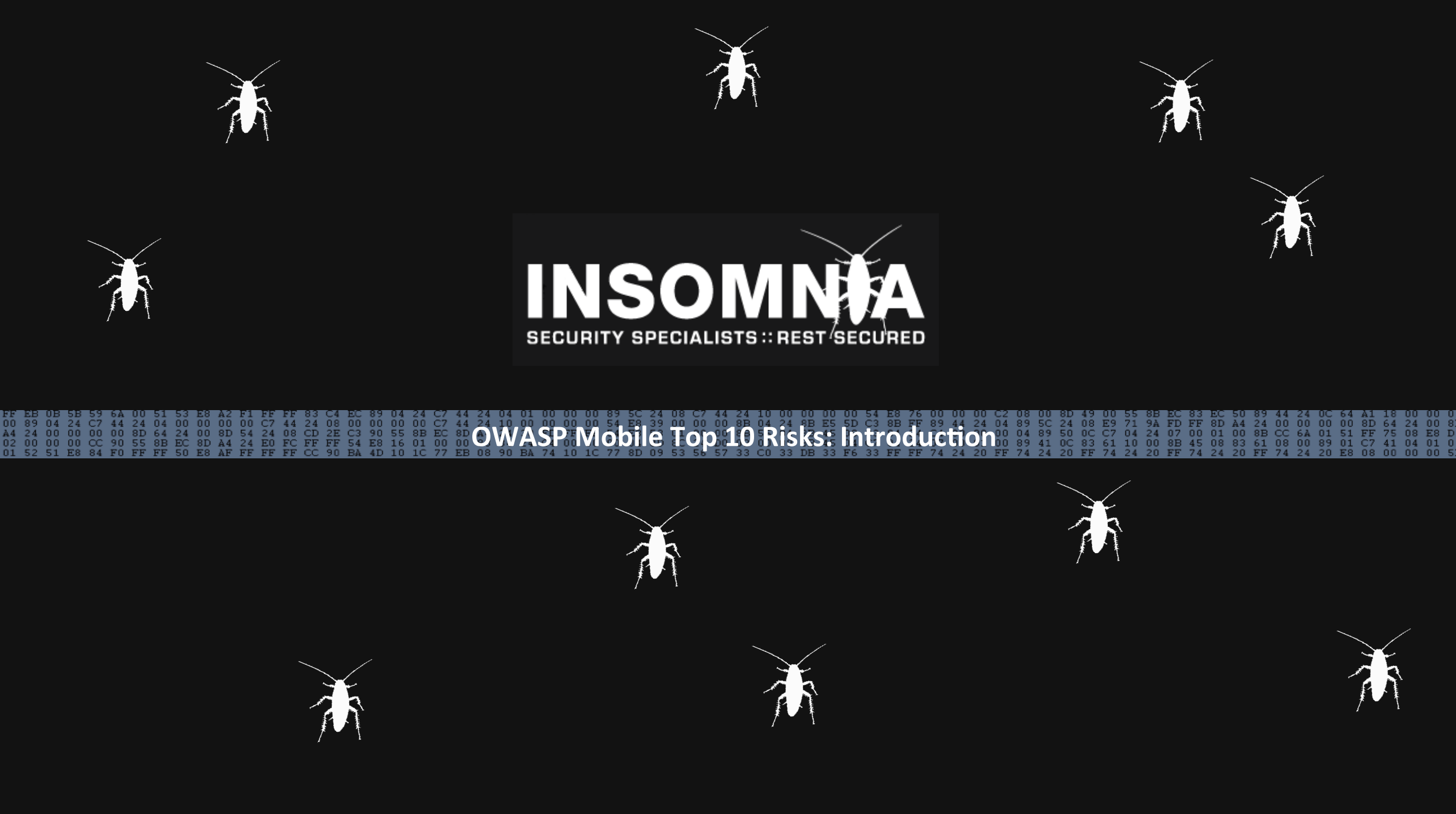
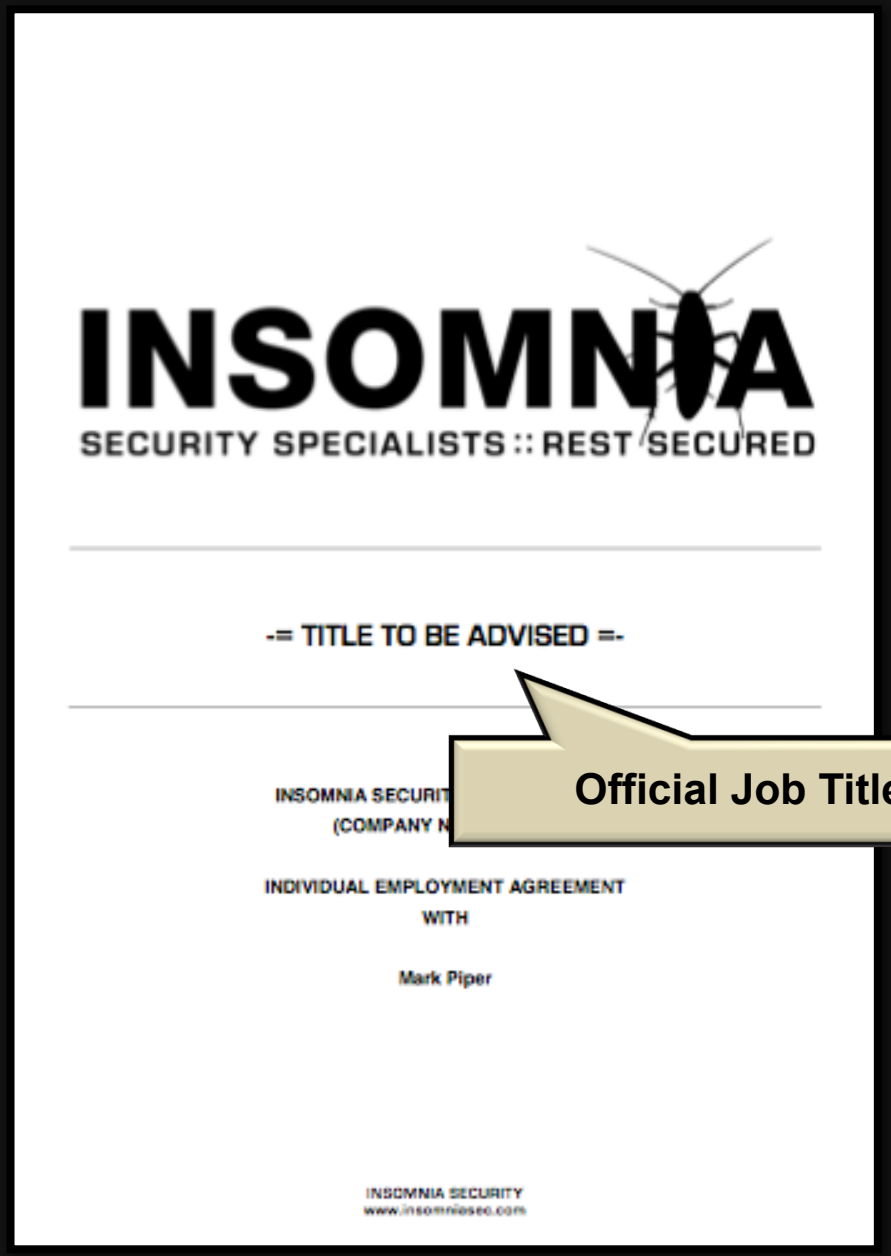


OWASP Mobile Top 10 Risks: Introduction



Official Job Title

Credit

This talk introduces the Top 10 work primarily by:

- Jack Mannino
- Mike Zusman
- Zach Lanier
- OWASP Data Submitters & Manglers

Mobile Application Summary

Deployment in a “hostile” environment

There is no “Security via Obsecurity”

Similar behaviour to a traditional in-browser applications

Similar to a thick client

Everything can be reversed

Three primary threats to consider

- Threat to user data
- Threat to device integrity
- Threat to end-point services

OWASP Mobile Top 10 Risk Project

Initially released in 2011

Attempt to understand mobile threats

Primary focus is on applications

Device security is “considered”

Server side endpoints are in scope

Is designed to be device / platform agnostic

M1: Insecure Data Storage

Common storage facilities

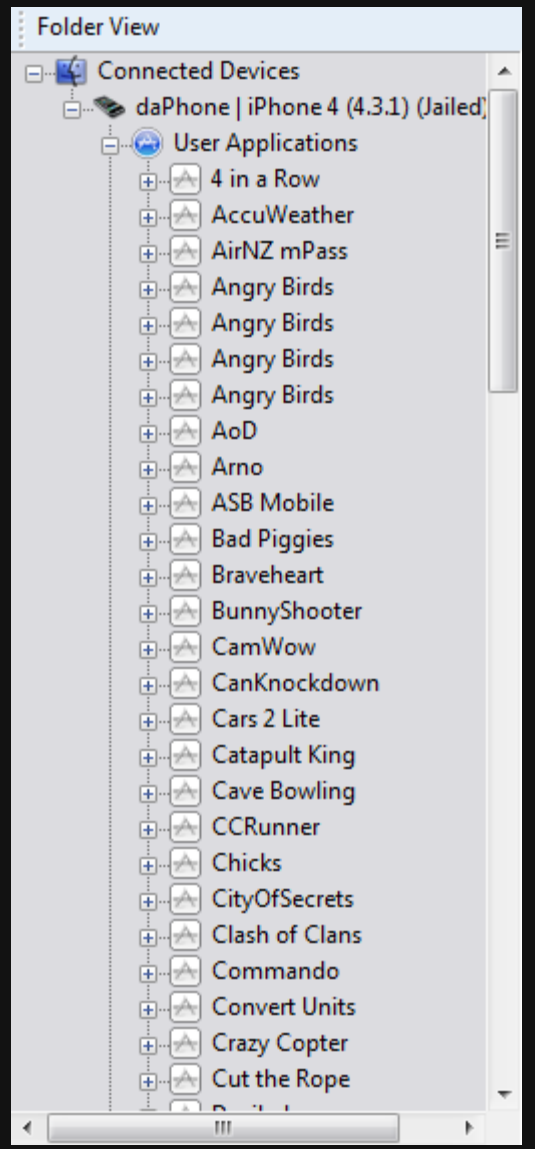
- Plist files
- SQLite DB's
- Text Files

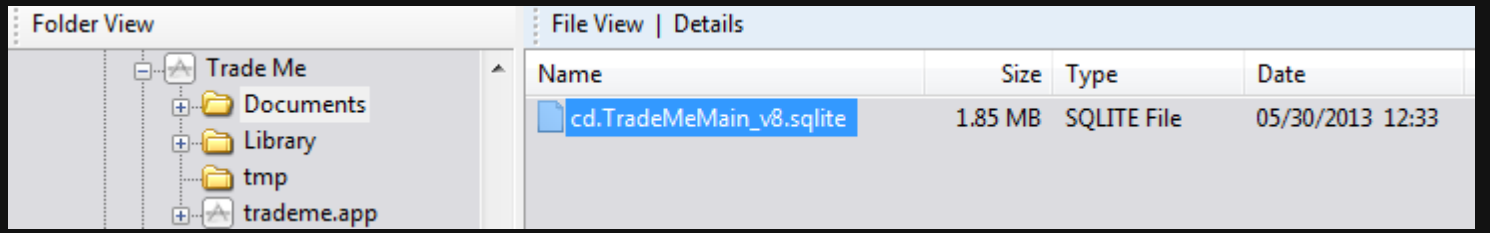
Items of Interest

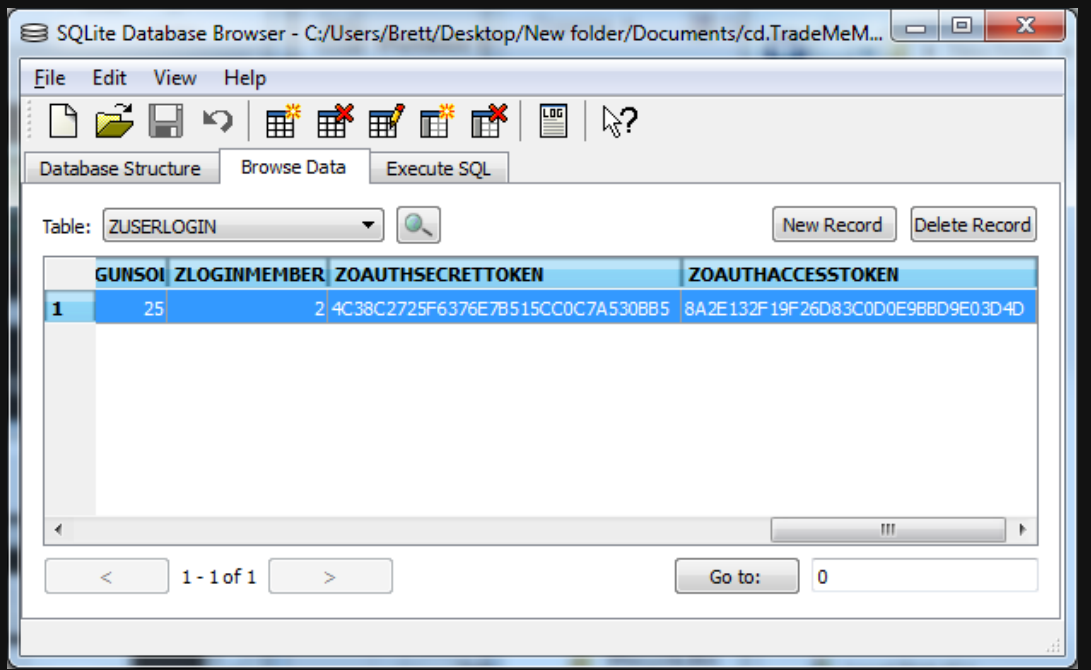
- Credentials
- Authentication tokens
- Unique identifiers
- CC / PII data

Not to be mistaken with M10 (Hardcoded)

M1: Insecure Data Storage (Example)







M2: Weak Server Side Controls

All the backend services

Application consumption generally over HTTP

Many of the common OWASP web issues apply:

- SQL injection
- XML / XXE issues
- Cross site scripting
- Poor authentication
- Poor authorisation

M3: Insufficient Transport Layer Protection

SSL enforcement
SSL enforcement consistency
Certificate & CA management

Things of interest:

- Self-signed certificates
- Appropriate length ciphers
- “Wildcard” certificate trust
- In-application certificate acceptance

M3: Insufficient Transport Layer Protection (Example)


```
$ curl --head "https://m.facebook.com/dialog/oauth?
type=user_agent&display=touch&redirect_uri=fbconnect%3A%2F
%2Fsuccess&sdk=ios&scope=&client_id=111111111111111"

HTTP/1.1 302 Found
Cache-Control: private, no-cache, no-store, must-revalidate
Content-Type: application/xhtml+xml; charset=utf-8
Expires: Sat, 01 Jan 2000 00:00:00 GMT
Location: http://m.facebook.com/login.php?
skip_api_login=1&api_key=111111111111111&signed_next=1&next=https%3A%2F
%2Fm.facebook.com%2Fdialog%2Foauth%<snip>
Pragma: no-cache
```

M4: Client Side Injection

Not all applications are “native”

Hybrid Applications

- Bundled HTML + JSON
- Wrapped HTTP pages

See also: M7 Security Decisions Via Untrusted Inputs

XSS / CSRF twists:

- Cross-application communications
- SMS sending
- Phone dialing
- In-application payment process

SQL Injection twists:

- SQLite **IS** a database.

M4: Client Side Injection (Example)


```
sqlite3 *database;
sqlite3_stmt    *statement;
if(sqlite3_open([databasePath UTF8String], &database) == SQLITE_OK)
{
    NSString *sql = [NSString stringWithFormat:@"INSERT INTO messages
VALUES('1','%@','%@','%@')", msg, user, displayname];
    const char *insert_stmt = [sql UTF8String];
    sqlite3_prepare_v2(database, insert_stmt, -1, &statement, NULL);
    if (sqlite3_step(statement) == SQLITE_DONE)
```

M5: Poor Authorisation and Authentication

Primarily an architecture issue

Security controls based on wrong assumptions

Many “unique” values may be compromised:

- IMSI
- IMEI
- UUID

Some identifiers may persist across hardware resets

M5: Poor Authorisation and Authentication (Example)

INSOMNIA SECURITY SPECIALISTS :: REST SECURED

```
ANDROID_ID: 9774d56d682e549c


Commit: 0fe27cf5bd1407bc7b4eabefaa91ff535582badc
Author: Doug Zongker dougz@android.com (Thu Aug 19 13:38:26 2010 -0700)
Committer: Doug Zongker dougz@android.com (Thu Aug 19 13:38:26 2010 -0700)
Tree: c37a29d2893c5554325b53ad0ed1da564ecc8183
Parent: 46906276448dd36e7a5cca38fbe9fdb3142f7948[diff]

"make android_id random seed depend on time as well as ro.serialno"
```

M6: Improper Session Handling

Mobile sessions persist over long periods of time

Revocation capability lacking

Classic session issues:

- Poor token generation
- Not appropriately expired server-side
- Session fixation attacks

M7: Security Decisions Via Untrusted Inputs

Commonly abuse of application “features”

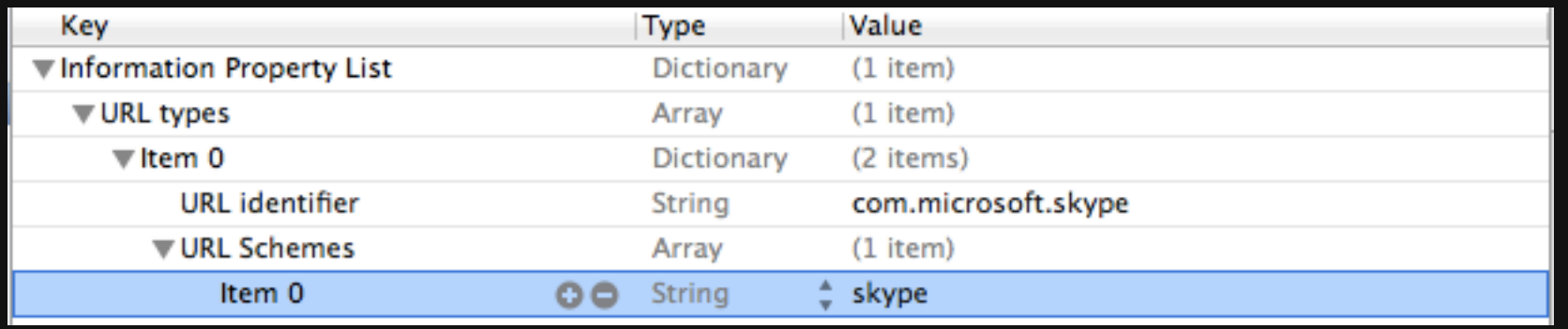
Two primary vectors of attack:

- iOS URL handlers
- Android intent handlers

Primary attack vectors:

- Cross-Application abuse
- Client Side Injection (browser, other app)

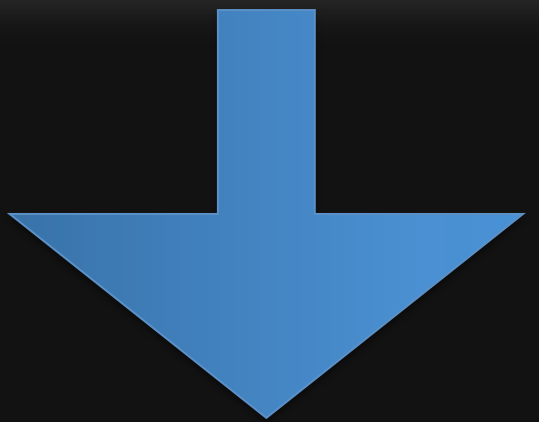
M7: Security Decisions Via Untrusted Inputs (Example)



Key	Type	Value
▼ Information Property List	Dictionary	(1 item)
▼ URL types	Array	(1 item)
▼ Item 0	Dictionary	(2 items)
URL identifier	String	com.microsoft.skype
▼ URL Schemes	Array	(1 item)
Item 0	String	skype

M7: Security Decisions Via Untrusted Inputs (Example)

Key	Type	Value
▼ Information Property List	Dictionary	(1 item)
▼ URL types	Array	(1 item)
▼ Item 0	Dictionary	(2 items)
URL identifier	String	com.microsoft.skype
▼ URL Schemes	Array	(1 item)
Item 0	String	skype



```
<iframe src="skype://kiwicon/910November2013?call"></iframe>
```

M8: Side Channel Data Leakage

Mobile devices are very “clever”

Contain all sorts of features

Law enforcement “dreamland”

Items of Interest:

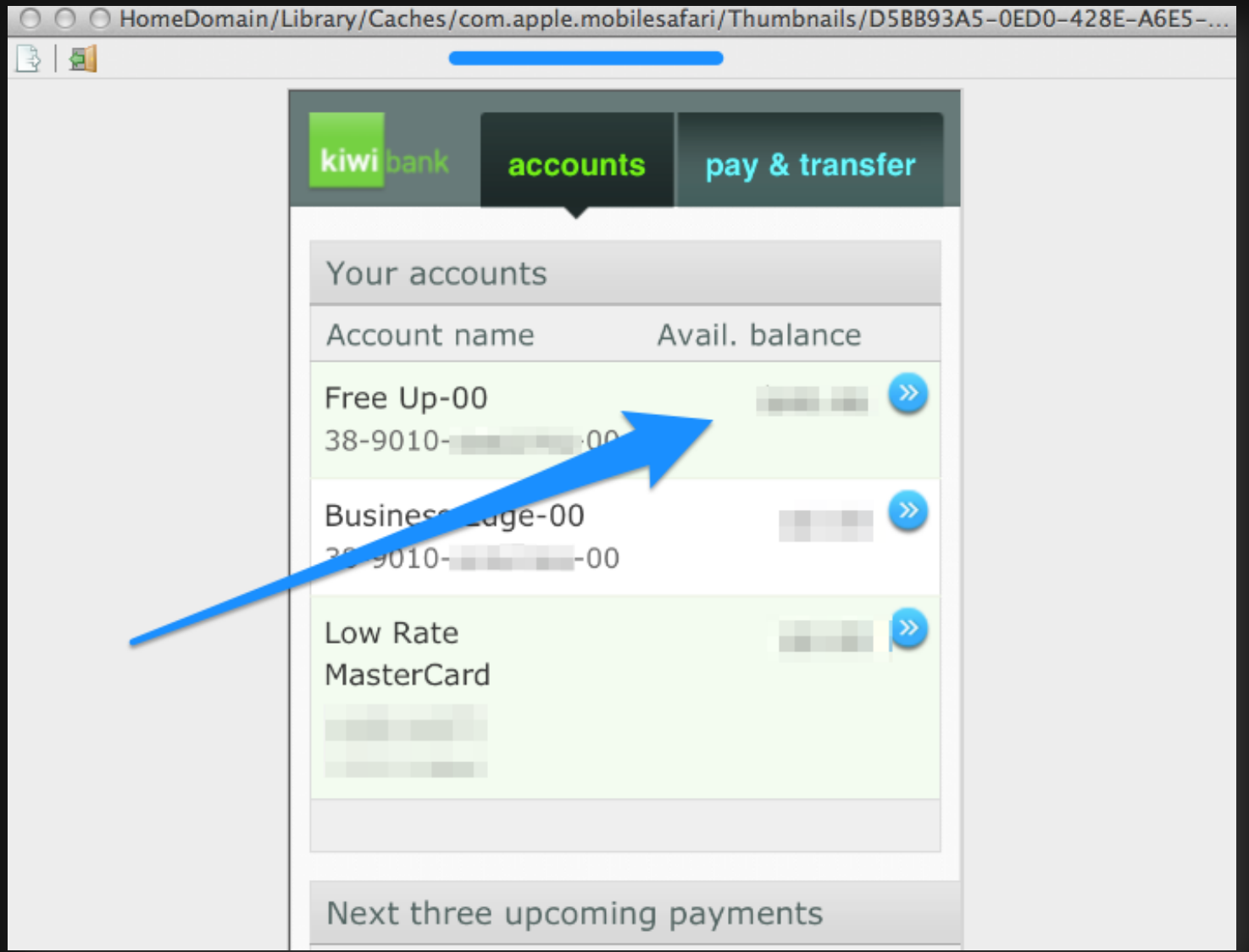
- Automatic screenshots
- Web caches
- Temp directories
- Console logging
- Autocorrect dictionaries

M8: Side Channel Data Leakage (Example)

INSOMNIA SECURITY SPECIALISTS :: REST SECURED

```
742:~ mark$ cd ~/stash/forensics/
742:forensics mark$ ls -l en_AU-dynamic-text.dat
-rw-r--r--  1 mark  staff  7076  9 Sep 21:56 en_AU-dynamic-text.dat
742:forensics mark$ strings -a en_AU-dynamic-text.dat | grep -i kiwicon
kiwicon
```

M8: Side Channel Data Leakage (Bonus)



M9: Broken Cryptography

Known secure libraries incorrectly implemented

NEVER ROLL YOUR OWN!

Common “encryption” implementations:

- Encoding (Base64)
- Obfuscation (XOR)
- Serialisation (go see Tom at 3pm)

M9: Broken Cryptography

Known secure libraries incorrectly implemented

NEVER ROLL YOUR OWN!

Common “encryption” implementations:

- Encoding (Base64)
- Obfuscation (XOR)
- Serialisation (go see Tom at 3pm)
- Combine all of the things!

Avoid known weak algorithms (MD5 etc)

Avoid known weak ciphers

M9: Broken Cryptography “Example”

M9: Broken Cryptography “Example”

M10: Sensitive Information Disclosure

Similar to M1 but hardcoded values

Everything can be reversed

Application assets, binaries and storage

Hardcoded secrets will always be revealed

Often identified “secrets” include:

- API keys
- Passwords
- Developer / Debug functionality

Conclusion

"Same bugs, different platform"

Many browser mitigations are lost in applications

Lack of understanding of mobile application relationships

Updated Top 10 due sometime soon (2013)

We expect to see little change given our experiences

www.insomniasec.com