .plist (annotation).

► You can use this information to ascertain who is invoking you

Securely Implementing URLSchemes

- ▶ Still, be careful of a **Ricochet Attack** where an external website can abuse an intermediary app, i.e.

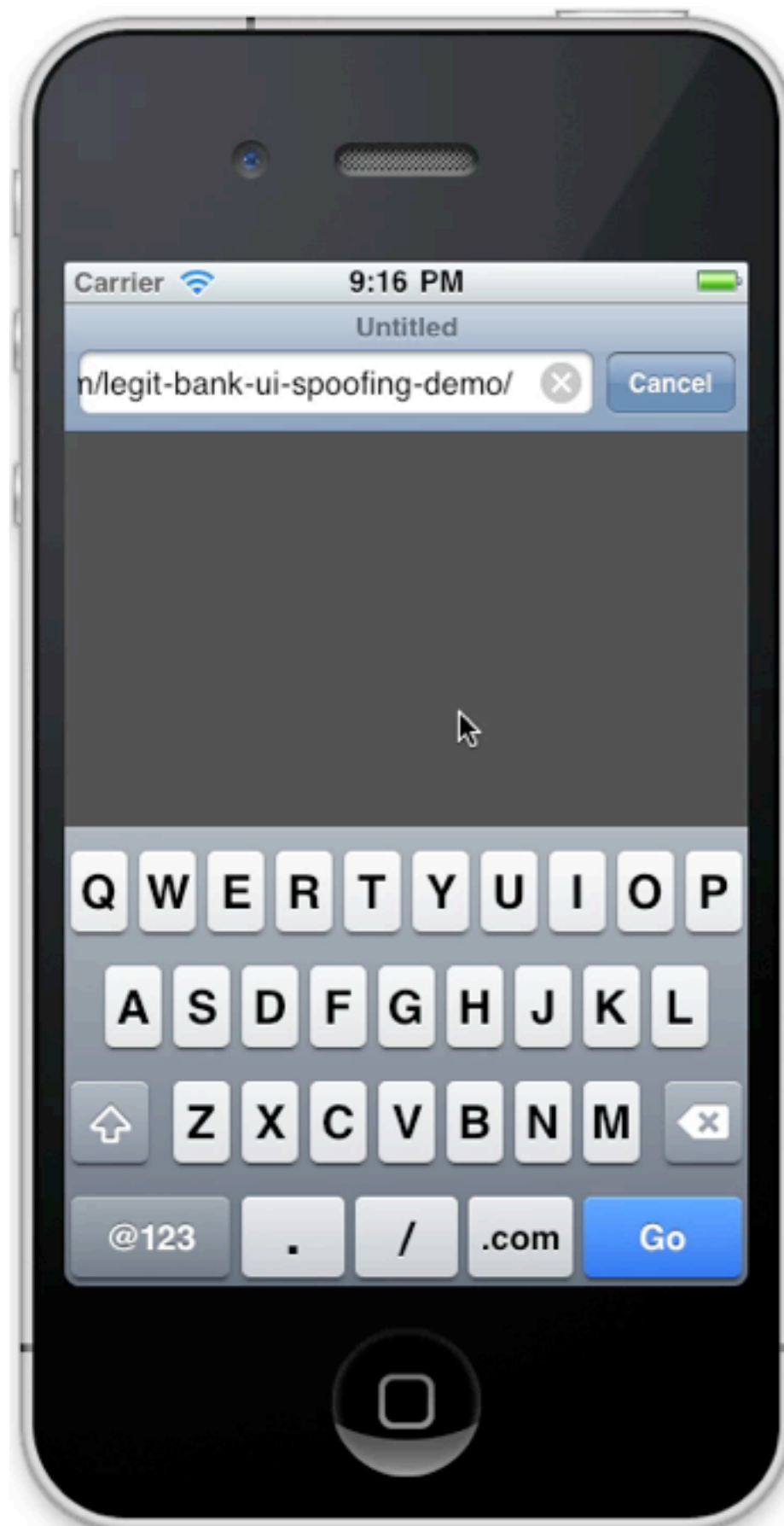
```
<iframe src="some_other_app://you://boom">
```

- ▶ Do not allow URLScheme transactions to edit/delete user data or change state
- ▶ Assume that your handler as well as the associated transaction strings are public
- ▶ Audit 3rd party Apps before allowing them into the enterprise

UI Spoofing on the iPhone



UI Spoofing on the iPhone: Video



The real website URL, i.e. <http://dhanjani.com> scrolls out of view

In this attack, we display a fake URL bar to trick the user

This can easily be deployed into phishing kits that check for iOS user-agent

Self-Pwnage: Twitter App for iPad. URL?

Can you spot the URL?

URL shorteners abound!

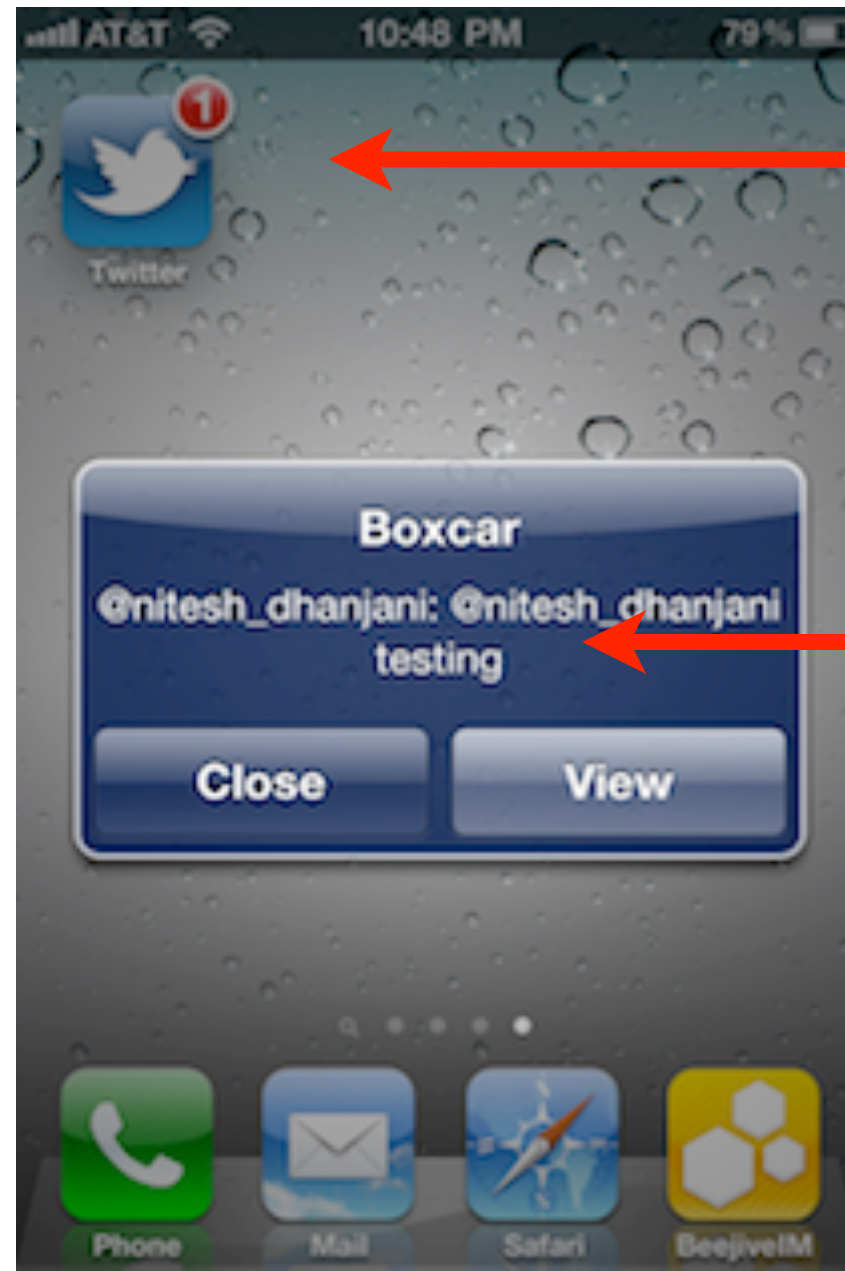
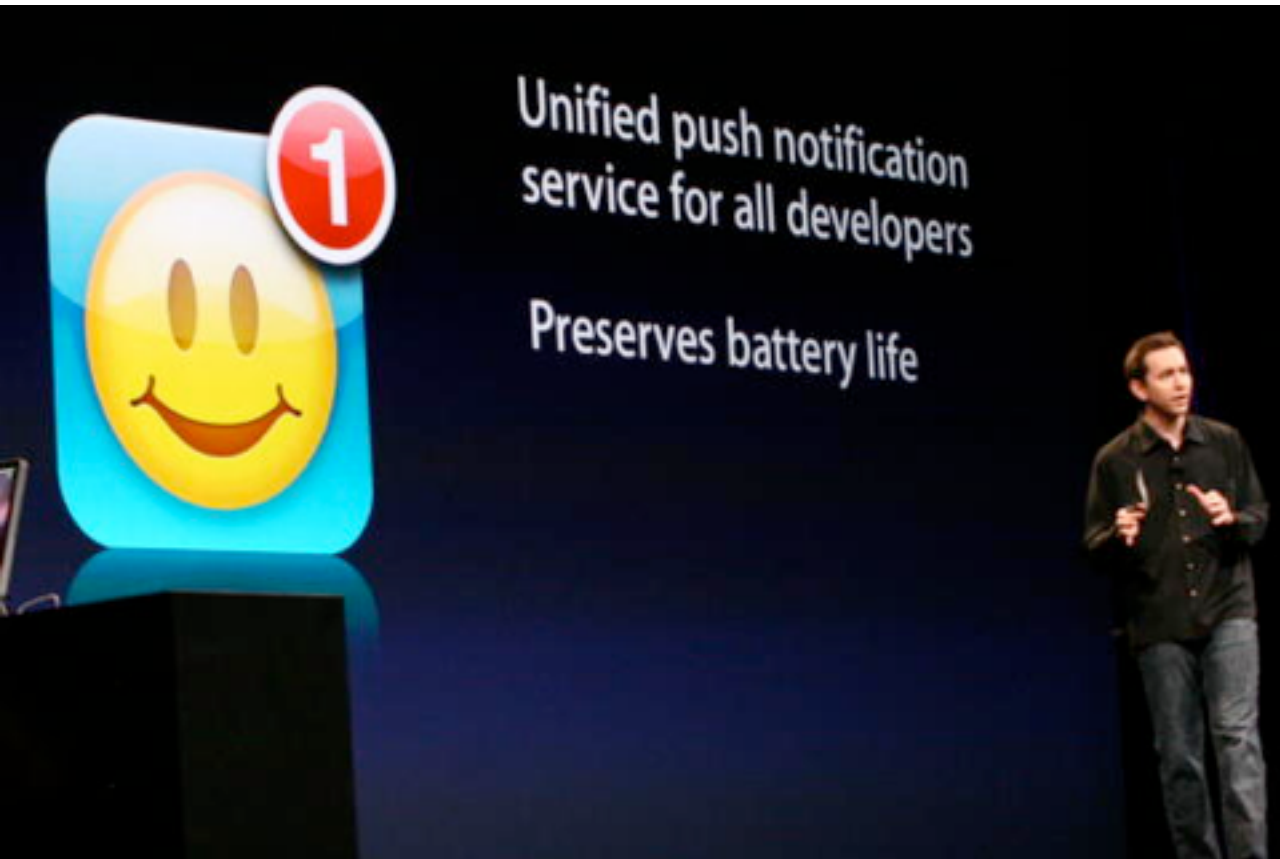
Malicious sites can check the User Agent and spoof UI

Custom Apps using UIWebView should display URL

Should Apple allocate screen real-estate (example: near the clock) to display URLs when UIWebView is invoked?



Push Notifications



Badge

+

Alert

+

Sound

I. Create an App ID on iOS Provisioning Portal

The screenshot shows the 'Add App IDs - iOS Provisioning Portal - Apple Developer' page. The browser address bar shows 'https://developer.apple.com/ios/manage/bundles/save.action'. The page has a navigation bar with 'Developer', 'Technologies', 'Resources', 'Programs', 'Support', and 'Member Center'. Below this is the 'iOS Provisioning Portal' header with a welcome message for 'Nitesh Dhanjani'. A left sidebar contains links: 'Home', 'Certificates', 'Devices', 'App IDs' (selected), 'Provisioning', and 'Distribution'. The main content area has tabs for 'Manage' and 'How To', with 'Manage' selected. The 'Create App ID' section displays a red error message: 'The bundle identifier you have specified is already in use. Please select another.' Below this, there are three main input sections: 'Description' (with a text box containing 'Push Test App'), 'Bundle Seed ID (App ID Prefix)' (with a 'Generate New' button), and 'Bundle Identifier (App ID Suffix)' (with a text box containing 'com.facebook.facebook' circled in red). An example 'com.domainname.appname' is shown next to the text box. A red arrow points from the error message to the 'Bundle Identifier' text box.

The APN trusts the “Bundle Identifier” in the cert to figure out the target App

An attempt use **com.facebook.facebook** is promptly rejected as a duplicate :-)

If you can get a cert from Apple with a duplicate Bundle Identifier, you could possibly send push notifications to another app (but will also need the device tokens)

2. Create an CSR to have Apple generate an APN

The private key stays on the desktop. The public key is in the CSR.

The APN cert that Apple provides back is specific to the app and tied to the App ID

3. Create a Provisioning Profile to deploy your App into a test iOS device with push notifications enabled for the App ID you selected

4. Export the APN certificate to the server side

You can choose to export the certificate to *.pem* format. This certificate can then be used on the server side of your app infrastructure to connect to the APN and send push notifications to targeted devices

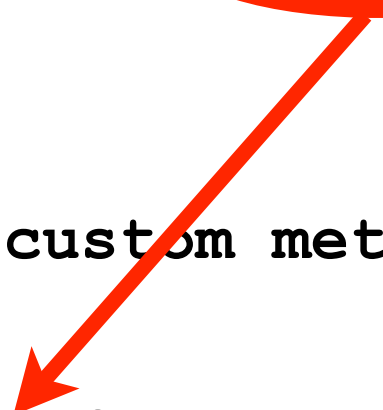
5. Code iOS applications to register for and respond to notifications

Register the Device with APN:

```
[[UIApplication sharedApplication]  
registerForRemoteNotificationTypes: (UIRemoteNotificationTypeBadge) ] ;
```

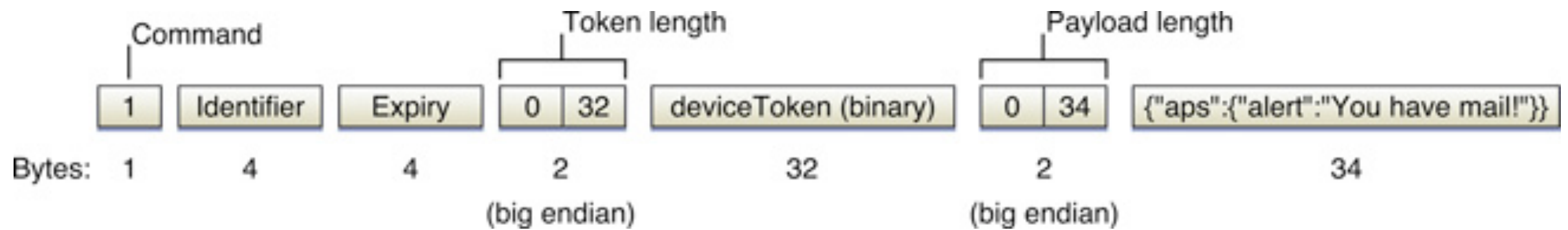
Implement the delegate:

```
- (void)application:(UIApplication *)app  
didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)devToken  
{  
    const void *devTokenBytes = [devToken bytes];  
    self.registered = YES;  
    [self sendProviderDeviceToken:devTokenBytes]; // custom method  
}
```



*NOT the same as UDID (specific to hardware).
Device Token is specific to the OS instance.*

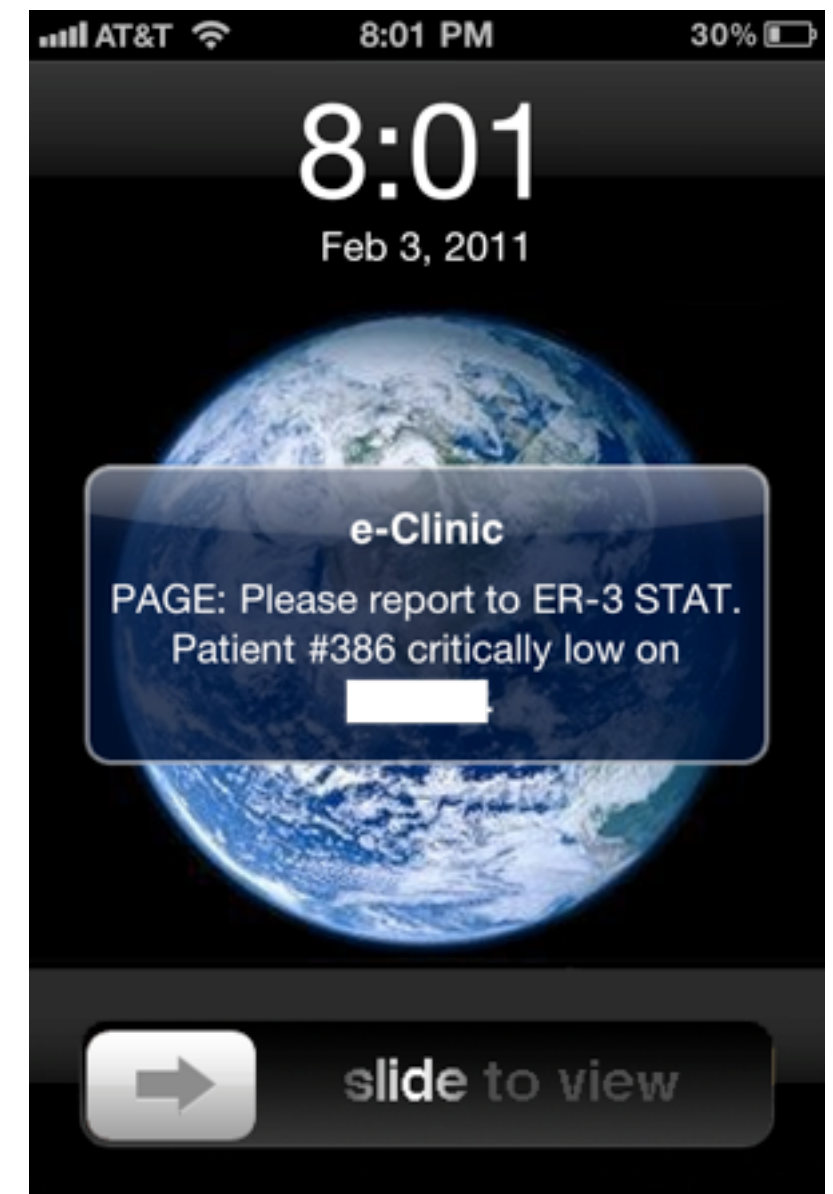
6. Implement provider communication



What Could Possibly Go Wrong?

- ▶ Do NOT send company confidential data through the APN
 - ▶ Yes, it is TLS encrypted
 - ▶ But Apple can see it
 - ▶ And even if you trust Apple, there might be legal ramifications

- ▶ Push delivery is not guaranteed so don't depend on it for critical notifications.



What Could Possibly Go Wrong?

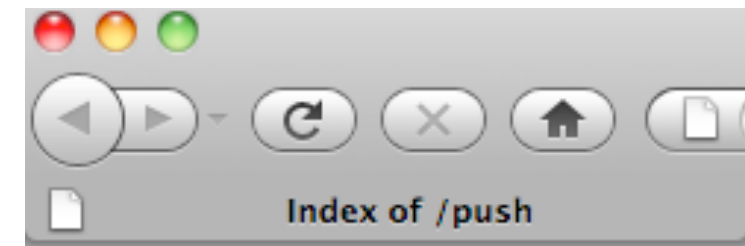
- ▶ **Do not allow the push notification handler to modify user data.** The application should not delete data as a result of the application launching in response to a new notification
- ▶ **Validate outgoing connections to the APN.** The root Certificate Authority for Apple's certificate is Entrust
- ▶ **Be careful with unsafe API.** Be careful with memory management and perform strict bounds checking (example: `memcpy`)

What Could Possibly Go Wrong?

► **Do not store your SSL certificate and list of deviceTokens in your web-root.**

From: [REDACTED]
Subject: **Re: Your private keys are showing**
Date: February 5, 2011 4:36:46 PM PST
To: Nitesh Dhanjani <nitesh@dhanjani.com>

[Show in Mailbox](#)



Thank you sir! Not sure how you found that but I appreciate you alerting me.

[REDACTED]
Sent from my iPhone

On Feb 5, 2011, at 1:52 PM, Nitesh Dhanjani <nitesh@dhanjani.com> wrote:

Hello,

You might want to stop exposing your private keys that you have with Apple's push notification servers:

[http://www.\[REDACTED\]/push/](http://www.[REDACTED]/push/)

I'd suggest moving these out of your webroot so the bad guys can't use them to send push notifications to your customers (assuming they have the device-tokens).

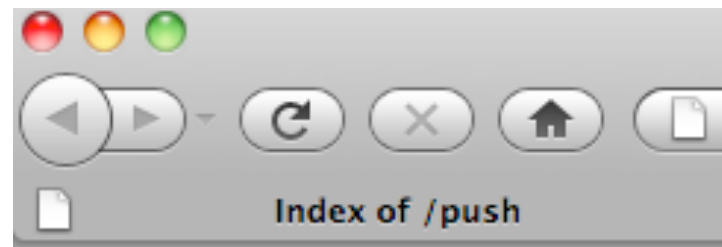
Hope this helps,

Thanks
Nitesh.

Index of /push

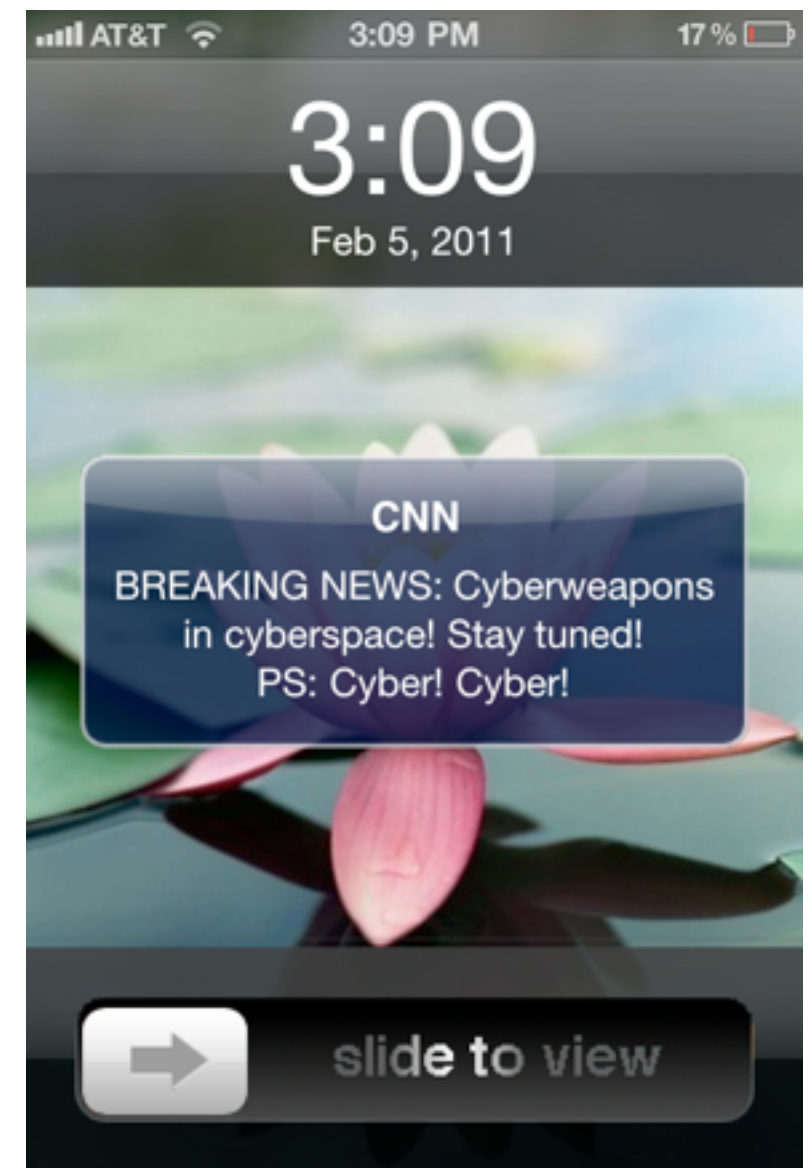
- [Parent Directory](#)
 - [.DS_Store](#)
 - [apns-prod.pem](#)
 - [apns-prod2.pem](#)
 - [apns.log](#)
 - [view_device_ids.php](#)
- Apache/2.2.16 (Unix) mo*

What Could Possibly Go Wrong?



Index of /push

- [Parent Directory](#)
 - [.DS_Store](#)
 - [apns-prod.pem](#)
 - [apns-prod2.pem](#)
 - [apns.log](#)
 - [view_device_ids.php](#)
- Apache/2.2.16 (Unix) mo*



```
APNS.host='gateway.push.apple.com'  
APNS.pem = '/path/to/pwn3d/pem/file'  
APNS.pass=''  
APNS.port=2195
```

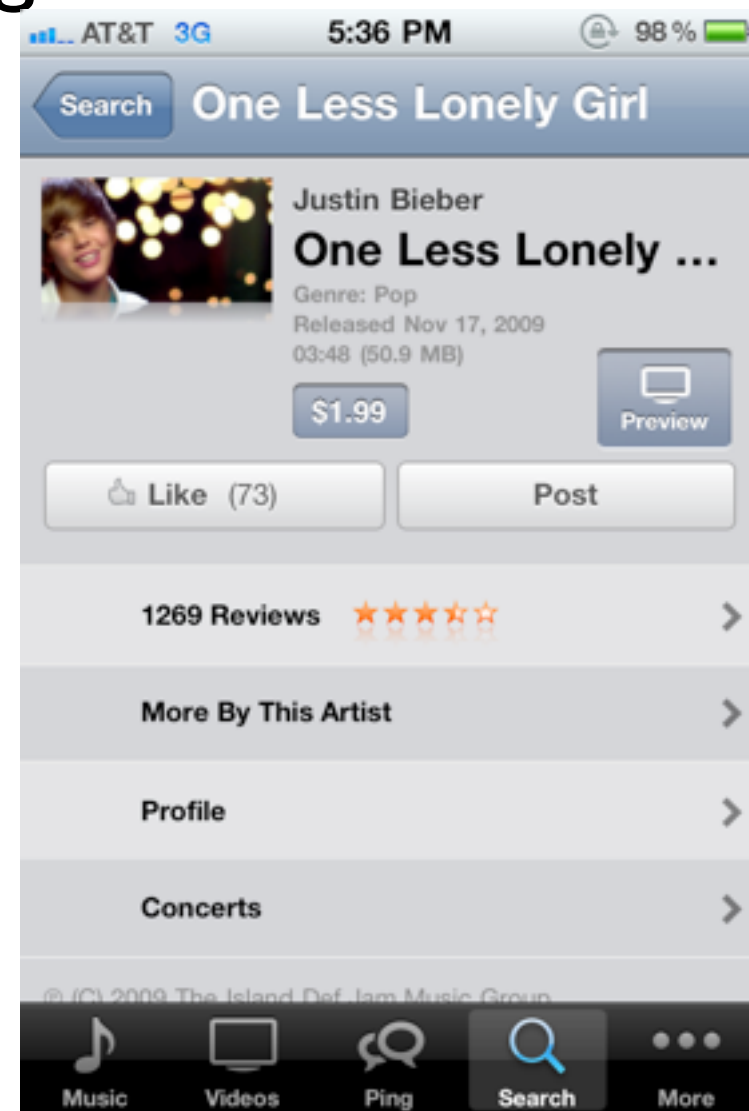
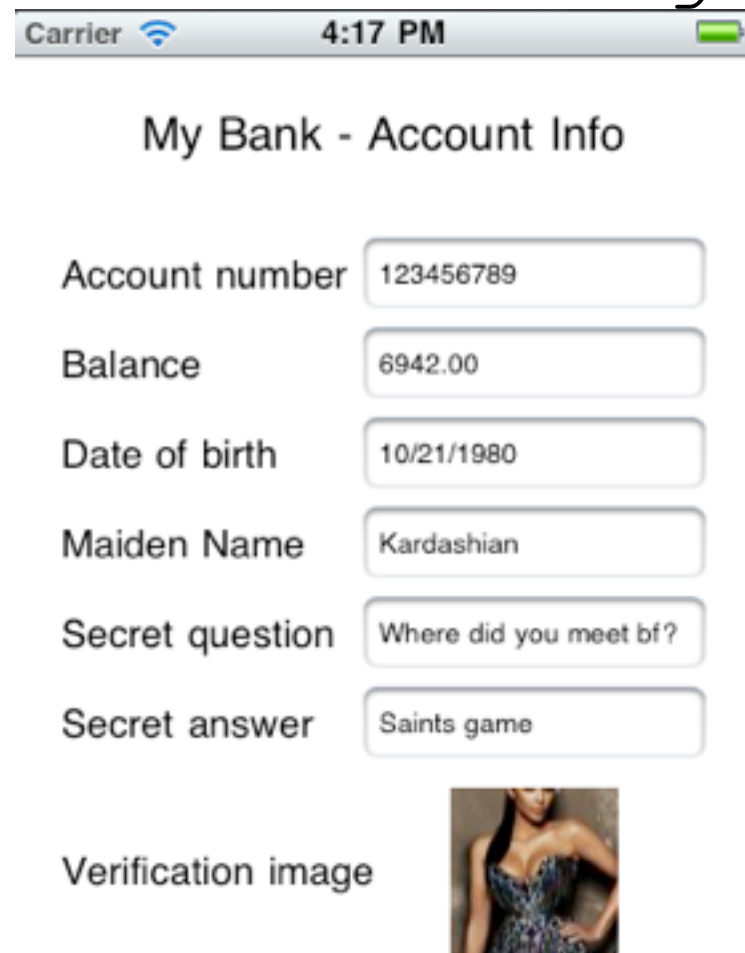
```
stolen dtokens.each do |dtoken|  
  APNS.send_notification  
    (dtoken, 'pwn3d')  
end
```

A Word About File Encryption / Data Protection

- ▶ Every file is individually protected
- ▶ You can also tie the user's passcode to the encryption mechanism
- ▶ Filesystem: Use `NSDataWritingFileProtectionComplete`
- ▶ KeyChain: Use `kSecAttrAccessibleWhenUnlocked` or `kSecAttrAccessibleAfterFirstUnlock`

A Word About File Encryption / Data Protection

- ▶ iOS takes screenshots of the App when the user presses the home button to animate transition
- ▶ It is recommended that the App set `window.hidden` to YES in the `DidEnterBackground` delegate



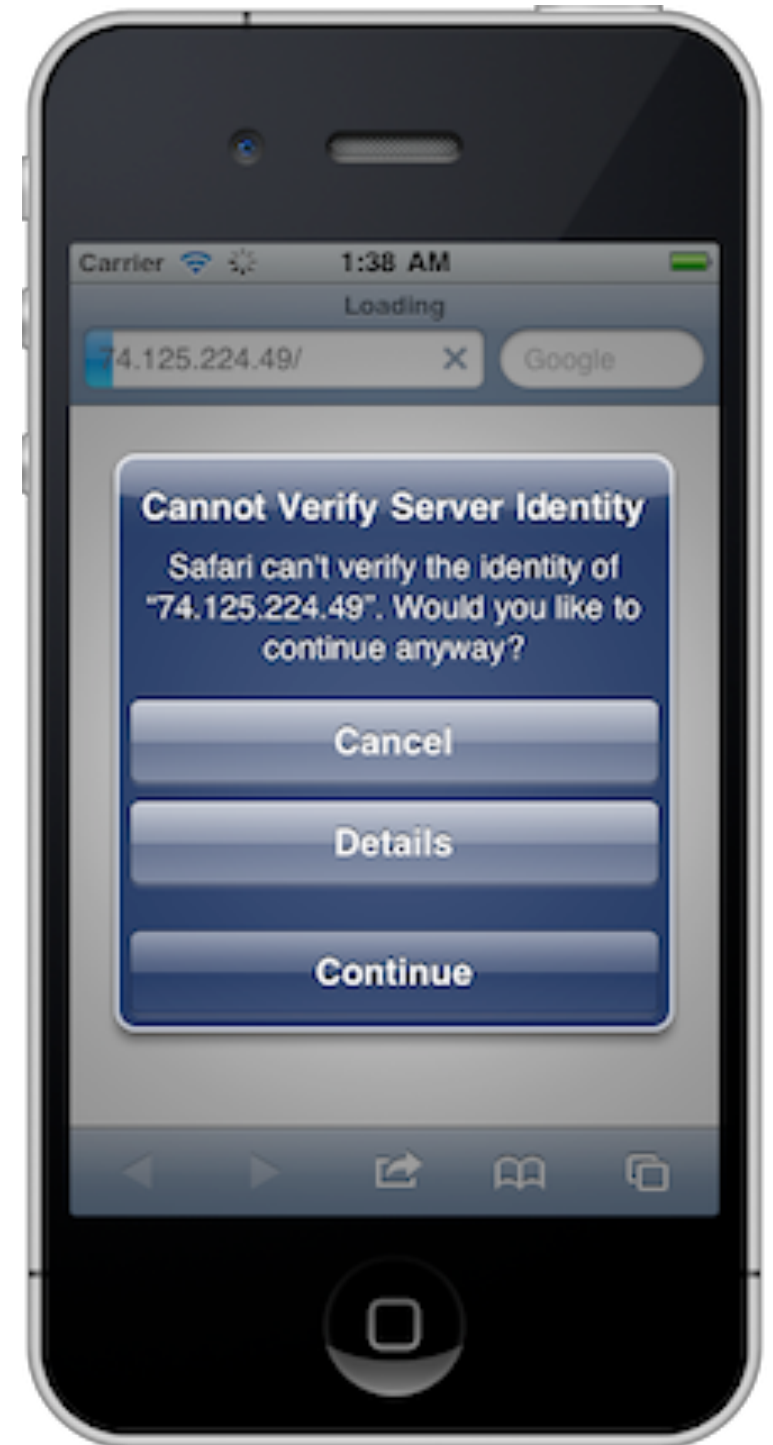
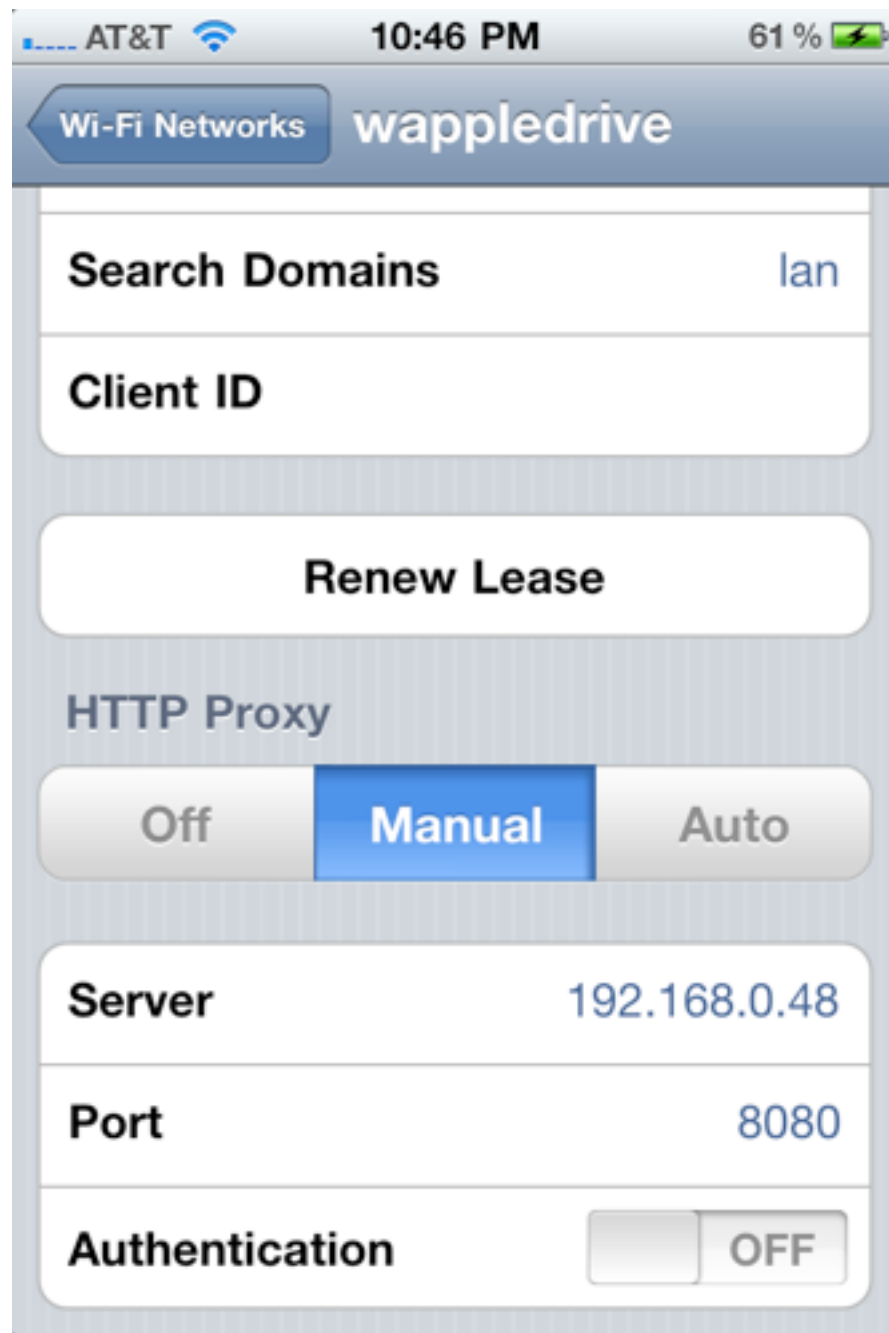
You don't want your private data exposed in screenshots;
Specifically the fact that you watch Justin Bieber music videos

“Upside Down Ternet” Treatment

- ▶ Take your Linux box to a Starbucks and broadcast a “FREE WIFI” SSID with the “Upside Down Ternet” NAT setup [<http://www.ex-parrot.com/pete/upside-down-ternet.html>]
- ▶ iPhone and iPad users will see images “upside down” as they browse or even use Apps
- ▶ Watch them hilariously try to rotate their iPhones and iPads as you sip your latte *like a boss*

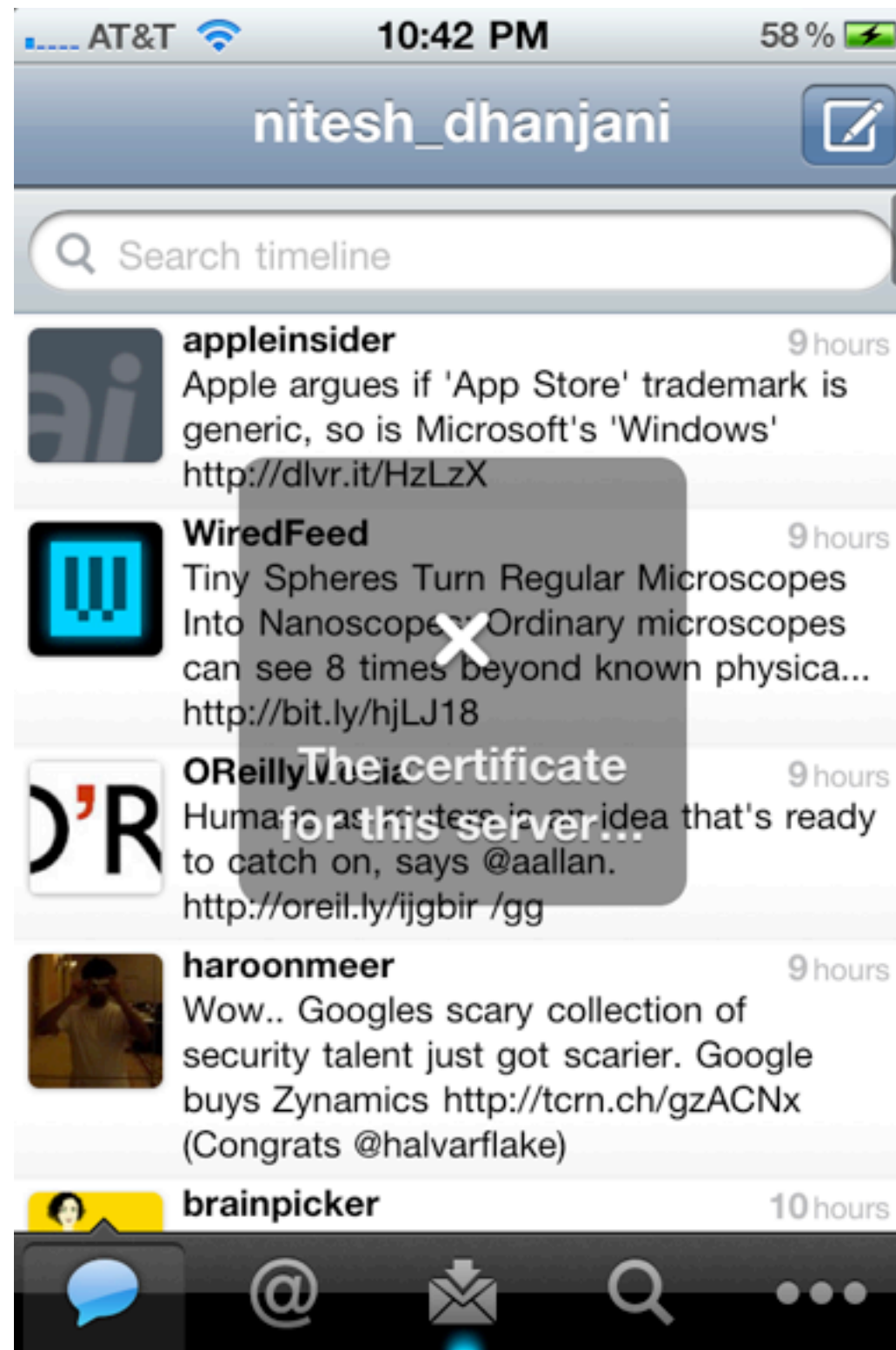


Intercepting HTTP(S) Traffic



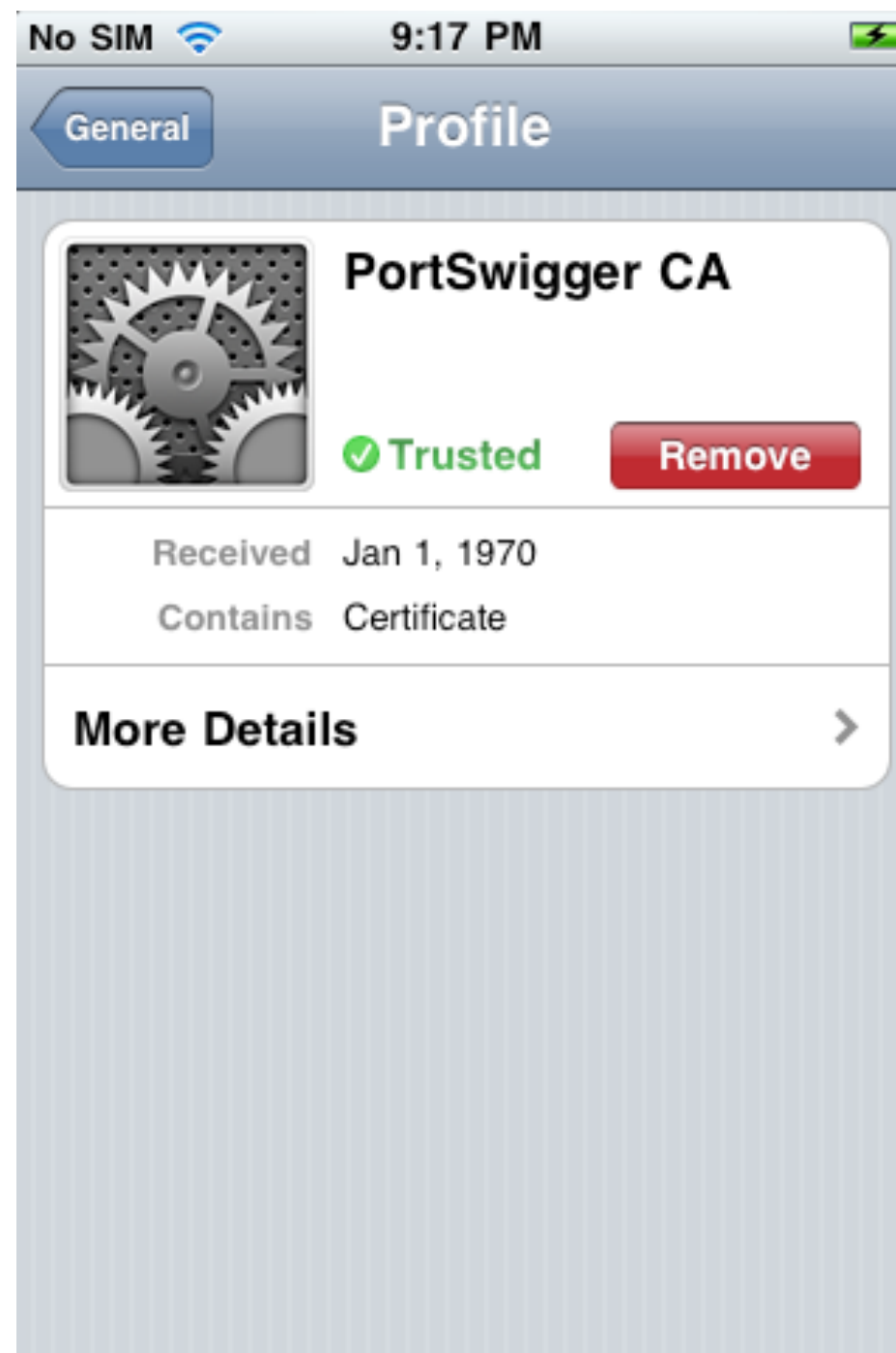
Using an HTTP proxy such as Burp will cause Safari on iOS to warn you of the (Common Name) mis-match in the SSL cert used by Burp

Intercepting HTTP(S) Traffic



Apps such as Twitter will out-right fail. This is a good thing.

Intercepting HTTP(S) Traffic



Just install the self-signed Burp/PortSwigger CA.

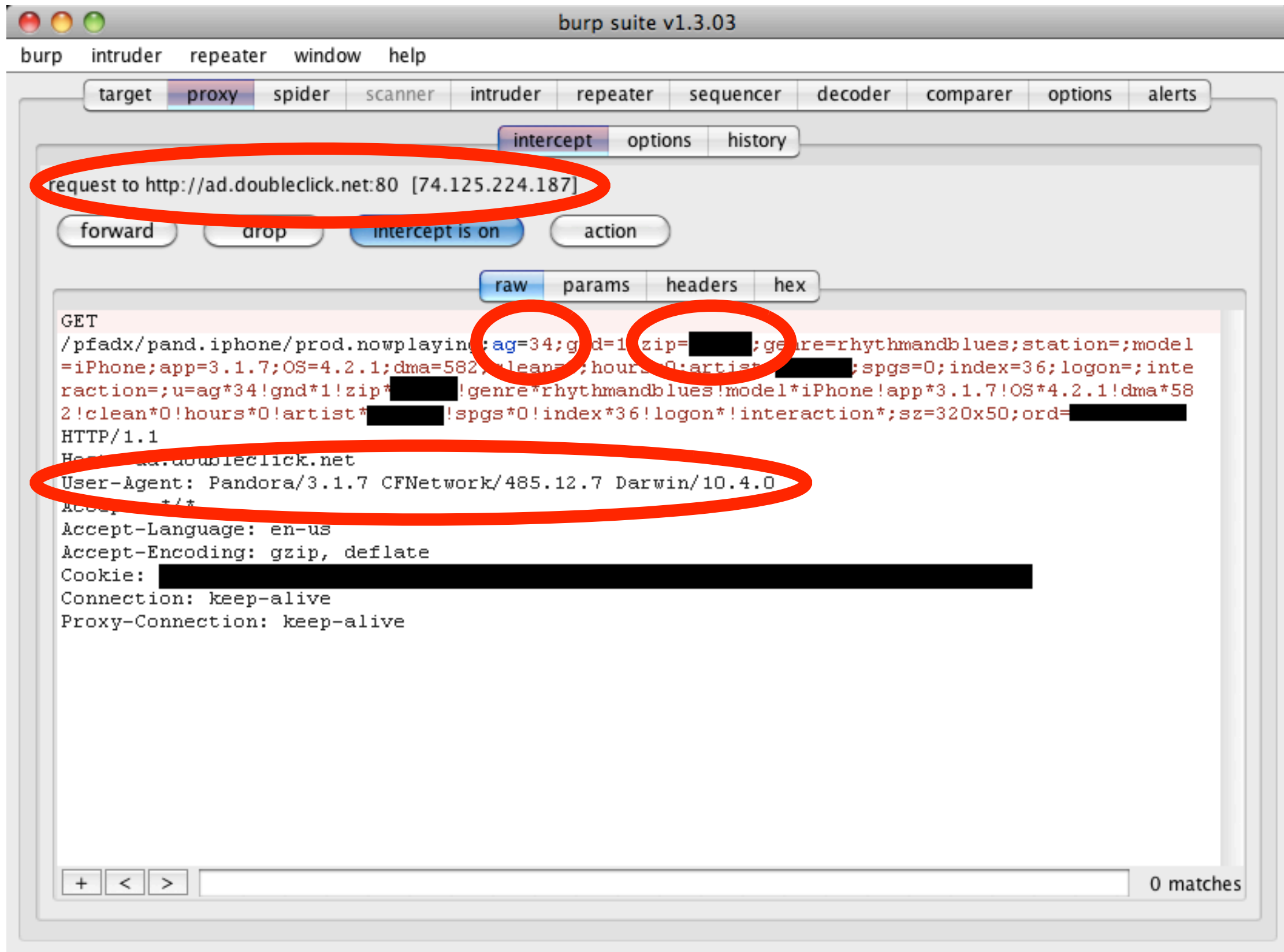
Code to Bypass Certificate Mismatch Check

```
- (BOOL)connection: (NSURLConnection *) connection  
canAuthenticateAgainstProtectionSpace: (NSURLProtectionSpace *)  
protectionSpace  
{  
    return [protectionSpace.authenticationMethod  
isEqualToString:NSURLAuthenticationMethodServerTrust] ;  
}  
  
- (void)connection: (NSURLConnection *) connection  
didReceiveAuthenticationChallenge: (NSURLAuthenticationChallenge  
*) challenge  
{  
    [challenge.sender useCredential: [NSURLCredential  
credentialForTrust: challenge.protectionSpace.serverTrust]  
forAuthenticationChallenge: challenge] ;  
}
```

[Can also call Private API `setAllowsAnyHTTSPSCertificate:forHost:` to bypass this check]

Obviously a bad idea. So, yeah, don't do this.

Your Age is Showing. Thank Pandora



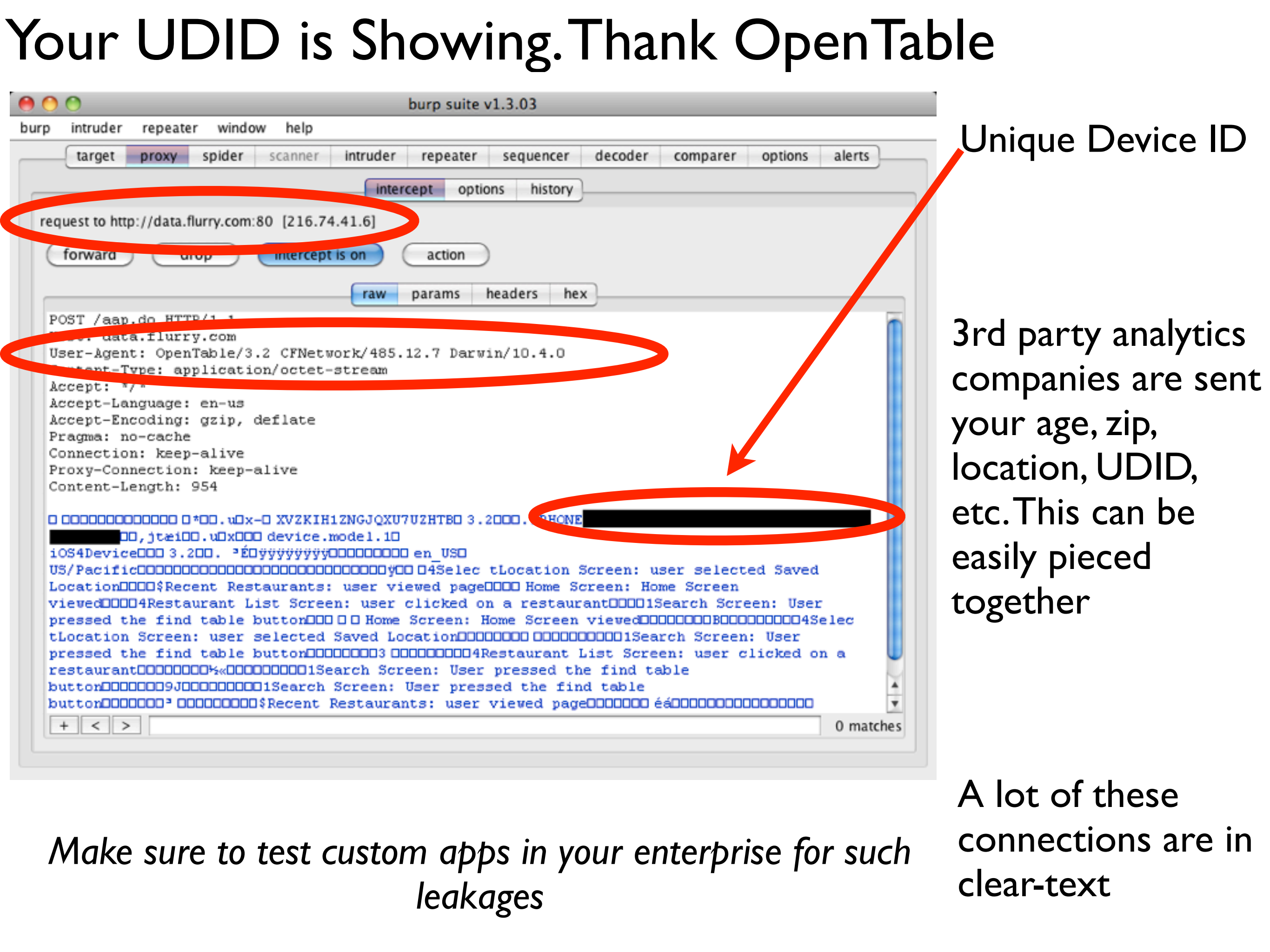
Your UDID is Showing. Thank OpenTable

Unique Device ID

3rd party analytics companies are sent your age, zip, location, UDID, etc. This can be easily pieced together

A lot of these connections are in clear-text

Make sure to test custom apps in your enterprise for such leakages



Your UDID is Showing. Thank OpenTable

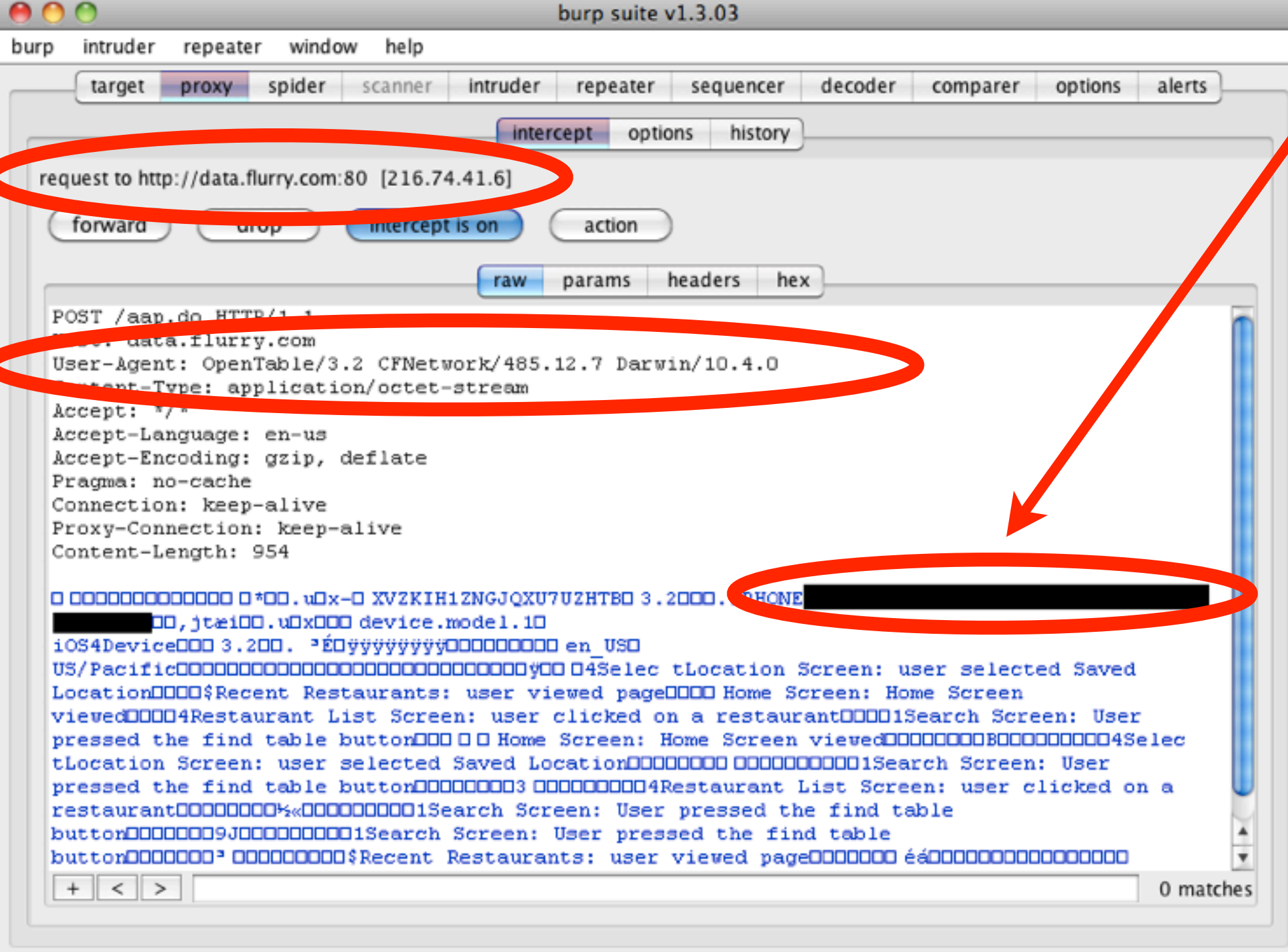
Unique Device ID

3rd party analytics companies are sent your age, zip, location, UDID, etc. This can be easily pieced together

A lot of these connections are in clear-text

Make sure to test custom apps in your enterprise for such leakages

Your UDID is Showing. Thank OpenTable



Unique Device ID

3rd party analytics companies are sent your age, zip, location, UDID, etc. This can be easily pieced together

A lot of these connections are in clear-text

Make sure to test custom apps in your enterprise for such leakages

Your UDID is Showing. Thank OpenTable

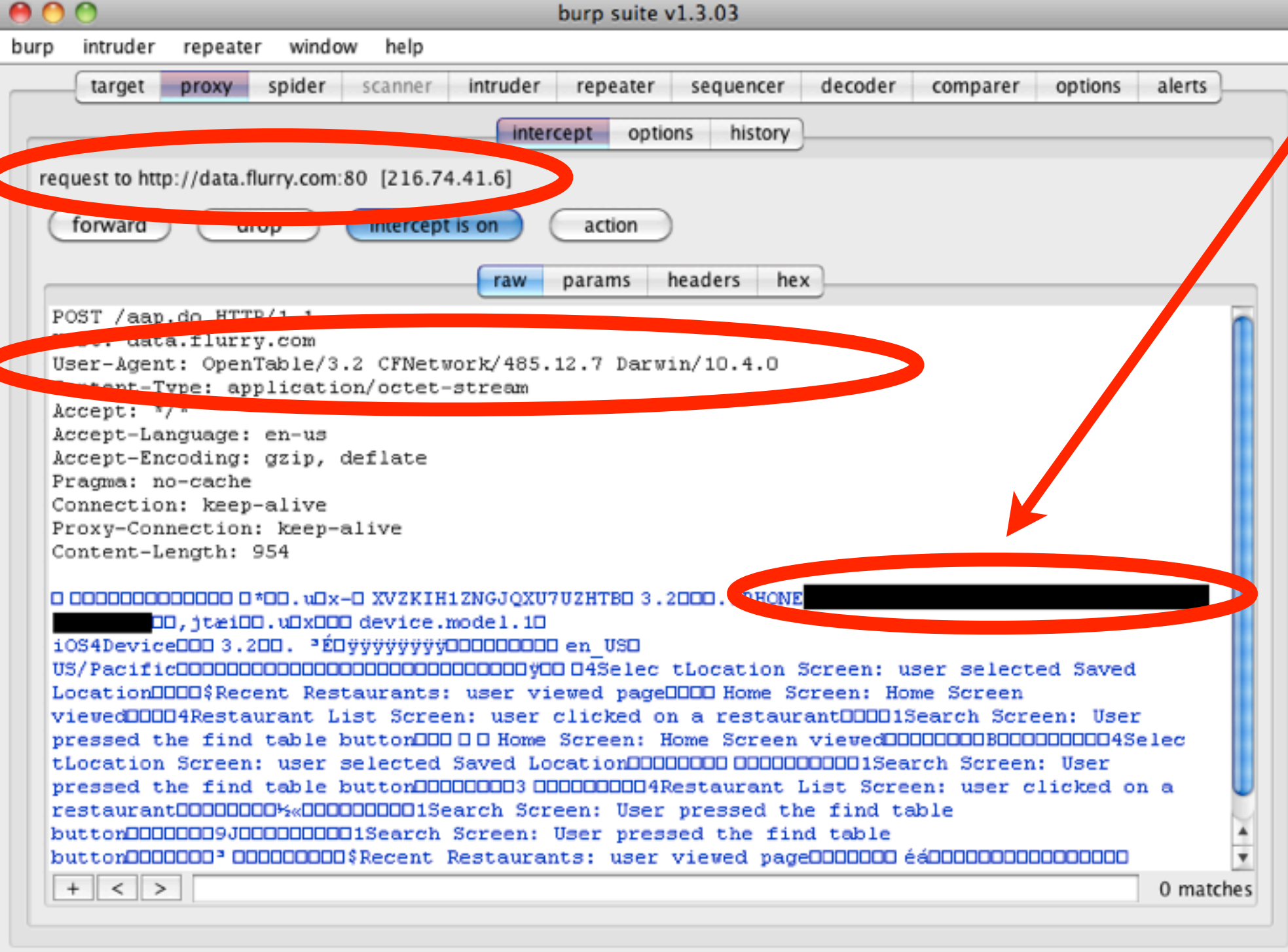
Unique Device ID

3rd party analytics companies are sent your age, zip, location, UDID, etc. This can be easily pieced together

A lot of these connections are in clear-text

Make sure to test custom apps in your enterprise for such leakages

Your UDID is Showing. Thank OpenTable



Unique Device ID

3rd party analytics companies are sent your age, zip, location, UDID, etc. This can be easily pieced together

A lot of these connections are in clear-text

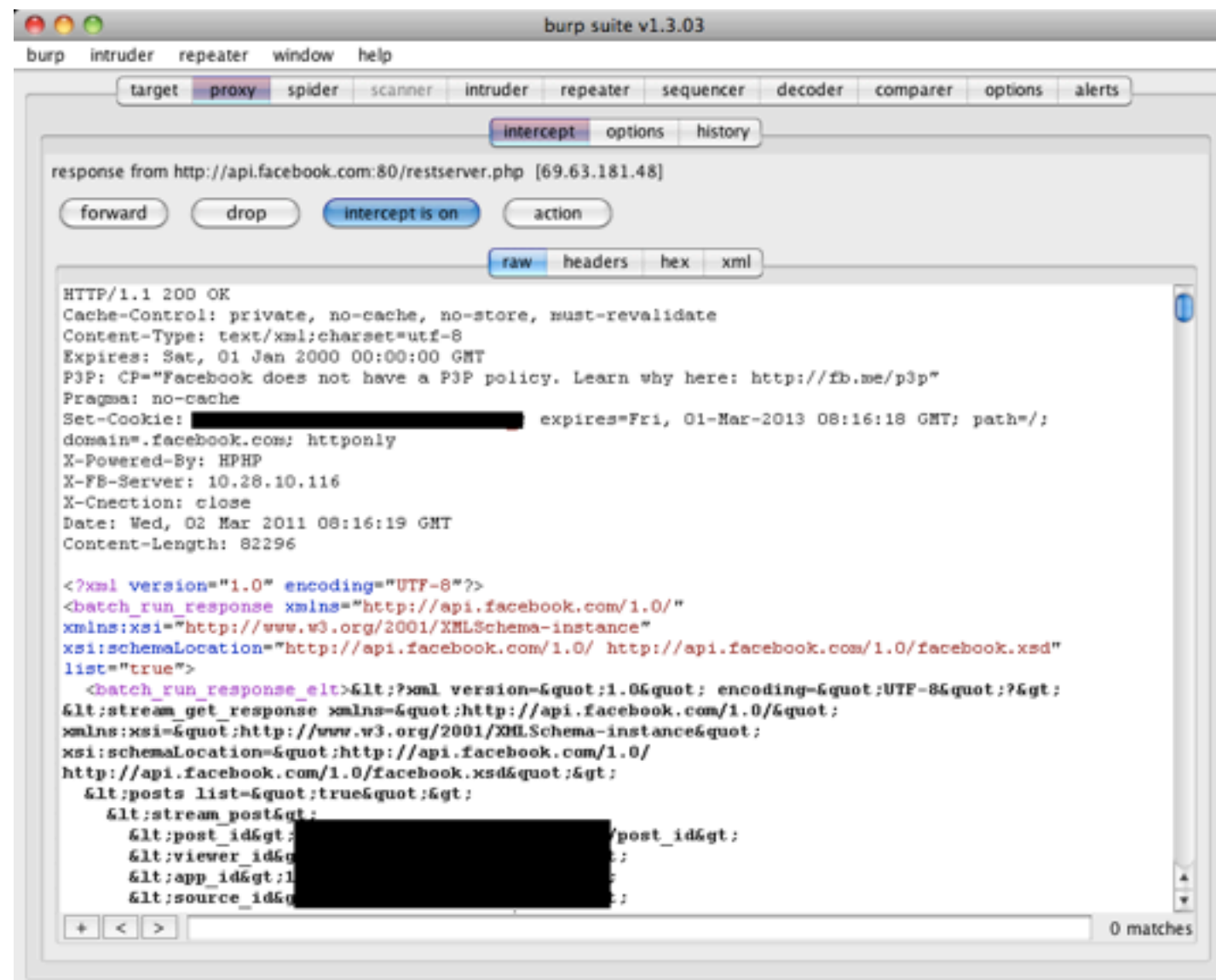
Make sure to test custom apps in your enterprise for such leakages

MiTM Forced De-cloaking

- ▶ Take your Linux box to a Starbucks, again
- ▶ Get folks to associate to your AP
- ▶ MiTM their traffic and inject the following

`<iframe src="fb://profile/"></iframe>`

- ▶ The user will be yanked into the Facebook App onto his or her own profile
- ▶ You have now de-cloaked their identity and scraped their Facebook wall. Congratulations!



Recommendations: Network Channel

- ▶ Use SSL! Seriously. It's 2011
- ▶ Audit your apps and 3rd party apps you buy
- ▶ Check to see if the Apps you most depend upon are not bypassing SSL exceptions
- ▶ Try not to leak data to 3rd party analytics services
- ▶ Check to see if any of the apps you use in the enterprise be abused to de-cloak identities

Location Services



- ▶ Primarily an experience among friends
 - ▶ Loopt: ~3million users
 - ▶ Foursquare: 6million users
 - ▶ Gowalla: ~1million users
 - ▶ Facebook Places: ~30-50million
- ▶ Originally not much incentive to “game” the system (a good thing)

Gaming the System

Description:
with Foursquare, be anywhere you want to be by venue id

References:
<http://groups.google.com/group/foursquare-api>
<http://www.mikekey.com/im-a-foursquare-cheater/>

```
msf auxiliary(foursquare) >  
msf auxiliary(foursquare) > set USERNAME notmyusername@host.com  
USERNAME => notmyusername@host.com  
msf auxiliary(foursquare) > set PASSWORD notmypassword  
PASSWORD => notmypassword  
msf auxiliary(foursquare) > set VENUEID 9186  
VENUEID => 9186
```

```
msf auxiliary(foursquare) > run
```

```
[*] HTTP/1.1 200 OK  
Content-Type: text/xml; charset=utf-8  
Date: Fri, 19 Mar 2010 13:59:28 GMT  
Content-Length: 1311  
Server: nginx/0.7.64  
Connection: keep-alive
```

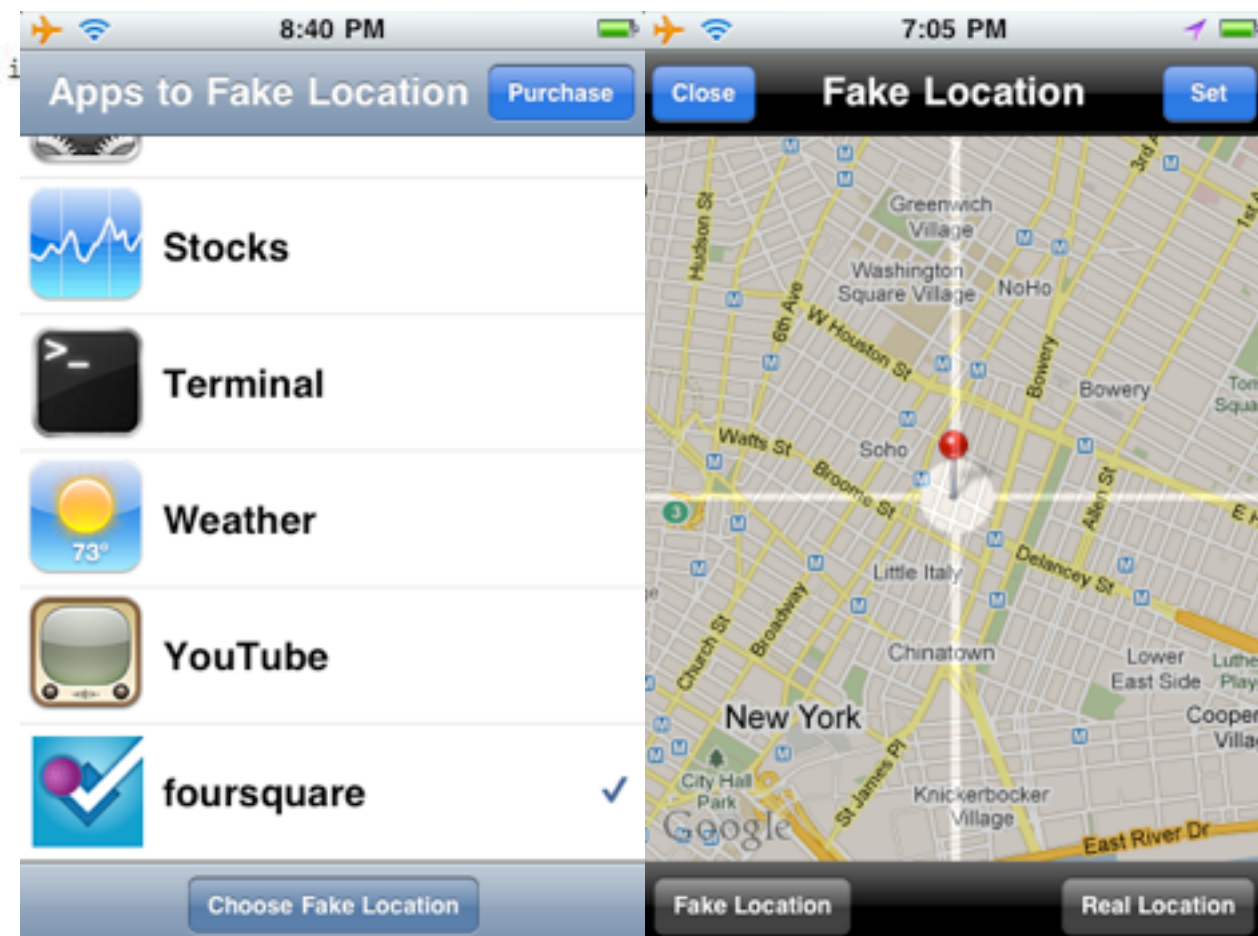
```
Fri, 19 Mar 10 13:59:28 +0000OK! We've got you @ Washington Monument. This is  
your 1st checkin here!9186Washington Monument  
79199Parks & Outdoors:Sculpture SNIP
```

```
[*] Auxiliary module execution completed
```

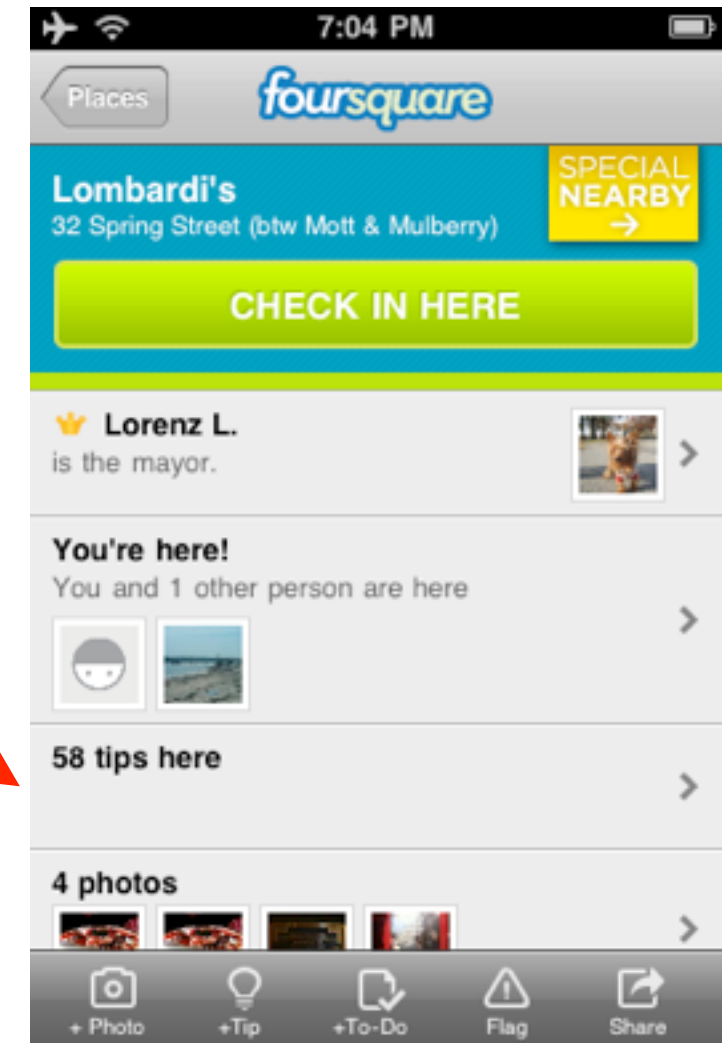
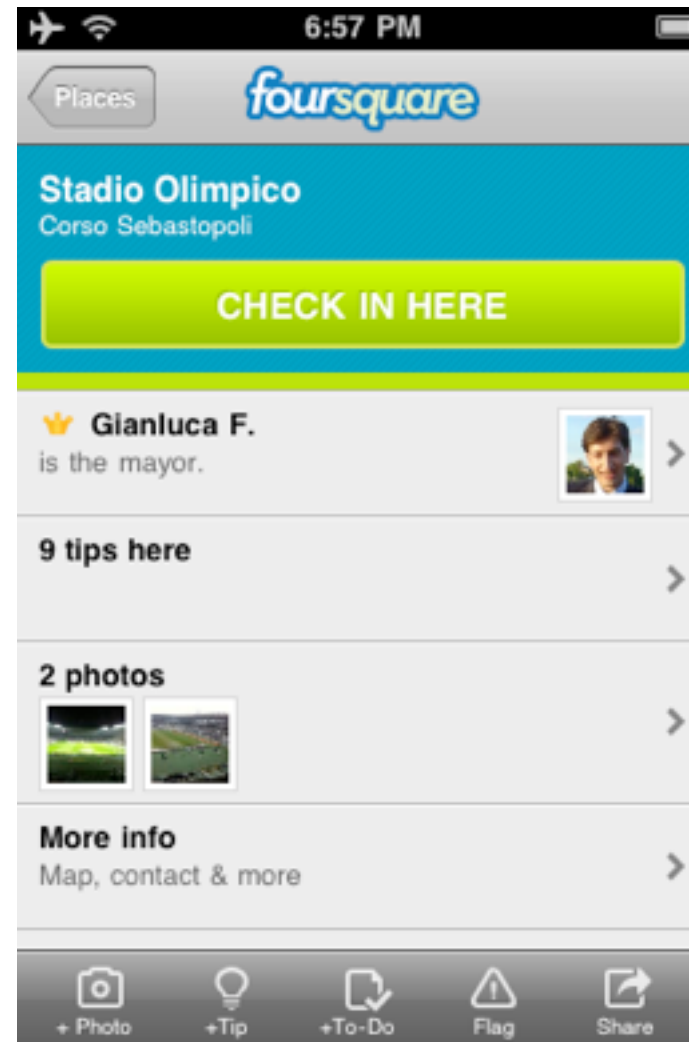
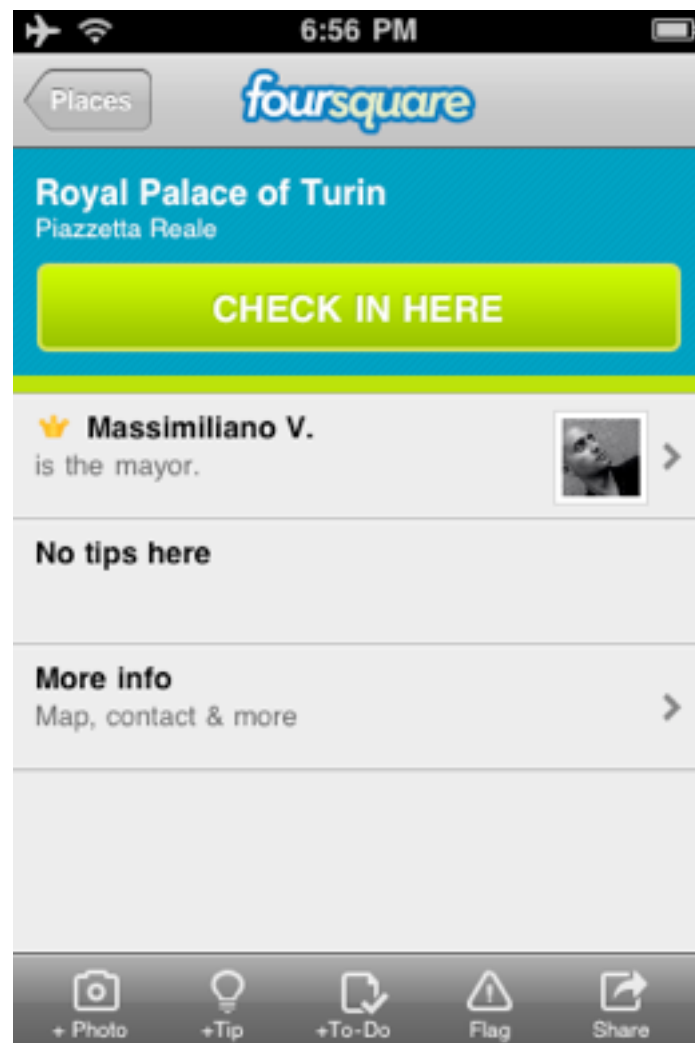
Metasploit module to implement
foursquare spoofing (only need valid
venueid)

FakeLocation available on
jailbroken phones through Cydia

Manually set your location and pick
apps to fake



Globetrotting



3.9 miles in
1 minute
Not bad

~3974 miles in
7 minutes
Probably not!

What Could Go Wrong?

- ▶ Businesses and enterprises are moving to use them for advertising, promotions, and productivity
- ▶ Business-focused component:
- ▶ Where is this going:
 - ▶ Mobile coupons/discounts
 - ▶ Personnel Tracking
 - ▶ Mileage accrual
- ▶ Enterprise security teams must be aware that certain applications are aware of the employee locations

Recommendations: Location Services

► Many iOS apps currently check this on the device itself:

```
- (void) locationManager: (CLLocationManager *) manager  
    didUpdateToLocation: (CLLocation *) newLocation  
    fromLocation: (CLLocation *) oldLocation
```

```
NSDate* newEventDate = newLocation.timestamp;  
NSDate* oldEventDate = oldLocation.timestamp;
```

► You're still trusting the client!

► Your sever-side application must be responsible for validating the check-in

► Two factor check-ins for sensitive operations (e.g., discounts for physical check-ins)

► Cell tower triangulation

► Any apps in use within the enterprise could be exposing location data to external parties; consider whitelisting apps that are able to access Location data



Take Aways

- ▶ The iOS platform introduces new ways of approaching mobile app development
- ▶ Follow precautions and best practices before accepting 3rd party Apps into the enterprise or coding your own
- ▶ Don't forget the traditional lessons
- ▶ Apply some of the thought processes presented in this discussion into your assessment methodology:
 - ▶ URLSchemes
 - ▶ (clear-text) Network channels
 - ▶ UI Spoofing
 - ▶ Push messaging
 - ▶ Data protection
 - ▶ Location awareness
- ▶ Leveraging individuals with talent in the field of iOS to test your enterprise applications and hook into the SDLC

?

Appendix: iOS Application Assessment Checklist

- ☐ Input and Output validate every dynamic input (user input, external HTML or database feed, URLs)
- ☐ Audit traditional unsafe methods that deal with memory (`memcpy`, `strcpy`, etc)
- ☐ Watch out for format string vulnerabilities
- ☐ `grep` for password strings and hard coded credentials / secrets
- ☐ `grep` for `NSURL`, `CFStream`, `NSStream` to locate network connections
- ☐ `grep` for SQL strings and SQLite queries
- ☐ Look for `setAllowsAnyHTTPSCertificate` and `didReceiveAuthenticationChallenge` to see if certificate exceptions are being bypassed
- ☐ Locate calls to `NSLog` to see what data is being logged
- ☐ Check implementation of `URLSchemes` in `handleOpenURL`
- ☐ Ensure information is being secured in the Keychain (`kSecAttrAccessibleWhenUnlocked` or `kSecAttrAccessibleAfterFirstUnlock` attributes when calling `SecItemAdd` or `SecItemUpdate`) and the file system (`NSDataWritingFileProtectionComplete`).
- ☐ Make sure `NSUserDefaults` is not being used to store critical data

Appendix: iOS Application Assessment Checklist

- ☐ Take a look at the server side code and web-root, including implementations and payloads sent to the APN. Make sure APN certs are protected by a pass-phrase
- ☐ Pay attention to `UIWebView` implementations: Where is the HTML being rendered from? Is the URL always visible?
- ☐ Make sure Copy-Paste functionality is disabled in sensitive fields (PHI, PII)
- ☐ Make sure UI fields that display critical data hide themselves in `applicationWillTerminate` and `applicationDidEnterBackground` to prevent screenshot caching
- ☐ Run the App and monitor data (Jailbreak/SSH or a tool such as PhoneView)
- ☐ Decrypt the binary and run 'strings'
- ☐ Install Burp CA and monitor + fuzz HTTP/HTTPS traffic
- ☐ Watch out for leakage of UDID and/or PII/PHI to 3rd party analytics services or in clear-text
- ☐ Make sure the server side architecture does not rely upon the iOS device to truthfully state its location (since this data can be intercepted and modified)