



security-assessment.com

iOS Applications Different Developers Same Mistakes



Paul Craig – Security-Assessment.com
Syscan 2012

- **Hello!**
 - My name is Paul Craig
 - Team Lead @ Security-Assessment.com Asia
 - Based here in Humid Singapore
- **I am a Penetration Tester with a Passion for Hacking.**
 - Me and my team live to hack the Singaporean Financial Services Industry.
 - We find bugs, exploit said bugs, and steal your cash..
 - “Increase the security bar of Singapore”



“So what do you do here ?”

“Oh, I mostly hack into banks and steal money... You?”

- **Singapore's Love Affair With Mobile Technology.**

- I moved to Singapore a year ago from New Zealand
- New Zealand is a bit different when it comes to technology....

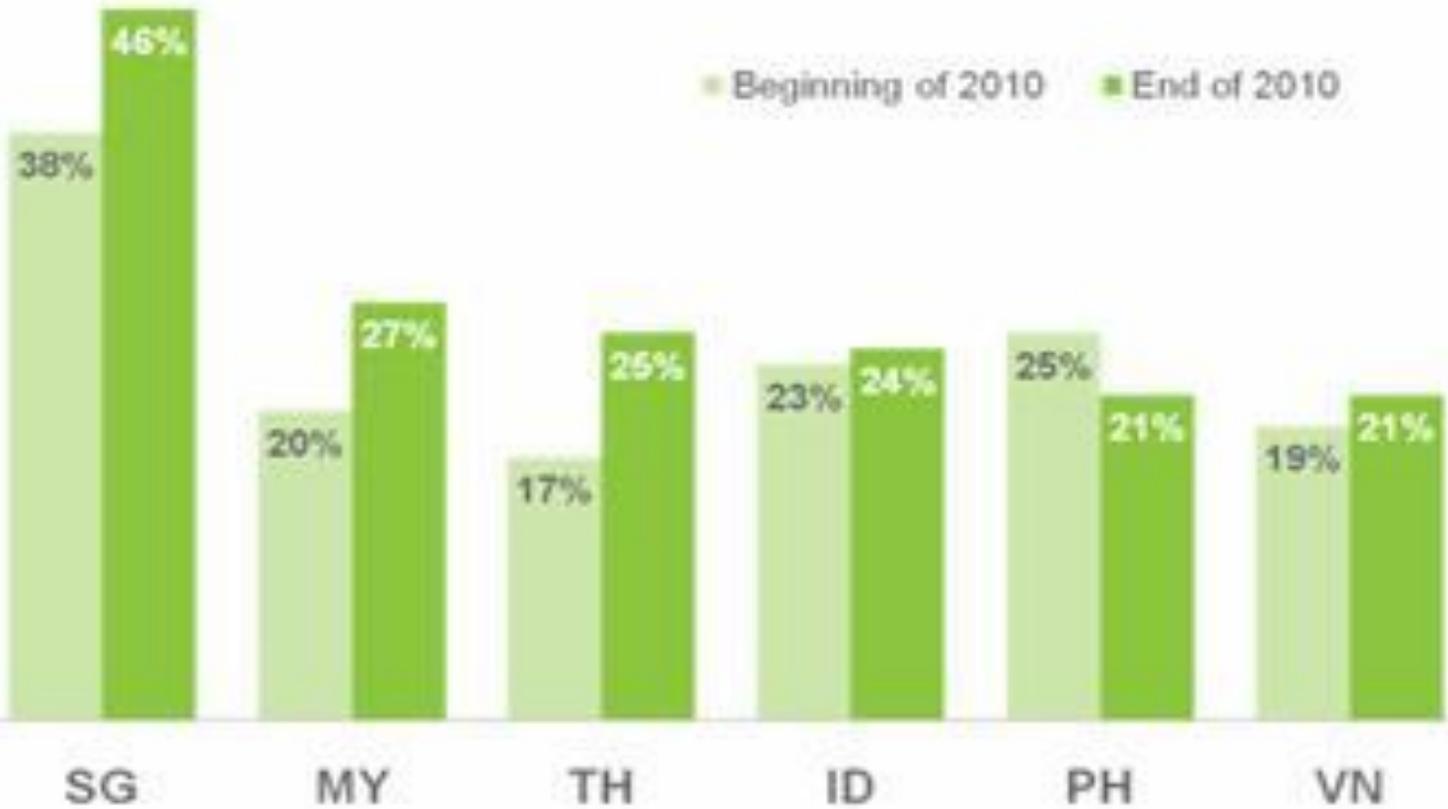
When I first moved here, I could believe just how **connected everyone is!**



“It’s called a camera-phone Brett”
Sadly this is not far from the truth.

Yep, I'm that creepy guy who takes photos of you on the MRT

Smartphone Ownership in Southeast Asia



Source: Nielsen Global Online Omnibus

rt-phones.
vice.
a rates.





■ APPS! APPS! APPS!

- Singaporean companies are rapidly pushing out iOS Apps
- Banking, Entertainment, Government, Finance.
- Singaporean Government alone has (or has participated in) 58!

In New Zealand mobile applications made up 5-10% of SA.com's work.

In Singapore its 60%+..

Singapore is becoming a hive of mobile application development.

- SG companies are now deploying more iOS applications than web
- Applications developed here are complex, large, and feature rich.

- **My first iOS Application Review in Singapore.**
 - I was nervous - my first week in Singapore.
 - My client was a multi-billion \$ multi-national FSI in Singapore.
 - *“This is Singapore.. These guys are pro when it comes to iPhones. This will be tough, I need to bring my top game..”*
 - Good nights sleep, extra shot latte, good breakfast..
 - Arrived at client - rubbed my hands together - “Yeah bring it on.”

1 hour later:,

**I'm Stealing money
Controlling the system**

**The app was a mess.
Completely broken.**



So I get to keep what I steal right?

- **Clearly we have a problem here...**

- I found a meeting room and sat down with the developers.
- Local Chinese Singaporean dev's, good guys.
- Graduated from NTU.
- 100% iOS developers!



- **Zero clue about security, nothing. 沒有!**

- The application was missing state / session management
- Direct object reference bugs everywhere
- What security controls existed were in the presentation layer.

“Its not that your security is bad, its more like you don't **HAVE** security”

I was shocked: “Really, I was nervous about this?”

- **Contents of this talk:**
 - iOS Applications 101
 - How I test iPhone/iOS Applications
 - Security Model

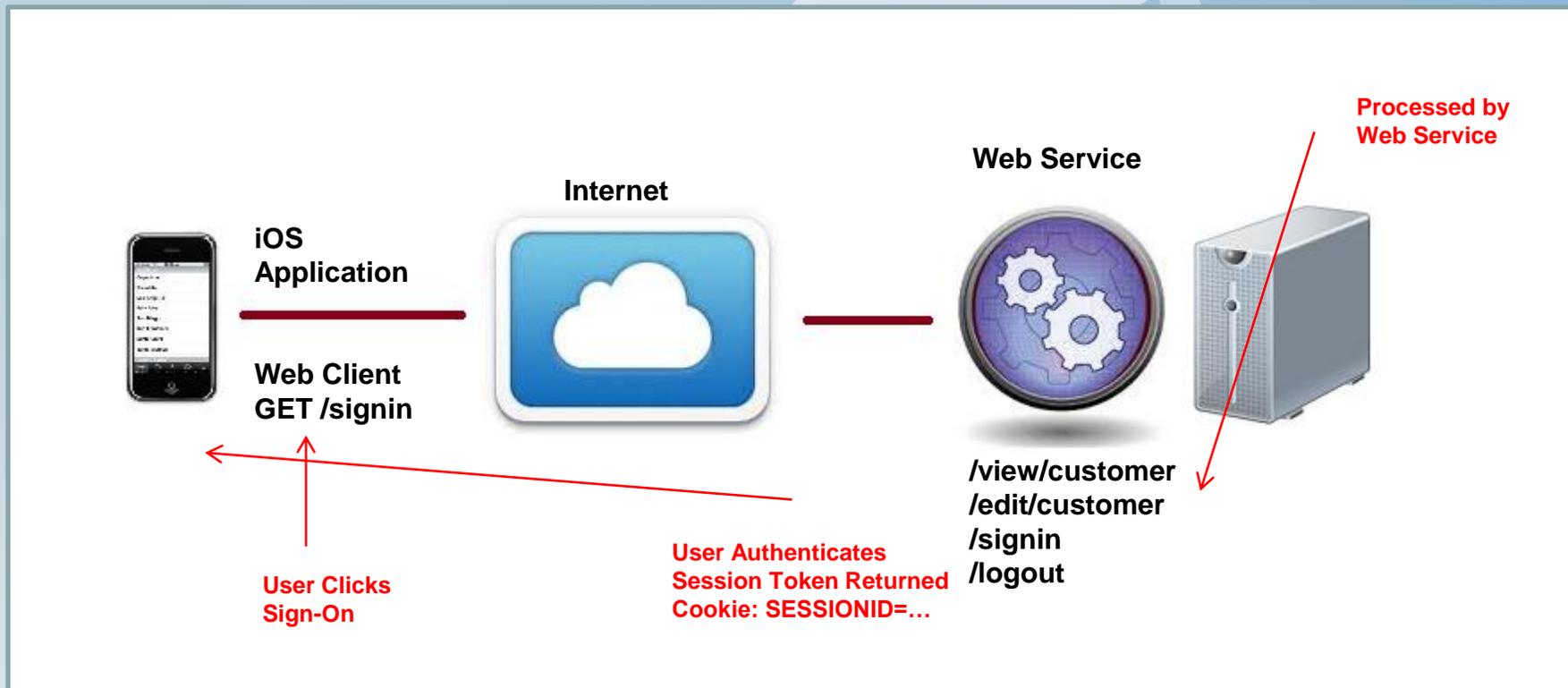
 - Mistakes Developers in Singapore Are Making:
 - The vulnerabilities I find in the largest Singaporean FSI's
“Your banks, your insurance providers, your wealth managers, your Telco's, your utility providers.. Your Singapore, my vulnearbilities.”

 - Moving forwards
 - What to do if your deploying an iOS Application.

■ iOS Applications 101:

- Most of the iOS API & iOS App Development is Objective-C.
- Some of the iOS API involves traditional C rather than Objective-C.
- iOS Applications are typically client / server based.
 - Similar to a thin client or Ajax based web application.
 - iOS Application & Remote Web Server
 - Pull / Push content from a remote server
 - View / manipulate the content
- Communications are performed using light-weight HTTP protocols
 - JSON / REST / XML / HTTP POST
 - Requests are commonly grouped to reduce network overhead
 - Use one request to retrieve all information.

- How a typical application looks:



■ Testing iOS Applications:

- iOS applications are thicker than an traditional AJAX application
 - Content stored on the phone
 - Resources (Images/Scripts/Music/Video)
 - Content repositories (Databases, XML files)
 - Scripts / other binaries

- Thinner than a traditional desktop application.

- Majority of content is stored on a remote server.

- Think of iOS applications as '**Skinny-Fat**' applications:

Urban Dictionary: "When someone is thin and looks great in clothes, but is all flabby underneath"

This unique design model provides a mixture of attack surfaces.

burp suite v1.3.03

burp intruder repeater window help

target proxy spider scanner intruder repeater sequencer decoder comparer options alerts

intercept options history

request to

forward

drop

intercept is on

action

raw params headers hex

GET /m?manifest=1 HTTP/1.1

Host:

User-Agent: Mozilla/5.0 (iPhone; U; CPU iPhone OS 4_2_1 like Mac OS X; en-us) AppleWebKit/53
Safari/6533.18.5

Cache-Control: max-age=0

Accept: */*

Accept-Language: en-us

Accept-Encoding: gzip, deflate

Cookie: MPRF=H4sIAAAAAAAAAAKv4f6Dz5TPVLiaGSUwKSQZmpobGSeaJ5saJcWmGyRamxkbGcqYlmcWaGKRbGJmkTmBlk
NID=44=H8s7q_OMOuHdWWOCmzfZQn2RTVIAyNyIsuVEuSzUMHuqARxZybJzkr1UmCQnH7AHCiPxs08W8DWqwf4Xz-XDd
PREF=ID=472395b539c49ba9:U=ddf753eff534f31f:TM=1298911090:LM=1298911090:S=1XODquhKpvKVBIfm

Connection: keep-alive

Proxy-Connection: keep-alive

- **Flabby Part of iOS Applications**

- Oil and Gas commodity trading company:

“What is the biggest risk from the launch of your iOS application?”

“What do I say when our lead trader or senior exec leaves his iPad in a taxi or bar on a Friday night?”

- **Management will ask me**

- What information was lost ?
 - Can the application still be used ?
 - Are we at risk?



- Very likely, very plausible: *“Its already happened before”*

Apple iPhone

Q: I don't create backups of my iPhone and don't even know how to
care?

Enter Passcode

Pass the Passcode Lock

Passcode Lock

Elcomsoft Phone Password Breaker



Gain access to password-protected BlackBerry, iPhone, iPad and iPod backups! Supporting all versions of BlackBerry and Apple smartphones released up to date, Elcomsoft Phone Password Breaker is the first GPU-accelerated password recovery tool to obtain original passwords protecting iPhone/iPod backups. Supporting multiple ATI and NVIDIA video cards, the new iPhone password recovery tool offers supercomputer performance at consumer prices.

- Do you trust the “Lock”

- **I had an epiphany one night in Singapore.**
 - Zen moment at a coffee shop in Geylang when I realized every problem with every iOS application developed in Singapore.
- **“All my clients are f%^&* it up”**
 - Everything I'm reviewing is breaking.
 - And they all break at the same places..
 - Local banks
 - Foreign banks
- **Common, this is Singapore**
 - How can so many people all be doing it wrong?



- **Developers didn't understand simple concepts.**
 - **Don't use the presentation layer to implement security controls.**
 - “How can you click that, its not enabled, cannot!!”
 - “Can..”
 - **Principal of Least Way Too Much Privilege**
 - Everything has every option enabled, every feature, every privilege.
 - “It works when I enable everything.”
 - **“Why would someone do that?”**
 - Security? What the hell is that, do I need that?
 - New developers, no previous development (security) experience

Bugs, Bugs, Bugs, Bugs
Welcome to my world

Real bugs, real .SG clients.

Names changed to protect the guilty.

These bugs no longer exist
(Thank god)

Why did these bugs exist to begin with?

- **Presentation Layer Security**
 - How many options do you see? Five?

```
GET /customers/view/138 HTTP/1.1
Host: coffeebean-uat.staging
User-Agent: Mozilla/5.0 (iPod; U; CPU iPhone OS 4_3_3
like Mac OS X; en-us) AppleWebKit/533.17.9 (KHTML,
like Gecko) Version/5.0.2 Mobile/8J2 Safari/6533.18.5
Keep-Alive: 300
Proxy-Connection: keep-alive
```

155**158**

A screenshot of a web application showing a list of coffee shop options. The options are: Kopitiam, Starbucks, Super Coffee (with a blue checkmark), Extreme Caffeine, and The Coffee Bean & Tea Leaf. Red arrows point from the numbers 155 and 158 in the adjacent code block to the Starbucks and Extreme Caffeine options respectively.

Change verb to any customer id

“Really? Really??

I'm not even trying here...

Cant you just play a little hard to get?

- **Symmetric vs. Asymmetric Cryptography**
 - Developers like to use cryptography as a method of keeping secrets safe.
 - However more often than not the cryptography is implemented incorrectly.
 - And it's the only security implemented.
 - This one has me beat using the last trick.

Encryption!

```
GET /process?97F5D82E4AD10287EF71B27C28D881FEA HTTP/1.1
Host: wwdev01.sid.org.sg
User-Agent: Mozilla/5.0 (iPod; U; CPU iPhone OS 4_3_3 like
Mac OS X; en-us) AppleWebKit/533.17.9 (KHTML, like Gecko)
Version/5.0.2 Mobile/8J2 Safari/6533.18.5
Keep-Alive: 300
Proxy-Connection: keep-alive
```

Asymmetric and Symmetric Cryptography

Asymmetric Cryptography: Public and Private keys share the same key

```

bufferPtr = malloc( bufferPtrSize * sizeof(uint8_t));
memset((void *)bufferPtr, 0x0, bufferPtrSize);
// memset((void *) iv, 0x0, (size_t) sizeof(iv));

NSString *key = @"123456789012345678901234";
NSString *initVec = @"init Vec";
const void *vkey = (const void *)key;
    
```

Red box highlighting the key string: `@"123456789012345678901234"`

```

__cstring:00093115 a12345678901234 DCB "123456789012345678901234",0
__cstring:00093115
__cstring:00093115
    
```

```

GET /process?97F5D82E4AD10287EF71B27C28D881FEA HTTP/1.1
Host: wwwdev01.sid.org.sg
User-Agent: Mozilla/5.0 (iPod; U; CPU iPhone OS 4_3_3 like
Mac OS X; en-us) AppleWebKit/533.17.9 (KHTML, like Gecko)
Version/5.0.2 Mobile/8J2 Safari/6533.18.5
Keep-Alive: 300
Proxy-Connection: keep-alive
    
```

AES128 Encrypted using CCCrypt and a Preshared key.

Thx for the AES Key..

Encryption Used Foolishly

/process = Process Payment

- GET /process?=F5D82E4AD10287EF7

GET /process?s=5191&t=2&a=100

- GET /process?=16A5CDE830F0638E5

GET /process?s=5190&t=2&a=100.

UserID 5190 just purchased \$100

- GET /process?= EF5A98230FE152E63

GET /process?s=5191&t=2&a=-100

I just purchased -100 worth..

Double Negative = **Positive**.. Account credited

s = UserID

t = Transaction Type

a = Amount Transfer

I'm in your banks, stealing your cash...

谢谢

Paul



- Som

- Auth

- Se
- Re
- Or

- This

- En
- GE

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<users>
<user>
  <username>C0102</username>
  <password>password</password>
  <userid>0</userid>
  <mail>XXXXXXXXXXXXXXXXXXXXXXXXXX</mail>
</user>
<user>
  <username>C1011</username>
  <password>wls3c</password>
  <userid>500</userid>
  <mail>XXXXXXXXXXXXXXXXXXXXXXXXXX</mail>
</user>
</users>
```

“Hey I know how to implement authentication. I’ll just return all of the users to the iPhone! iOS can authenticate the user locally!”

■ Information Stored Insecurely

- Developers love to store information on mobile devices.
- There is a **correct** and an **incorrect** way of doing this.
- Private information should not be stored on the phone, unless protected.

- This includes:
 - SQLite Databases
 - Logging profilers NSLog/Alog
- At best your information is encrypted with an AES key.
 - Credit Card Numbers, Usernames, Passwords, Messages

If your phone gets stolen (or lost), what did you loose?

Key	Type	Value
Root	Dictionary	(15 items)
anyMorePages	Boolean	<input checked="" type="checkbox"/>
anyMorePosts	Boolean	<input checked="" type="checkbox"/>
autosave_clear_preference	Boolean	<input type="checkbox"/>
autosave_enabled_preference	Boolean	<input checked="" type="checkbox"/>
CurrentBlogIndex	Number	0
pages-iostesting.wordpress.com	Array	(1 item)
refreshCommentsRequired	Boolean	<input type="checkbox"/>
refreshPagesRequired	Boolean	<input type="checkbox"/>
refreshPostsRequired	Boolean	<input type="checkbox"/>
statsDate	Date	Jan 4, 2011 2:38:51 PM
statuses-iostesting.wordpress.com	Dictionary	(4 items)
wpcom_authenticated_flag	String	1
wpcom_password_preference	String	princess123
wpcom_username_preference	String	iostesting

ues

Value
Value

■ Key Chains

- “The keychain is where an iPhone application can safely store data that will be preserved across a re-installation.”
- Keychains are backed up whenever the user backs up the device via iTunes.
- **Individual application key chains are secured with attributes.**

```

CFTyperef kSecAttrAccessibleWhenUnlocked;
CFTyperef kSecAttrAccessibleAfterFirstUnlock;
CFTyperef kSecAttrAccessibleAlways; ←
CFTyperef kSecAttrAccessibleWhenUnlockedThisDeviceOnly;
CFTyperef kSecAttrAccessibleAfterFirstUnlockThisDeviceOnly;
CFTyperef kSecAttrAccessibleAlwaysThisDeviceOnly;
    
```

Most permissive option

The option most developers pick.

GMail as MS Exchange	Password	Always
iChat.VeniceRegistrationAgent	Token	Always-ThisDeviceOnly
Identity Certificate (e.g. for VPN)	Certificate	Always-ThisDeviceOnly
LDAP	Password	Always
MCEmail Account (probably created by iPCU profile)	plist with IMAP password	Always
MS Exchange	Password	Always
Visual Voicemail	Password	Always
VPN IPsec Shared Secret	Password	Always
VPN XAuth Password	Password	Always
VPN PPP Password	Password	Always

- 4: Gain access to any of the following keys: kSecAttrAccessibleWhenUnlocked or kSecAttrAccessibleAfterFirstUnlock.

- **“Shit, the developers should have turned that off/on”**

- SSL settings are usually disabled when an application is in development.

```
setValidatesSecureCertificate
kCFStreamSSLAllowsExpiredCertificate
kCFStreamSSLAllowsExpiredRoots
kCFStreamSSLAllowsAnyRoot
kCFStreamSSLValidatesCertificateChain
validatesSecureCertificate = NO;
```

- These settings are usually turned back on before an app is launched.

“Usually...”

I do love that word.



“Did you turn SSL validation back on?
Because I sure as fuck didn’t.”

■ Disassemble your way to victory.

```

_cstring:0009450C aUPO DCB "u=%E&p=%E&o=%E",0 ; DATA XREF: __cfstring:cfstr_UPO↓
_cstring:0009451B aUserdefaultuse DCB "UserDefaultUserID",0 ; DATA XREF: __cfstring:cfstr_Userdefaultuse↓
_cstring:0009451B ; DATA XREF: __cfstring:cfstr_Userdefaultuse↓
_cstring:0009452D aHttpsUat_pbe_1 DCB "https://uat. [redacted] /chgpwd.aspx" ; DATA XREF: __cfstring:cfstr_HttpsUat_pbe_1↓
_cstring:0009452D ; DATA XREF: __cfstring:cfstr_HttpsUat_pbe_1↓
_cstring:0009452D DCB 0
_cstring:0009456E aChangepwdnotif DCB "ChangePWDNotification",0 ; DATA XREF: __cfstring:cfstr_Changepwdnotif↓
_cstring:0009456E ; DATA XREF: __cfstring:cfstr_Changepwdnotif↓
_cstring:00094584 aNewpasswordr_1 DCB "newPasswordResponse:",0 ; DATA XREF: __cfstring:cfstr_Newpasswordr_1↓
_cstring:00094584 ; DATA XREF: __cfstring:cfstr_Newpasswordr_1↓
_cstring:00094599 aTIdIdCpflag DCB "<TD ID=",0x22,"CPFLAG",0x22,">",0 ; DATA XREF: __cfstring:cfstr_TdIdCpflag↓
_cstring:00094599 ; DATA XREF: __cfstring:cfstr_TdIdCpflag↓
_cstring:000945AA aTIdIdCpmsg DCB "<TD ID=",0x22,"CPMSG",0x22,">",0 ; DATA XREF: __cfstring:cfstr_TdIdCpmsg↓
_cstring:000945AA ; DATA XREF: __cfstring:cfstr_TdIdCpmsg↓
_cstring:000945BA aStockquotelist DCB "%E?stockquotelist&msg=2&group=%d",0 ; DATA XREF: __cfstring:cfstr_Stockquotelist↓
_cstring:000945BA ; DATA XREF: __cfstring:cfstr_Stockquotelist↓
_cstring:000945DB aTop10notificat DCB "Top10Notification",0 ; DATA XREF: __cfstring:cfstr_Top10notificat↓
_cstring:000945DB ; DATA XREF: __cfstring:cfstr_Top10notificat↓
_cstring:000945ED aTop10respons_1 DCB "top10Response:",0 ; DATA XREF: __cfstring:cfstr_Top10respons_1↓
_cstring:000945ED ; DATA XREF: __cfstring:cfstr_Top10respons_1↓
_cstring:000945FC aSinglestockinf DCB "%E?singlestockinfo&msg=%d&group=%d&code=%E&width=263&height=185",0 ; DATA XREF: __cfstring:cfstr_Singlestockinf↓
_cstring:000945FC ; DATA XREF: __cfstring:cfstr_Singlestockinf↓
_cstring:0009463C aSinglestocki_2 DCB "%E?singlestockinfo&msg=%d&group=%d&code=%E",0 ; DATA XREF: __cfstring:cfstr_Singlestocki_2↓
_cstring:0009463C ; DATA XREF: __cfstring:cfstr_Singlestocki_2↓

```

- Recent PT of a banking iOS app.
- Production was strong – zero findings. 😞
- Strings / references to UAT were present in the production application
 - UAT environment was internet accessible!
 - Different code base (and insecure as hell!!)

- What about SQL Injection ?

- SQL Injection has been the bane of developers for years..
- iOS Supports SQLite

load_extension(X,Y) The load_extension function loads the shared library file named X. The return value is always a NULL. If Y is not NULL, the load_extension function is used. The load_extension function is used to load or unload a shared library.

ATTACH DATABASE

attach-stmt:

```

const char *sql,
int (*callback) (void*,int,char**,char**),
void *,
char **errmsg
);
/* An open database */
/* SQL to be evaluated */
/* Callback function */
/* 1st argument to callback */
/* Error msg written here */

```

Incorrect way

- What can we do with SQL Injection on iOS?

- **Pragmatic Errors – Oh, so many ways to get it wrong..**

- System Timers

- The default iOS timer group (currentRunLoop) will pause if you click on a menu
 - Client side work right.

- NSDataDetector

- You can use phone numbers, converted

UIDataDetectorTypes

Defines the types of information that can be detected in text-based content.

```
enum {
    UIDataDetectorTypePhoneNumber
    UIDataDetectorTypeLink
    UIDataDetectorTypeAddress
    UIDataDetectorTypeCalendarEvent
    UIDataDetectorTypeNone
    UIDataDetectorTypeAll
};
typedef NSUInteger UIDataDetectorTypes;
```

Yeah you guessed it, everyone uses this one...



■ URI Handlers

- *“One of the coolest features of the iPhone SDK is an application’s ability to “bind” itself to a custom URL scheme and for that scheme to be used to launch itself from either a browser or from another application on the iPhone. Creating this kind of binding is so simple, its almost criminal not to use it in your application!”*
- Most of the time developers use this feature sparingly
 - ` Call me now`
- And some what not sparingly.
 - `app://transfer/9999/032-01233311/022-0479890`

Clickable on any page which has a Detector set to parse links (or Everything)

■ NULL Bytes

- iOS is affected by NULL byte attacks “Poison NULL Byte”
- User supplied values passed to POSIX-C functions by objective-c libraries.
- Objective-C does not NULL terminate
- POSIX C does

String\x00string

becomes

String

- String termination bugs.

Click



■ Help, Help, Help

- Everyone seems to need help when deploying an iOS application.
- Clients simply ask me “What should I be doing?.”

“iOS Security is a hybrid of traditional desktop application security and web security.”

■ My advice..

- Treat iOS applications with importance now, because they are only going to grow in functionality and demand.
- If your outsourcing your application development provide Security Guidelines to the external developers.
- Avoid using multiple outsourcing partners for iOS + Backend development.
- Involve security testing/advice early in the project – 20% 80% 100%.

- **How to make it easier for me (and you)**
 - Strangely enough clients never know what to do when I come to review an iOS application.

- **Things you should do**
 - Get it tested.
 - Provide source code to the testers.
 - If you cant provide source code - provide documentation.
 - Provide the binary before it goes to the App Store.
 - Make sure the test is performed using a Jail Broken device.

■ Conclusion:

- Singapore is the iPhone hub of the world
- iOS is however still considered 'bleeding-edge'
- Hobbyist come commercial developers

- Apple have tried to make a secure SDK
- Developers **still** manage screw it up.
- This is only going to get worse as both the iPhone and iOS Apps increase in complexity and functionality.

- iOS development should be part of your standard Application Development Life Cycle.

- **Questions ? Comments**

- Paul Craig

- Paul.Craig@security-assessment.com

黑客?

Think you can hack? Got talent?

We are hiring!