Security in
knowledge

APPLICATION SECURITY: FROM WEB TO MOBILE. DIFFERENT VECTORS AND NEW ATTACK

John T Lounsbury
Vice President Professional Services, Asia Pacific
INTEGRALIS

Session ID: MBS-W01

Session Classification: Advanced

Mobile Applications - How Many do **you** have?

► With the ubiquitous usage of smart phones and PDAs the Web Application market has exploded

► What started primarily as games and productivity aids have developed into business and corporate tools

  ► Evolving from static content to highly interactive user interfaces exchanging data and information of all types

  ► The line between the personal world and the business world is almost non-existent

► A lack of development standards, frameworks, and language have inadvertently opened up a vast, new targeted threat landscape

► These threats, and how to deal with them, represent one of today's single largest information Security risks

INTEGRALIS
an NTT Communications Group Company

Some Interesting Statistics

► Android holds the current market share (around **70%**)

► There are other players:

  ► Apple – **21%**

  ► BlackBerry – **3%**

  ► Windows – **3%**

  ► Linux – **2%**

  ► Other – **1%**

► **73%** of organizations have been hacked at least once in the past two years through insecure web applications

► On average, every web application has an average of **12** vulnerabilities

How much security can there be in a $0.99 application?

► Developers are primarily focused on development efficiency, it is rare that development organizations consider the security implications of their application

► Secure Development Life Cycle practices and implementations are still not widely accepted and implemented – especially in the smaller shops that where many mobile applications are developed

INTEGRALIS
an NTT Communications Group Company

Differences with Web and Mobile Applications

► A large number of web application security vulnerabilities are generally associated with a lack of input validation (SQL Injection, XSS, Open Redirects, Remote File Includes, etc)

► With web applications, attackers generally attack from a pure black-box methodology.

► In general, greater skill and knowledge around reverse engineering is required for assessing and attacking mobile applications because the code is already in the hands of the attackers

Mobile Application Threat Model
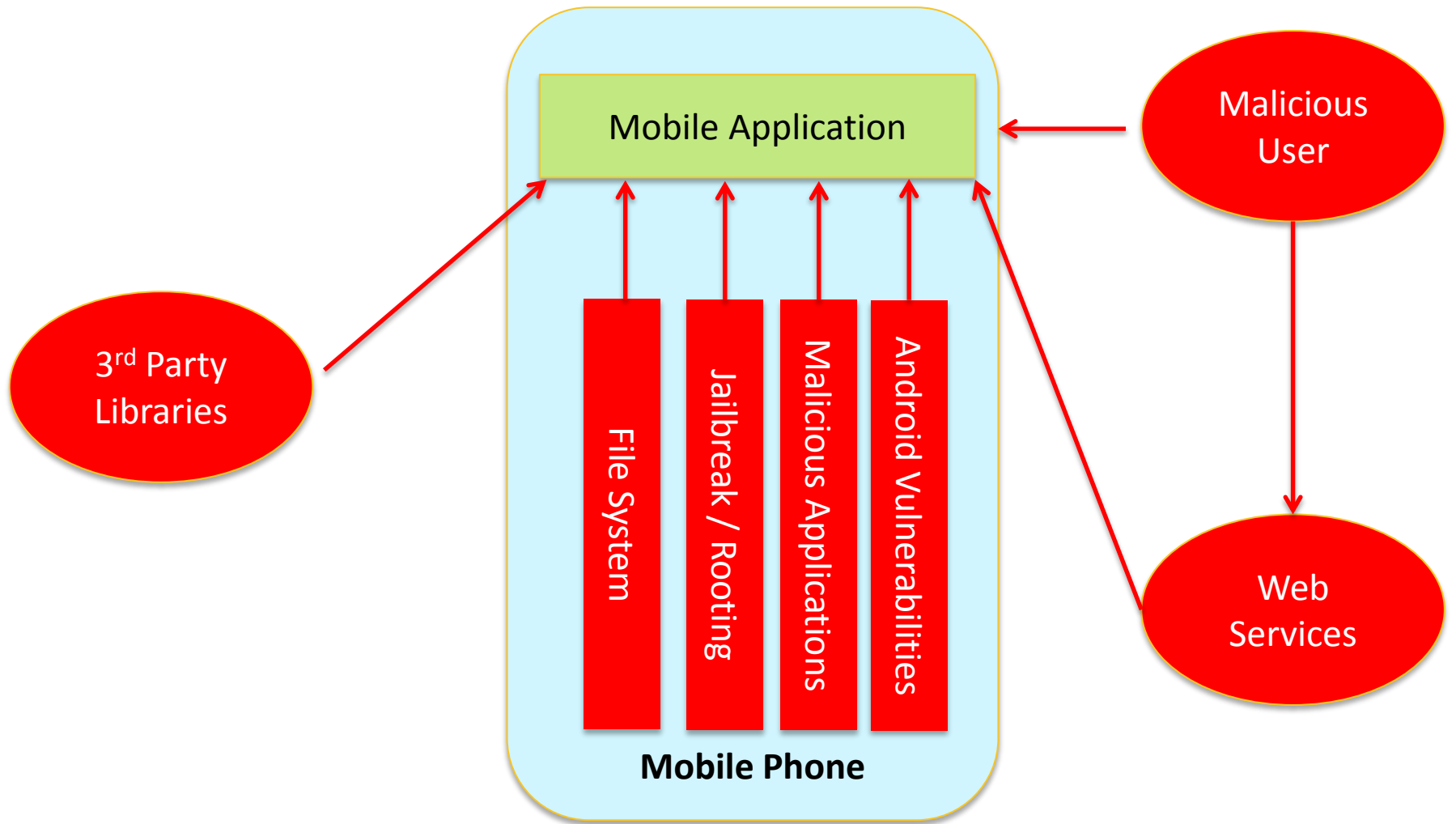
Mobile Application Attack Surfaces

► The Application and Code

    ► Identification and manipulation of client side logic

► Permissions

    ► Are application components sufficiently protected

► Data-in-Transit

    ► Is the data protected as it's sent to the server

► Data-at-Rest

    ► Is the data stored on the device secured

► The Server

    ► Does the server validate data from the mobile application

The Application and Code

► Both Android and iOS applications can be extracted from the phone in their binary format

   ► Android: Java compiled to DEX VM bytecode

   ► iOS: Obj-C compiled to ARM binaries

► A mixture of techniques is then used to decompose the application:

   ► Analysis of the AndroidManifest.xml File

   ► Static Analysis

   ► File System Analysis

   ► Dynamic Analysis

   ► Reverse Engineering

The Application and Code
It's in the attacker's hands

► Compilation and de-obfuscation introduce a degree of difficulty for an attacker attempting to retrieve the code in its original form, however it's possible for the determined and capable hacker

► Dynamic analysis allows for an application to be extracted from a mobile device and executed within an emulator

► Execution within an emulator can highlight exactly what API calls are being executed, and which files on the underlying file system are being accessed

The Application and Code
Getting back to the Source Code (Android)
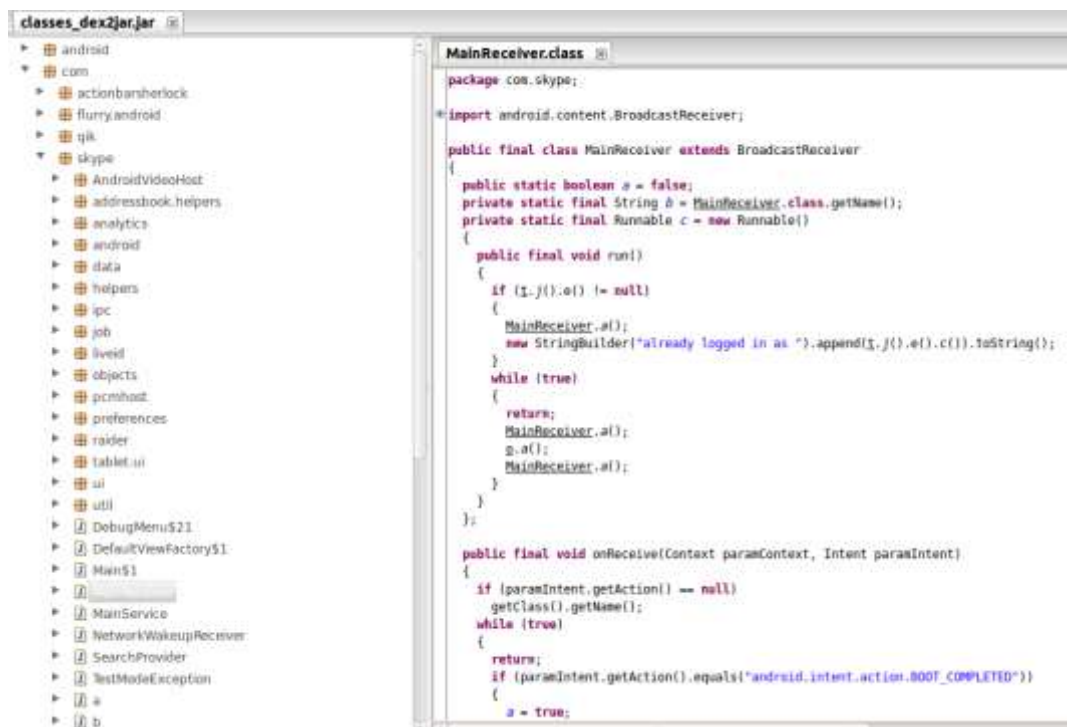
```
[neil@europa skype]$  unzip com.skype.raider-2.apk
Archive:  com.skype.raider-2.apk
  inflating: assets/raider-2.0-market-live.cert
 extracting: assets/video.cfg
  inflating: res/anim/disappear.xml
  inflating: res/anim/fade.xml
  inflating: res/anim/fade_in.xml
```

```
[neil@europa skype]$  ls
total 20272
-rw-rw-r--  1 neil neil  2289128 Feb 21 18:36 resources.arsc
-rw-rw-r--  1 neil neil    18464 Feb 21 18:36 AndroidManifest.xml
-rw-rw-r--  1 neil neil  2965468 Feb 21 18:36 classes.dex
-rw-r--r--  1 neil neil 15453186 May 28 18:31 com.skype.raider-2.apk
drwxrwxr-x  3 dev  dev      4096 May 28 18:32 ..
drwxrwxr-x  2 neil neil     4096 May 28 19:08 assets
drwxrwxr-x 40 neil neil     4096 May 28 19:08 res
drwxrwxr-x  3 neil neil     4096 May 28 19:08 com
drwxrwxr-x  3 neil neil     4096 May 28 19:08 lib
drwxrwxr-x  7 neil neil     4096 May 28 19:08 .
drwxrwxr-x  2 neil neil     4096 May 28 19:08 META-INF
```

► classes.dex contains the Java bytecode in Dalvik compatible form

► Continue to decompile it back into Java source code

INTEGRALIS
an NTT Communications Group Company

The Application and Code
Getting back to the Source Code (Android)

► Most applications are compiled with obfuscation which makes things more difficult (like this example) and quite tedious

► It boils down to how much time and diligence the attacker has

The Application and Code
Getting back to the Source Code (Android)

► Other applications haven't been obfuscated at the compilation stage
► The source code in this case is very close to the original

The Application and Code
Getting back to the Source Code (Android)

► Developers who don't obfuscate their code prior to release are helping the bad guys out

► Simply by reading through the source code you can determine:

  ► How the application communicates to the server
  ► The types of requests sent to remote servers and their format
  ► If the application interacts with other components on the phone
  ► If the application is writing files to the underlying operating system
  ► Cryptographic Functions
  ► Third party libraries in use

► All this helps in identifying potential attack surfaces of the application

► One of the best ways to attack a client-server application is to write your own client to communicate to the server

The Application and Code
Getting back to the Source Code (Android)

► The RequestAuthorization class shows exactly how the authentication token is created (along with the secret key)

```java
public class RequestAuthorization
{
  private static String PATTERN = "000111011111011100011110101011110110100010011100110001100001000110";
  private static String SECRET = "iEk21fuwZApXlz93750dmW22pw389dPwOk";
  private static String STATIC_TOKEN = "m198sOkJEn37DjqZ32lpRu76xmw288xSQ9";

  public static String createRequestAuthorizationToken(String paramString1, String paramString2)
    throws NoSuchAlgorithmException, UnsupportedEncodingException
  {
    String str1 = SECRET + paramString1;
    String str2 = paramString2 + SECRET;
    MessageDigest localMessageDigest = MessageDigest.getInstance("SHA-256");
    localMessageDigest.update(str1.getBytes("UTF-8"));
    String str3 = new String(toHex(localMessageDigest.digest()));
    localMessageDigest.update(str2.getBytes("UTF-8"));
    String str4 = new String(toHex(localMessageDigest.digest()));
    String str5 = "";
    int i = 0;
    if (i >= PATTERN.length())
      return str5;
    int j = PATTERN.charAt(i);
    StringBuilder localStringBuilder = new StringBuilder(String.valueOf(str5));
    if (j == 48);
    for (char c = str3.charAt(i); ; c = str4.charAt(i))
    {
      str5 = c;
      i++;
      break;
    }
  }
}
```

The Mobile Application
Logical Flaws

► We looked at a popular game which requires purchases of "gold" to progress further – it could probably get quite expensive

► Upon decompiling the application and navigating through the obfuscated code, we were drawn to this call to a file on the device itself:

► The gold value is stored in this preferences file on the device

```
sp = getSharedPreferences("account_info", MODE_PRIVATE);
gold_amount = sp.getInt("gold", -1);
```

► A simple edit and you're a millionaire within the game

  ► We had 2722 gold but now we have 1.00M !

► The server doesn't keep track of this value!

  ► Not stored on the server

  ► Not validates

The Mobile Application

Logical Flaws

► Mobile application research, shows that developers are implementing business logic into their applications

► Developers are often under the impression that if their code is obfuscated, it cannot be decompiled

   ► Thus they decide to implement logic into the applications which should be done on the server side

► An application that hasn't been obfuscated means that logic flaws can usually be identified quickly!

Permissions

- ► Mobile applications are installed with different privilege levels
- ► A user's mobile device will usually have applications from unknown developers alongside applications they trust for everyday tasks
  - ► For example, their mobile banking application and free games
- ► Under the Android security model every application runs in its own process and using a low-privileged user ID
- ► Applications can only access their own files by default
- ► With this isolation in place, applications are able to communicate via different components
- ► ***These communications between components are a critical area of focus for assessing the security of a mobile application***

Permissions

► Android Applications are typically made up of multiple components:
  ► Activities
  ► Services
  ► Content Providers
  ► Broadcast Receivers

► Permissions are granted to each component for as long as the application is installed on the device

► If these components are not properly secured, malicious or other rogue programs can interact with them

► ***Most applications that we have looked at recently do not have well defined permissions***

Permissions

What did we find?

► Poorly protected components for a number of well known and popular applications

- ► Poorly protected activities (GUI screens) can be launched instead of the intended Activity
- ► Exported Activities can be launched by other applications (think CSRF for mobile applications)
- ► Broadcast messages vulnerable to eavesdropping or DoS (Denial of Service)

► ***Our evaluation showed that most Android applications do not adequately place permissions around their components***

► ***Some applications were found to contain the 'debug' flag. Even though this is removed by default within the Eclipse ID, it still managed to end up in the production application!***

INTEGRALIS
an NTT Communications Group Company

Data-In-Transit

► Most applications will at some point communicate with a server endpoint

► Confidentiality and strong authentication is a must

► Authentication:
  ► Verify the entity that the application is communicating with

► Confidentiality:
  ► Prevent snooping
  ► Man-in-the-middle attacks

Data-At-Rest

► One of the largest challenges with mobile applications is how they secure data on the device

► Many popular applications use SQLite database files which are not adequately protected

► If the device is lost or stolen, it would be trivial for an attacker to pull these databases and view the records

► ***On both Android, and iOS it's fairly easy to get root access and access application data and configuration files***

    ► ***These files contain usernames, passwords, encryption keys, and other sensitive account information***

The Server

History Repeats Itself

► Through our analysis of popular mobile applications, we continued to find old web application vulnerabilities:
  ► Username Enumeration
  ► SQL Injection
  ► Broken session management
  ► Information Disclosure
► Does the server perform strong validation on these requests
► Does the server allow the client to make logic decisions (this should be a big NO-NO, but happens all the time)
► Does the server validate the mobile application
  ► A common tactic is to reverse engineer the mobile application and write a rogue client that communicates with the server
  ► Are clients validated

The Server

Harvesting User Information

► The mobile application communicates to the server over HTTPS (good) and using JSON requests. It looks something like this:

```
POST https://XXXXXXXXXXXXXX/data/user_settings
user=test_user2013
timestamp=9327680311866
request_tok=xjs7akl77wlslxjd882xkxz
```

► The server would then respond to this request, with the user's account information

```
{
    "phone": "+852XXXXXXXX",
    "user": "test_user2013",
    "action": "test_user2013@hotmail.com",
    "logged": true
}
```

► What happens if we change the username parameter… ?

INTEGRALIS
an NTT Communications Group Company

The Server

Harvesting User Information

► The server responds with the user information for that user

► This isn't good. There's obviously an authentication problem here

HTTP Request

```
POST https://XXXXXXXXXXXXXXXXXX/data/user_settings
user=test_user_2_2013
timestamp=9311680313861
request_tok=xjs7akl77wlslxjd882xkxz
```

HTTP Response

```
{
    "phone": "+852XXXXXXXXX",
    "user": "test_user_2_2013",
    "action": "test_user_2_2013@hotmail.com",
    "logged": true
}
```

INTEGRALIS
an NTT Communications Group Company

The Server

Harvesting User Information

► This is a serious vulnerability but unfortunately it happens quite a lot

► All an attacker needs to have is a valid username and by exploiting the vulnerability, he is able to obtain:

  ► The user's mobile phone number
  ► The user's email address

► Through social engineering he may further his attacks against the user

► He could also write a script to enumerate valid users within the system and obtain their information

► This application has been downloaded millions of times!

Mobile Application Study

► For the purpose of this presentation, we undertook a security study into 40 mobile applications

► These applications consisted of:
  ► Games
  ► Banking Applications
  ► Communications/Messaging
  ► Entertainment

► The purpose was to identify common vulnerabilities associated with mobile application development

INTEGRALIS
an NTT Communications Group Company

Mobile Application Assessment Results
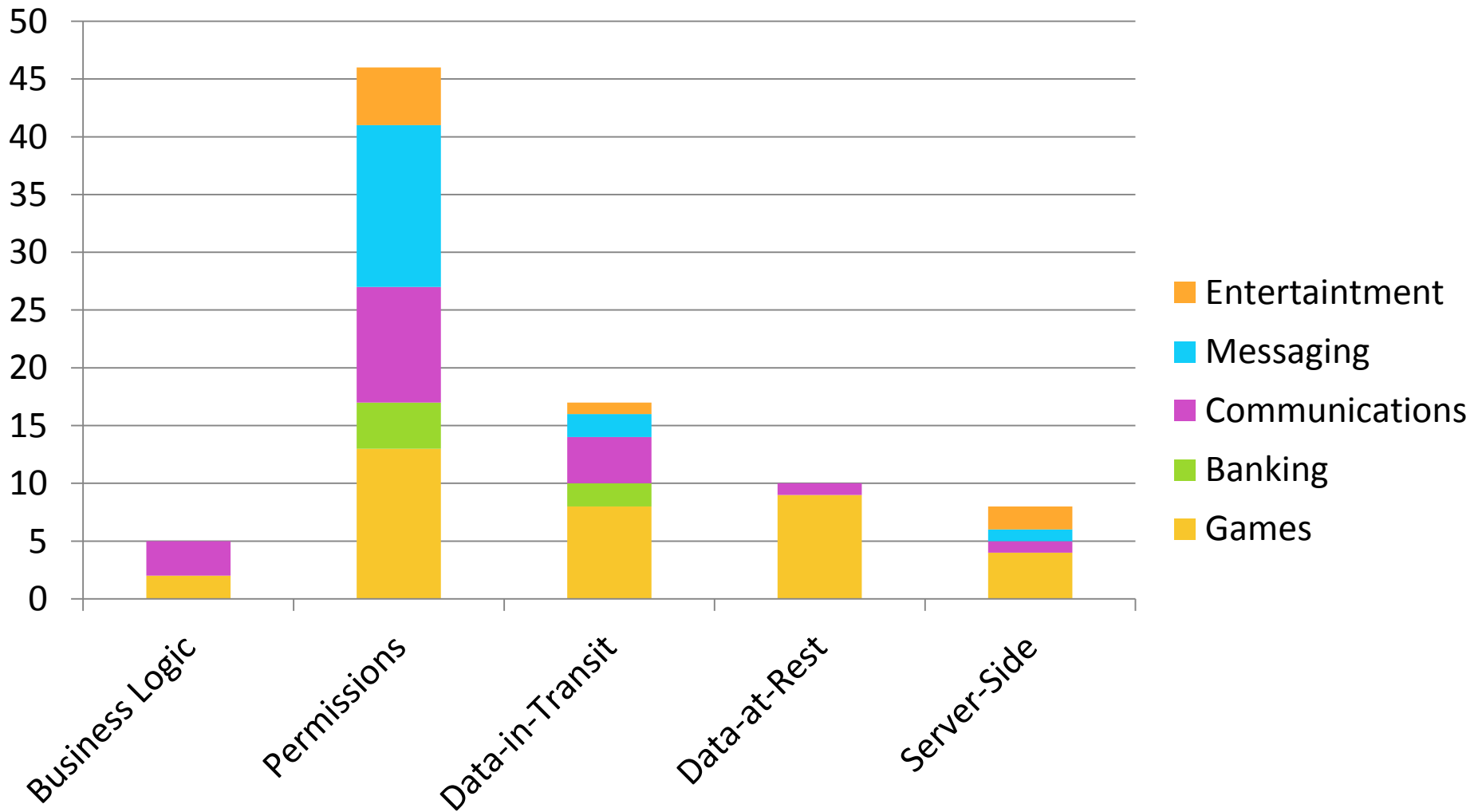
Summary

► The threat surface of mobile applications differs significantly rom that of web applications

► Web application security has improved, but 5 year old vulnerabilities are still common and the introduction of web application-to-mobile application dynamics has both exacerbated the existing vulnerabilities and introduced new ones

► Development of mobile application development guidelines and assessment capabilities are needed

► The risk of liability to corporations could be potentially very serious

► Organizations should have a framework for regular assessment of i-house developed mobile applications and a solution for assuring the security of applications on BYOD

RSACONFERENCE
ASIA PACIFIC **2013**

Questions and Answers

Security around Mobile Applications

- ► There isn't a lot out there right now in terms of best practice and guidelines
- ► Most developers are quite new when it comes to mobile application development

- ► There are a lot of good books and resources on how to break and find flaws in mobile applications

- ► But there's not a lot out there to help developers understand and mitigate against security flaws

- ► Malware targeting the Android platform is growing rapidly

- ► Will the mobile application arena be a repeat of what was experienced with web application security?