



Software Security Goes Mobile

Jacob West
Director, Software Security Research

HP Enterprise Security



RSACONFERENCE2012

Motivation



Redefining the phone and the computer

Money: Beyond ringtones and 99¢ games



Landscape

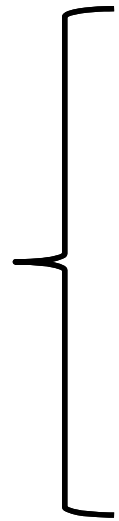


What is mobile?

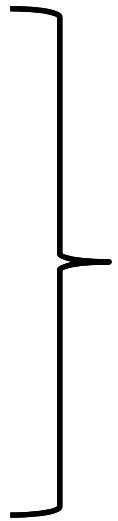
What **matters**?

Who cares?

Mobile Threats



Seven ways to
hang yourself
with **Google Android**



Parting Thoughts



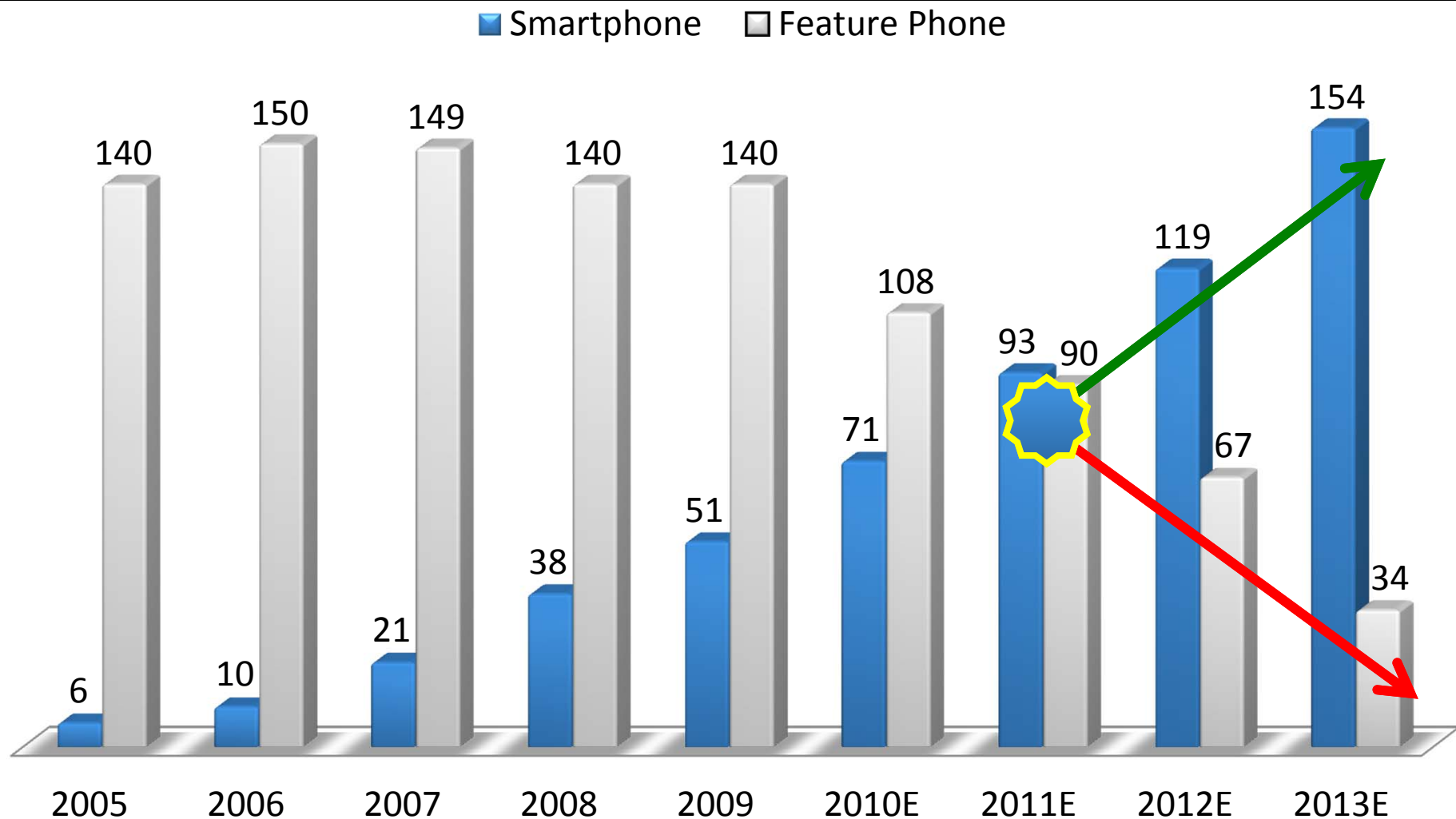
Questions you can ask
to begin **improving** your
mobile security today



Motivation

RSA CONFERENCE 2012

Smartphones > Feature Phones



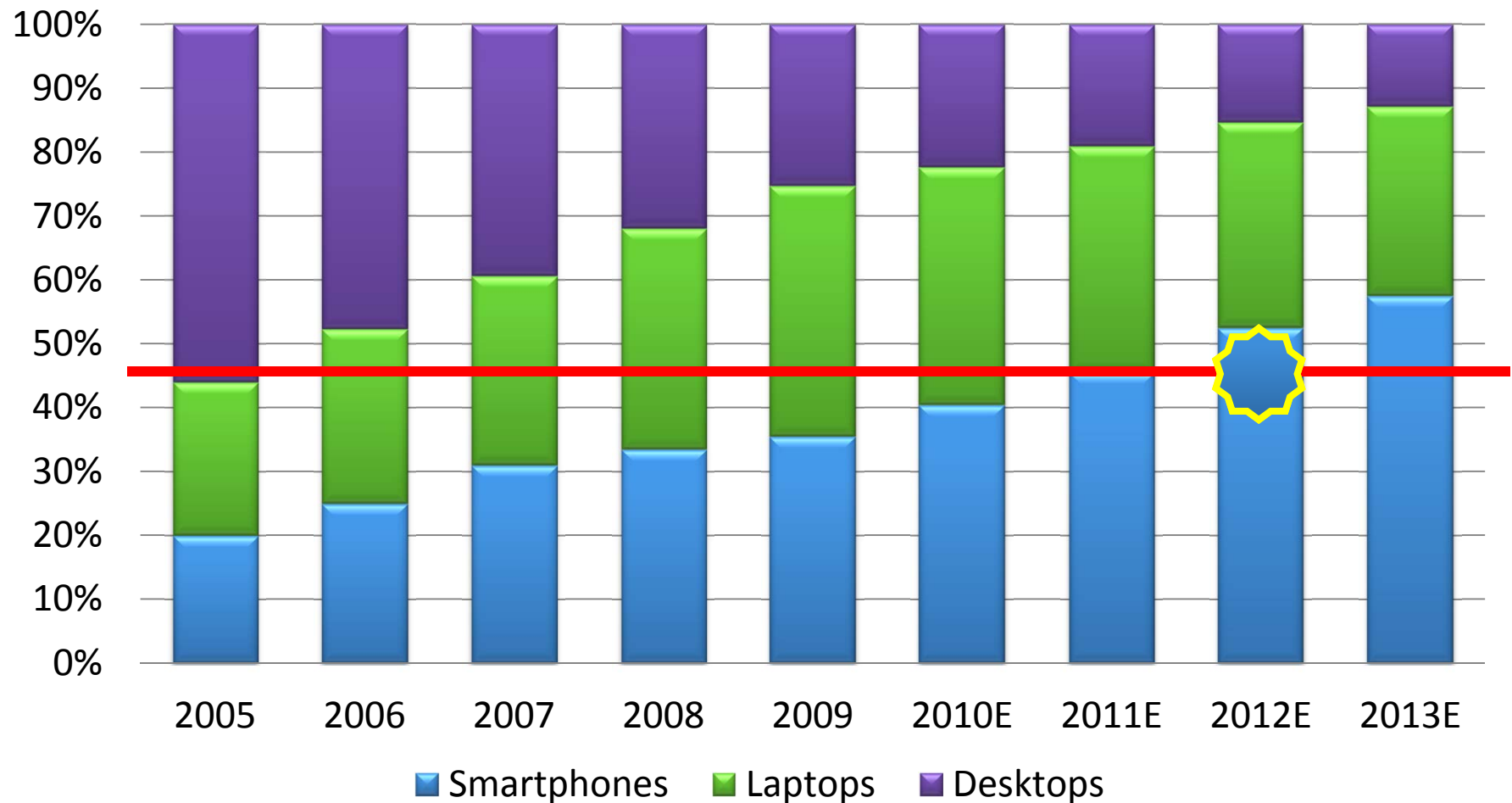
Source: Morgan Stanley Research



Enterprise Security



Smartphones > PCs



Source: Morgan Stanley Research



Enterprise Security



Pages View

e Rise



?



$\frac{1}{2}$



2



$3\frac{1}{2}$



$6\frac{1}{2}$

Source: Morgan Stanley Research



Enterprise Security



Smartphones Serve As Pocket PCs and Extend Desktop Experience

81%
Browsed the internet

Smartphone Activities Within Past Week
(Excluding Calls)

77%
Used a search engine

68%
Used an App

48%
Watch videos

Source: The Mobile Movement Study, Google/Ipsos OTX MediaCT, Apr 2011

Base: Smartphone Users (5013).

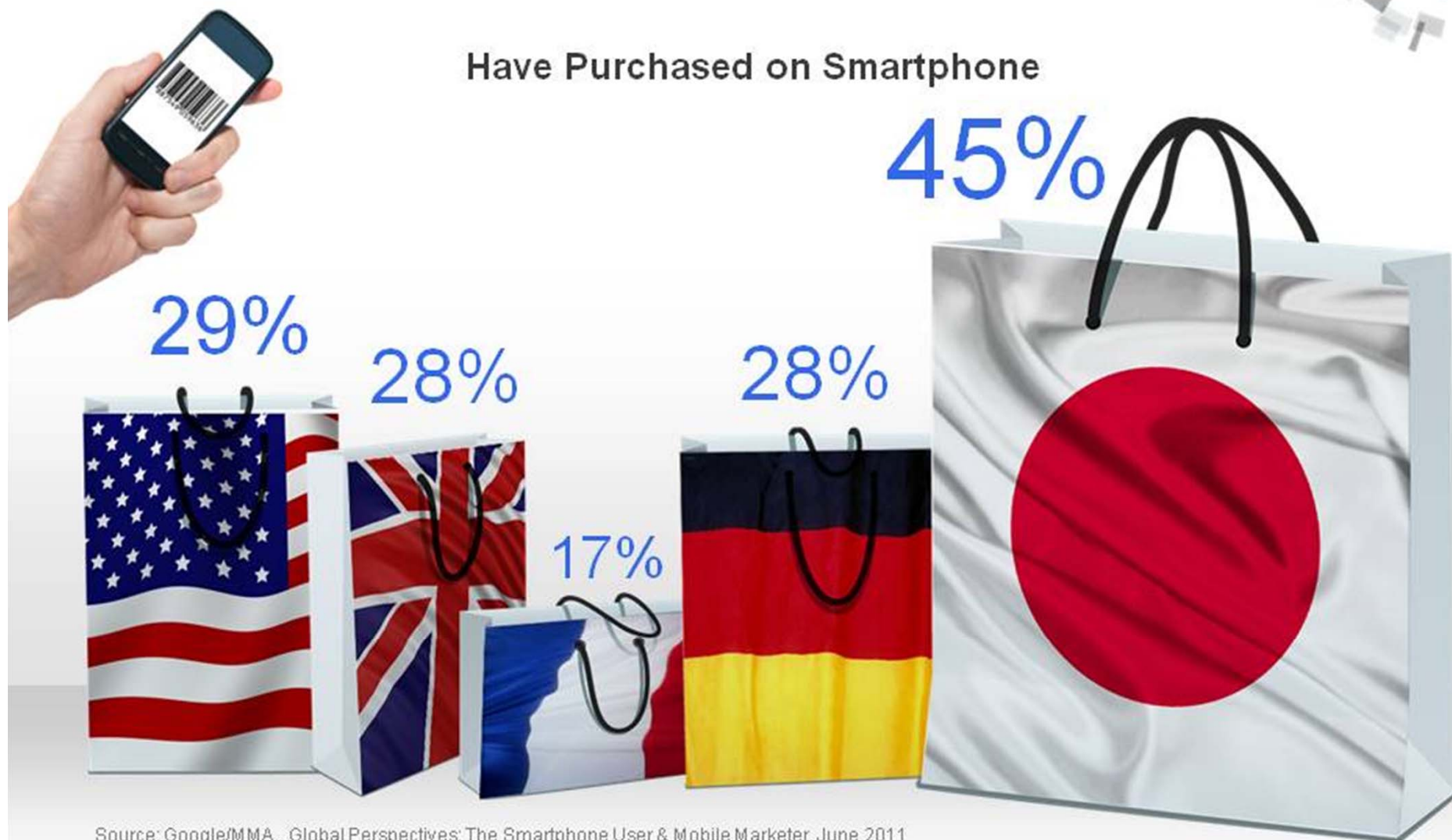
Q. Aside from making or receiving calls, which of the following activities, if any, have you done on your smartphone in the past week?

thinkmobile
with Google

Mobile is an Emerging Point of Purchase



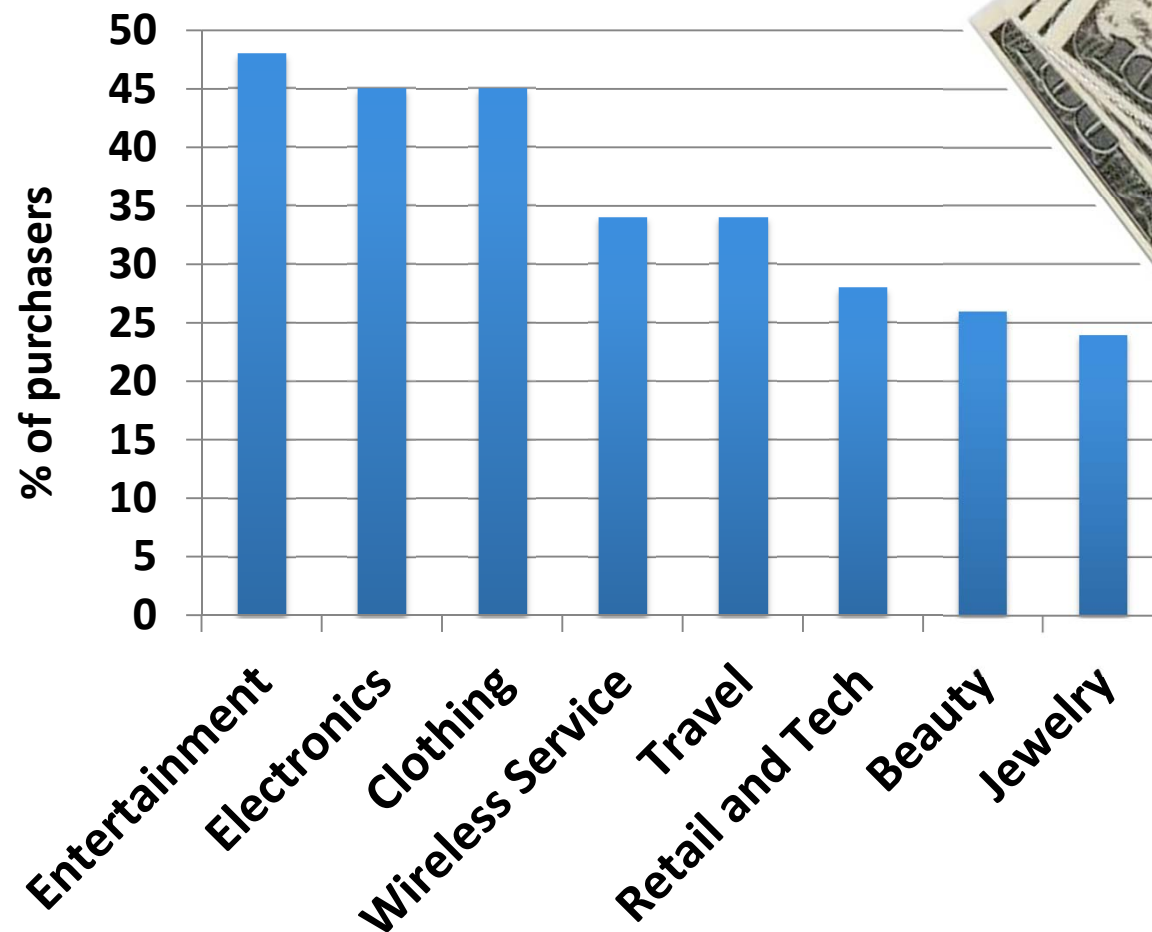
Have Purchased on Smartphone



Source: Google/MMA, Global Perspectives: The Smartphone User & Mobile Marketer, June 2011
Base: Smartphone Users (US: 6000; UK: 2000; FR: 2000; DE: 2000; JP: 1000).

Q. Have you ever purchased a product or service over the Internet on your smartphone?

Mobile Purchasers



spend \$300/year

Source: Google The Mobile Movement Study



Enterprise Security

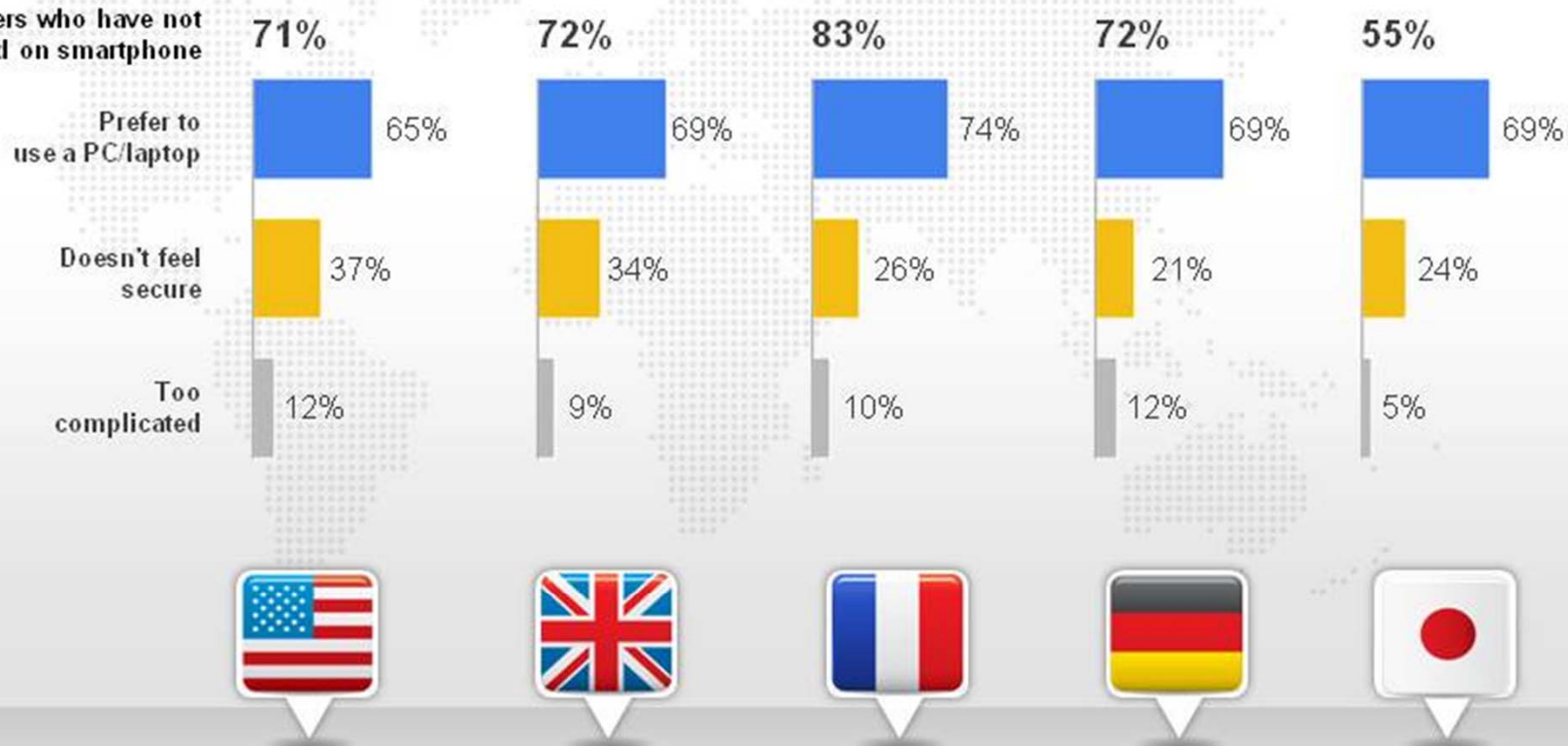


Preference For Computer and Concerns of Security Are Main Barriers to Mobile Purchasing



Reasons for Not Purchasing on Smartphone

Users who have not purchased on smartphone



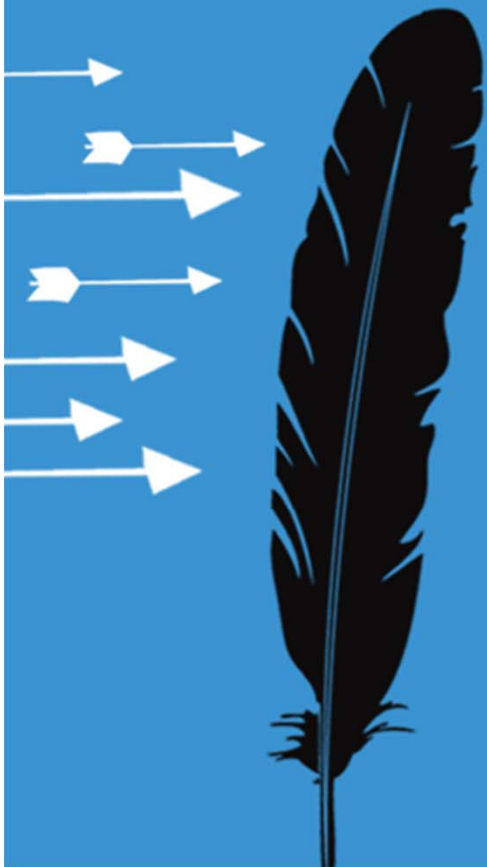
Source: Google/MMA, Global Perspectives: The Smartphone User & Mobile Marketer, June 2011

Base: Smartphone Users (US: 6000; UK: 2000; FR: 2000; DE: 2000; JP: 1000).

Base: Smartphone Users Who Have Not Made a Purchase on Device (US: 4444; UK: 1559; FR: 1653; DE: 1442; JP: 554).

Q. Why have you not made a purchase using your smartphone?

Mobile Landscape



RSACONFERENCE2012

Big Questions

- What is mobile?
- What matters?
- Who cares?

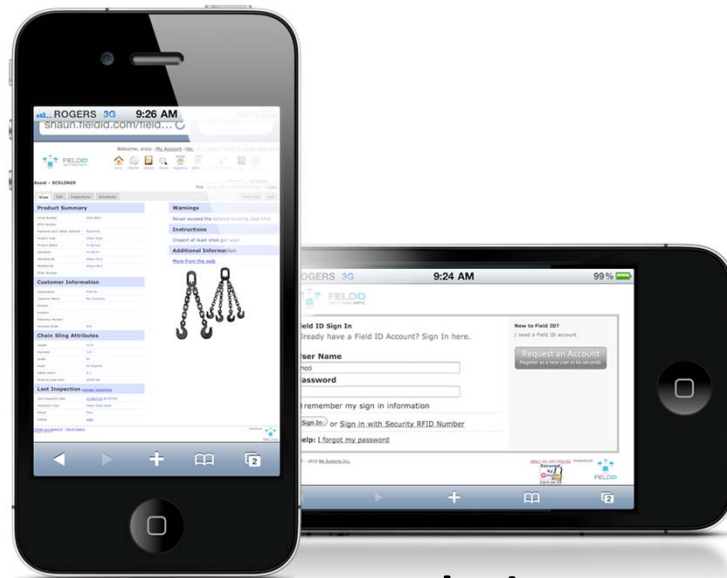


Enterprise Security

RSACONFERENCE2012



What *is* Mobile?



device

connection



server

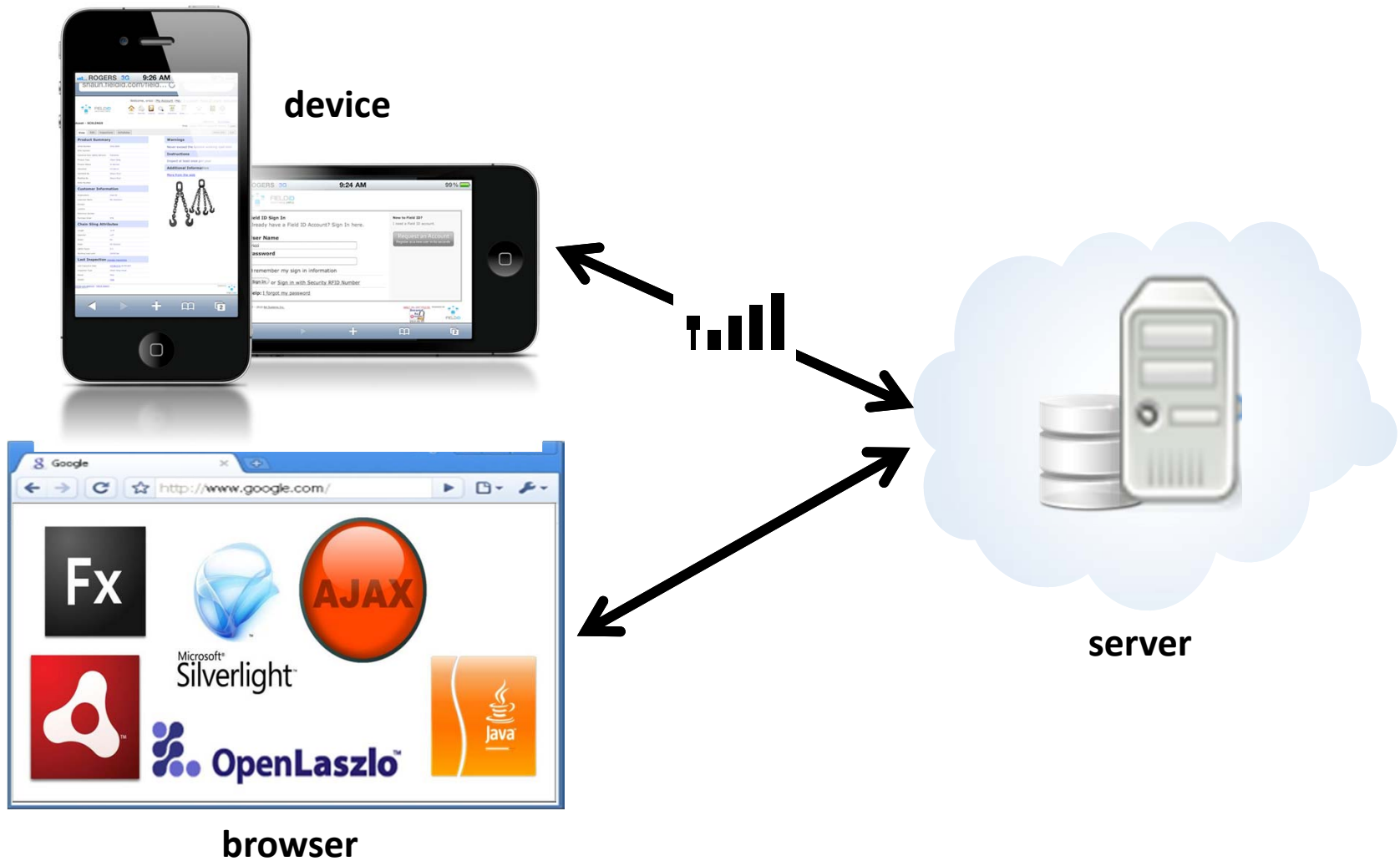


Enterprise Security

RSA CONFERENCE 2012



Familiar Model

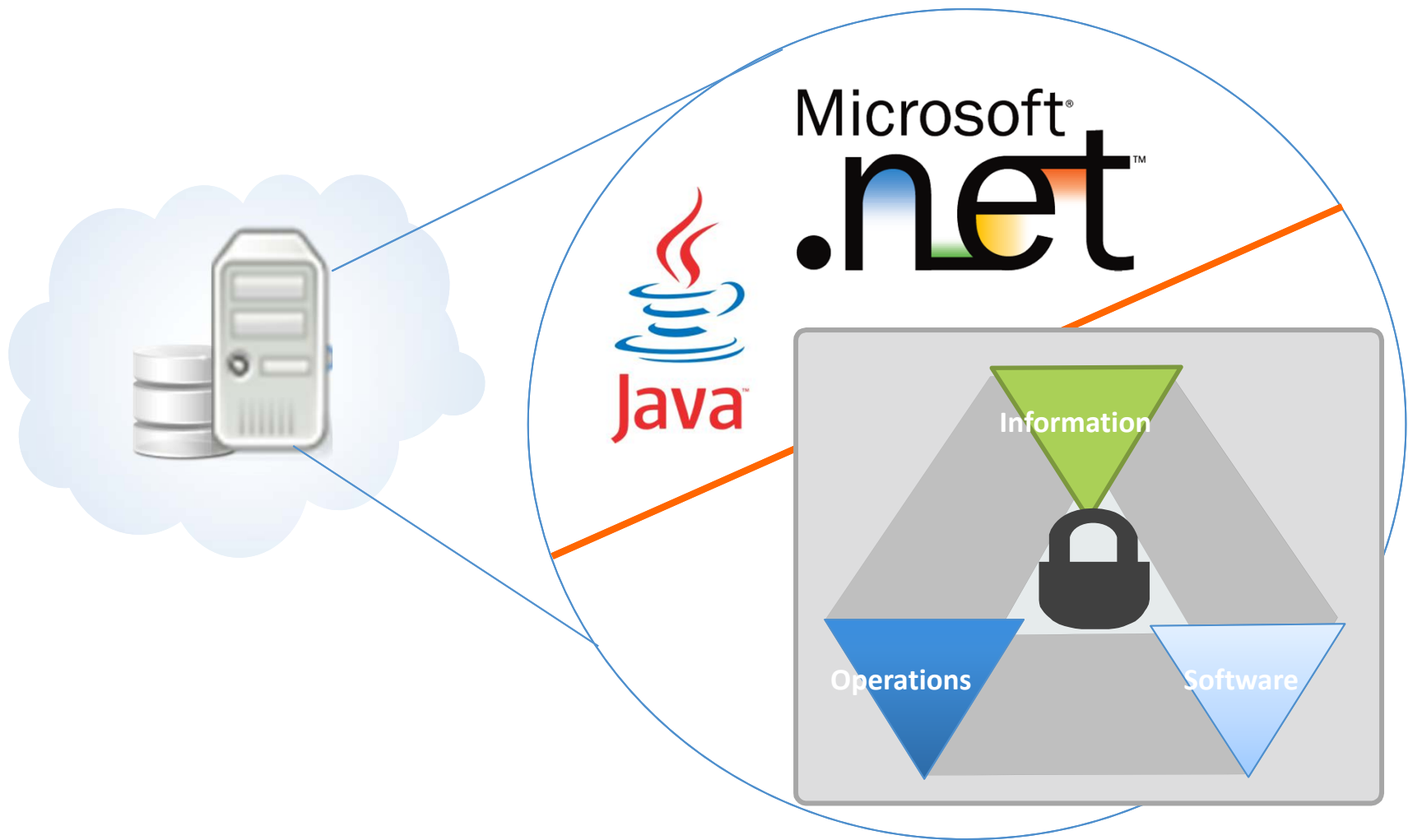


Enterprise Security

RSA CONFERENCE 2012



Same Ol' Server



Enterprise Security

RSACONFERENCE2012



Client-Side Persistence



- Local data persistence
- Similar to HTML 5
- Invisible to users and always available



Mobile OS



- Benefit of hindsight
- Security features
 - Read-only stack
 - Data encryption
 - Permissions
- Confusing
 - Wait, permissions?



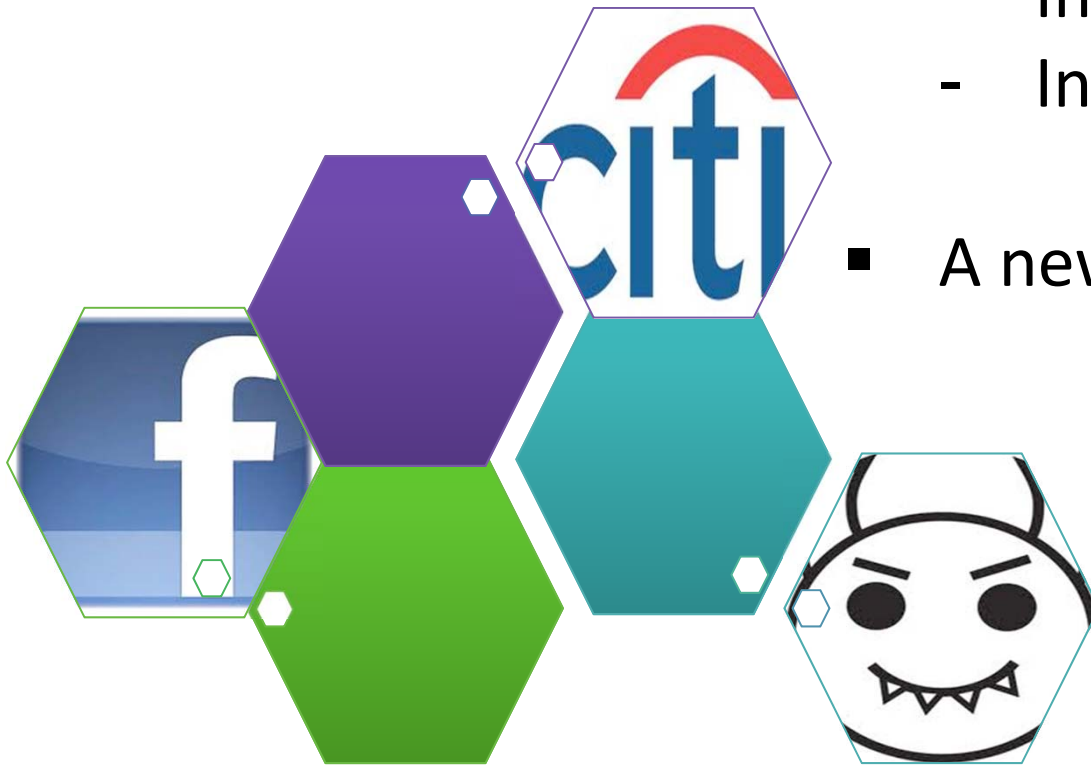
Enterprise Security

RSACONFERENCE2012



Can't We All Get Along?

- Formal communication
 - Inter-application
 - Intra-application
- A new trust boundary



What Matters?

Old

- Handling sensitive user and app data
- Environment and configuration
- Old standbys like XSS and SQL injection



New

- Local storage (SD card, ...)
- Communication (SMS, MMS, GPS, ...)
- Security features (Permissions, backups, ...)



Who Cares?

App Owners and Developers

- Confidentiality, Integrity, **A**vailability (intra-app)

Device Manufacturer / OS Builder / Providers

- Confidentiality, Integrity, **A**vailability (inter-app)
- Privacy beyond data (location, behavior, ...)
- Generating unexpected charges

Users

- To be determine...





Mobile Threats

RSA CONFERENCE 2012

Google Android Vulnerabilities

1. Intent Hijacking
2. Intent Spoofing
3. Sticky Broadcast Tampering
4. Insecure Storage
5. Insecure Network Communication
6. SQL Injection
7. Promiscuous Privileges



1. Intent Hijacking

Description

- Malicious app intercepts an intent bound for another app, which can compromise data or alter behavior

Cause

- Implicit intents
(do not require strong permissions to receive)

Fix

- Explicit intents and require special receiver permissions



1. Intent Hijacking

IMDb App

Showtime
Search

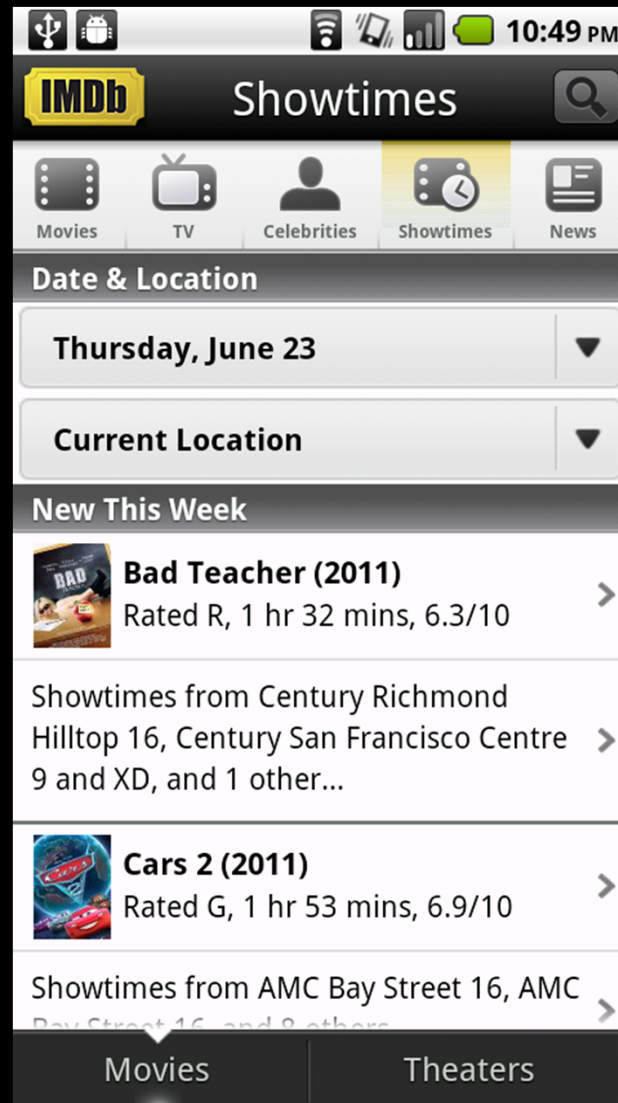


Implicit Intent
Action: *willUpdateShowtimes*

Results UI

Handles Actions:
willUpdateShowtimes,
showtimesNoLocationError

1. Intent Hijacking



1. Intent Hijacking

IMDb App

Showtime
Search



Implicit Intent
Action: *willUpdateShowtimes*

Results UI

Handles Actions:
willUpdateShowtimes,
showtimesNoLocationError

Eavesdropping App



Handles Actions:
willUpdateShowtimes,
showtimesNoLocationError

Malicious
Receiver

2. Intent Spoofing

Description

- Malicious app spoofs a legitimate intent, which can inject data or alter behavior

Cause

- Public components
(necessary to receive implicit intents)

Fix

- Use explicit intents and receiver permissions
- Only perform sensitive operations in private components



2. Intent Spoofing

Spoofing App



Action:
showtimesNoLocationError

Malicious
Component



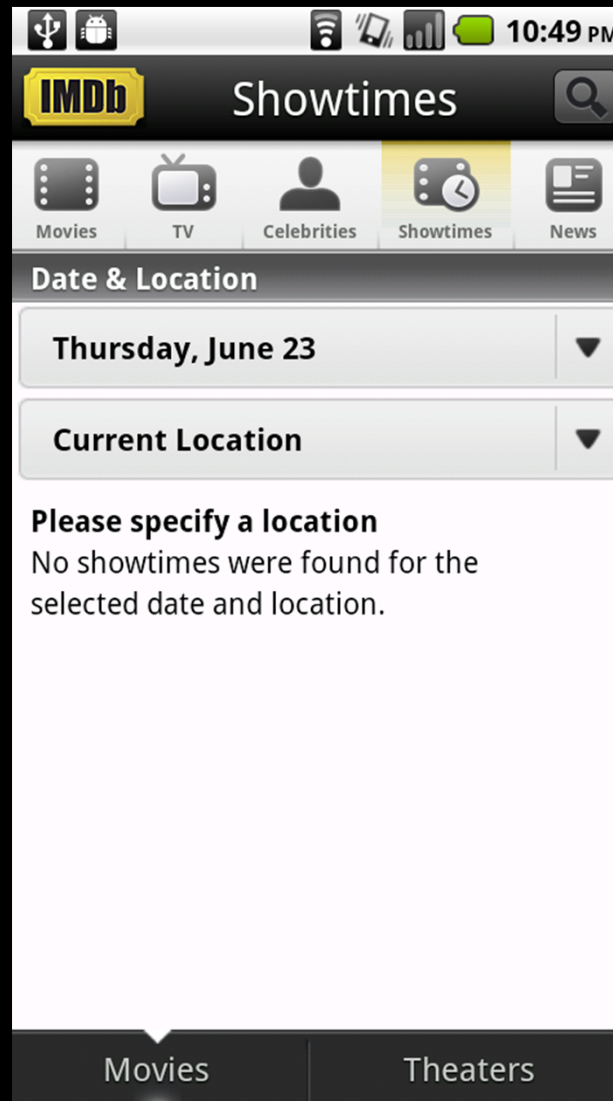
IMDb App

Showtime
Search

Results UI

Handles Actions:
willUpdateShowtimes,
showtimesNoLocationError

2. Intent Spoofing



3. Sticky Broadcast Tampering

Description

- Persistent intents used by legitimate apps can be accessed and removed by malicious apps

Cause

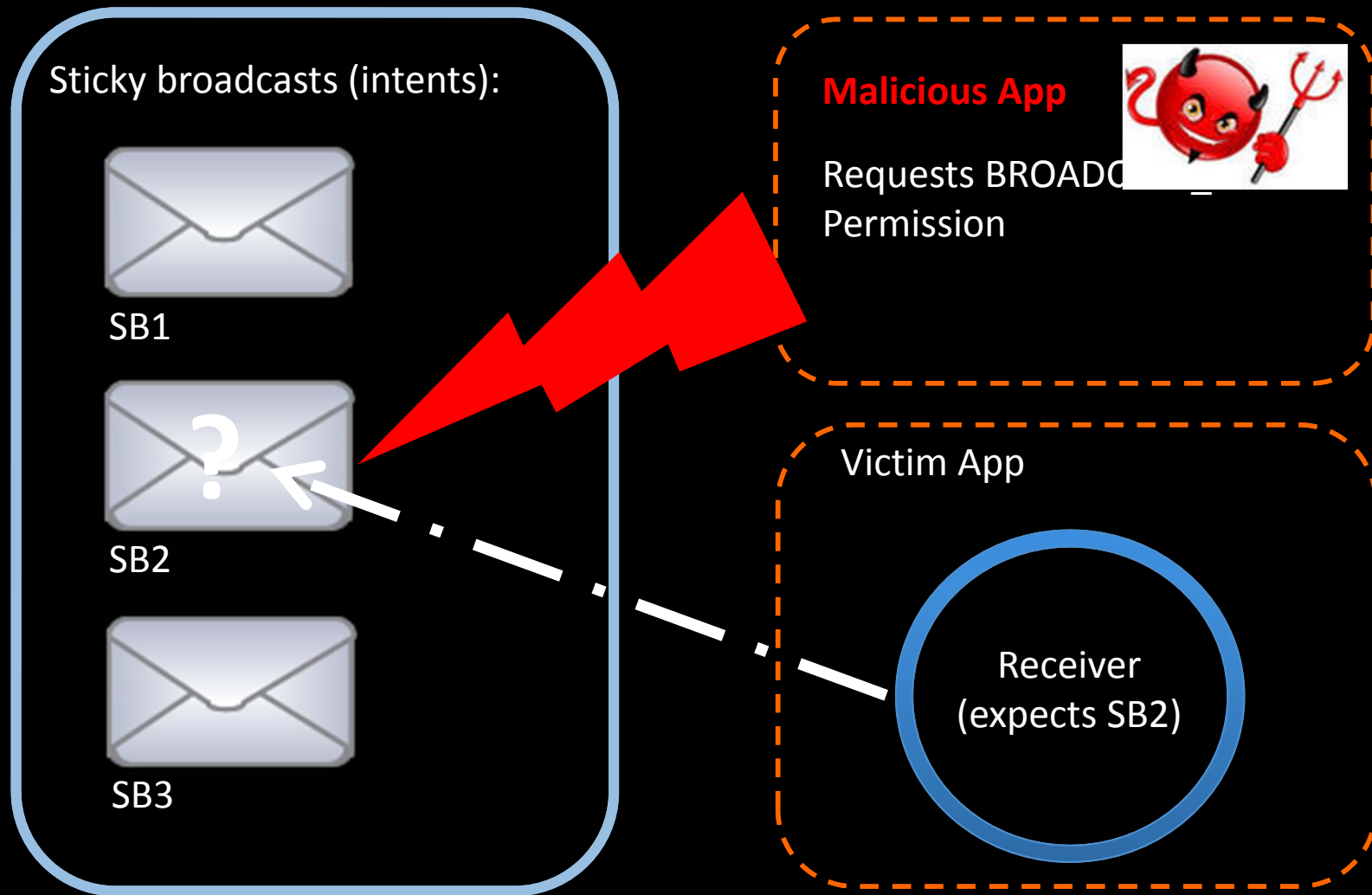
- BROADCAST_STICKY allows to full access to any sticky broadcasts

Fix

- Use explicit, non-sticky broadcasts protected by receiver permissions



3. Sticky Broadcast Tampering



4. Insecure Storage

Description

- Local storage is accessible to attackers, which can compromise sensitive program data

Causes

- Local files are world-readable and outlive uninstall

Fix

- Use SQLite database
- Use internal storage and make the file private
- Encrypt the data (Don't store the key on SD card!)



4. Insecure Storage

- Kindle app saves e-books (.mbp and .prc) in a folder on the SD card
 - Depending on DRM, accessible to other apps
 - Saves covers of books (privacy violation)
 - Folder is retained after uninstall of app

5. Insecure Communication

Description

- Data sent over unencrypted channels can be intercepted by attackers sniffing network

Cause

- Non-HTTPS WebView connections

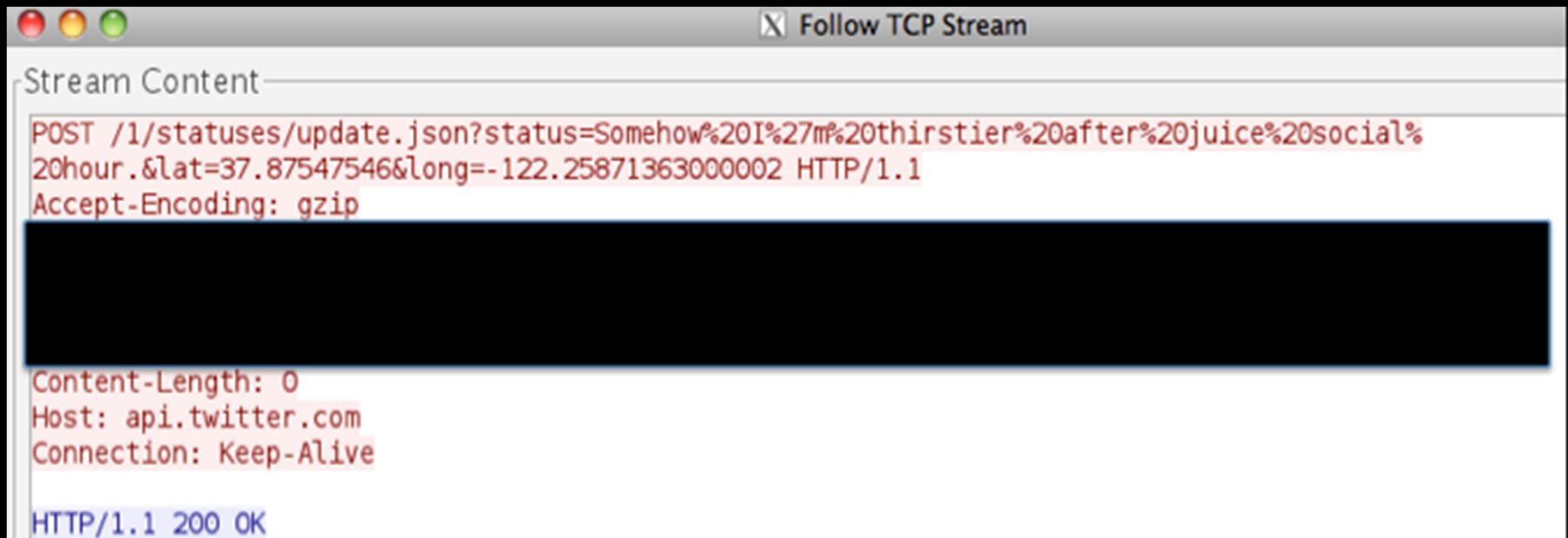
Fix

- Ensure sensitive data only sent over encrypted channels



5.a Insecure Communication

- Twitter: Tweets are sent in the clear



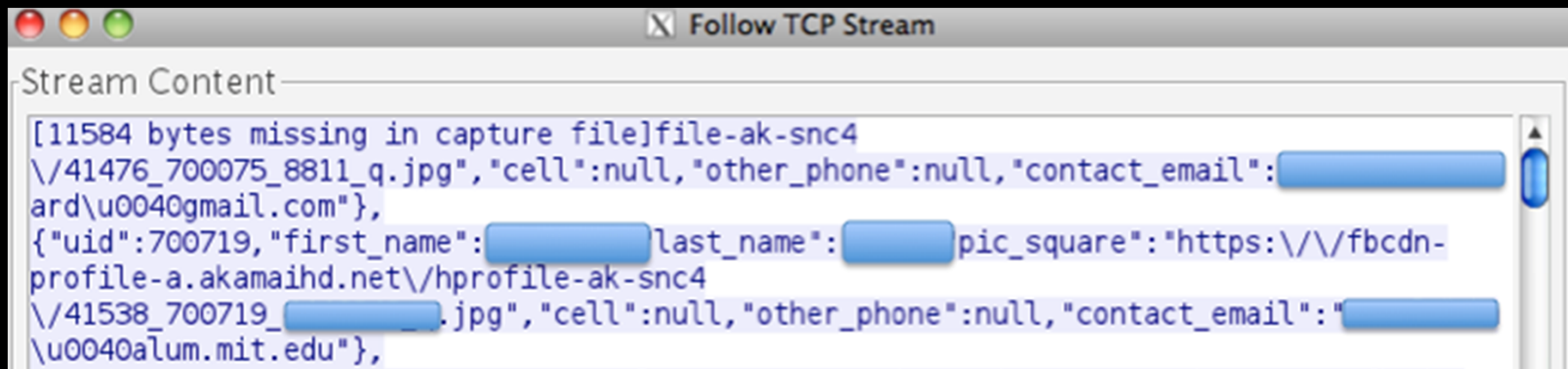
The screenshot shows a window titled "Follow TCP Stream" with a "Stream Content" tab. The content displays a clear-text HTTP POST request to the Twitter API. The request line is "POST /1/statuses/update.json?status=Somehow%20I%27m%20thirstier%20after%20juice%20social%20hour.&lat=37.87547546&long=-122.25871363000002 HTTP/1.1". The headers include "Accept-Encoding: gzip", "Content-Length: 0", "Host: api.twitter.com", and "Connection: Keep-Alive". The response line is "HTTP/1.1 200 OK". A large black rectangular box redacts the body of the request.

```
POST /1/statuses/update.json?status=Somehow%20I%27m%20thirstier%20after%20juice%20social%20hour.&lat=37.87547546&long=-122.25871363000002 HTTP/1.1
Accept-Encoding: gzip
Content-Length: 0
Host: api.twitter.com
Connection: Keep-Alive
HTTP/1.1 200 OK
```

<https://freedom-to-tinker.com/blog/dwallach/things-overheard-wifi-my-android-smartphone>

5.b Insecure Communication

- Facebook: Despite 'fully encrypted' option on the Web, mobile app sends in the clear



6. SQL Injection

Description

- Allows malicious users to alter or view (query string injection) database records

Cause

- Untrusted data used to construct a SQL query or clause

Fix

- Parameterized queries



6. SQL Injection

```
c = invoicesDB.query(  
    Uri.parse(invoices),  
    columns,  
    "productCategory = '" +  
        productCategory + "' and  
        customerID = '" + customerID + "'",  
    null, null, null,  
    "'" + sortColumn + "'",  
    null  
);
```

HI, THIS IS
YOUR SON'S SCHOOL.
WE'RE HAVING SOME
COMPUTER TROUBLE.



6. SQL Injection

```
productCategory = Fax Machines  
customerID = 12345678  
sortColumn = price
```

```
select * from invoices  
  where productCategory = 'Fax Machines' and  
     customerID = '12345678'  
  order by 'price'
```

OH, DEAR - DID HE
BREAK SOMETHING?

IN A WAY -)



6. SQL Injection

```
productCategory = Fax Machines'  
or productCategory = \"
```

```
customerID = 12345678
```

```
sortColumn = \" order by 'price
```

```
select * from invoices
```

```
where productCategory = 'Fax Machines' or
```

```
productCategory = \"'--and customerID =
```

```
'12345678'--order by '\"
```

```
order by 'price'
```

DID YOU REALLY
NAME YOUR SON
Robert'); DROP
TABLE Students;-- ?



OH, YES. LITTLE
BOBBY TABLES,
WE CALL HIM.

6. SQL Injection

```
c = invoicesDB.query(  
    Uri.parse(invoices),  
    columns,  
    "productCategory = ? and customerID = ?",  
    {productCategory, customerID},  
    null,  
    null,  
    "sortColumn = ?",  
    sortColumn  
);
```

WELL, WE'VE LOST THIS
YEAR'S STUDENT RECORDS.
I HOPE YOU'RE HAPPY.



AND I HOPE
YOU'VE LEARNED
TO SANITIZE YOUR
DATABASE INPUTS.

7. Promiscuous Privileges

Description

- Requesting unneeded permits privilege escalation attacks and desensitize users to privilege requests

Causes

- **Deputies**
- Artifacts from testing
- **Confusion (inaccurate/incomplete doc, forums)**

Fix

- Identify unnecessary permissions before shipping



7.a Promiscuous Privileges

User App

Does NOT need
CAMERA permission

Wants Picture



Implicit Intent
Action: *IMAGE_CAPTURE*

Camera App

Needs CAMERA
permission

Takes
Picture

Handles Action:
IMAGE_CAPTURE

7.a Promiscuous Privileges

Third hit on Google search

3 Answers

active

oldest

votes



It broadcasts whenever you connect or disconnect from Wifi, in other words, Wifi State.

8

You can do it using the following intent-filters:

- android.net.wifi.WIFI_STATE_CHANGED
- action android:name="android.net.wifi.STATE_CHANGE
- android.net.wifi.action.CONNECTION_CHANGE



Which needs the following permission:

- uses-permission android:name="android.permission.ACCESS_WIFI_STATE"

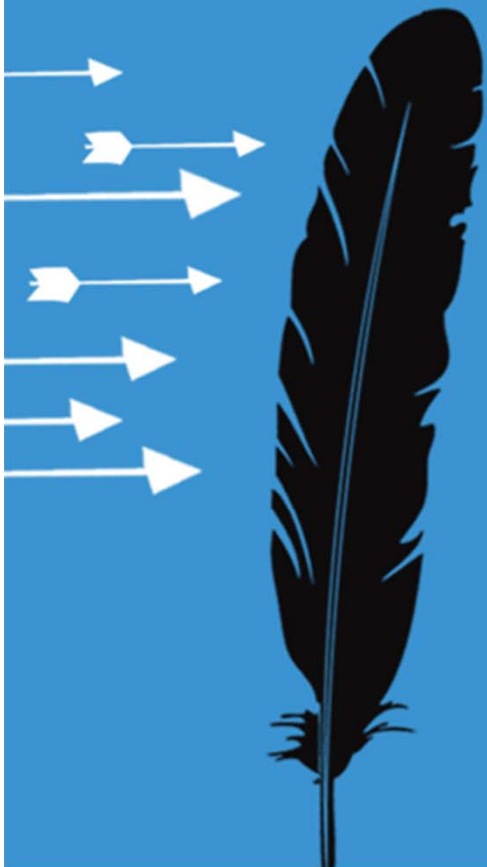
Not true for android.net.wifi.STATE_CHANGE

Empirical Results: DEFCON '11

Vulnerability Type	% of apps with > 1
1. Intent Hijacking	50%
2. Intent Spoofing	40%
3. Sticky Broadcast Tampering	6%
4. Insecure Storage	28%
5. Insecure Communication	N/A
6. SQL Injection	17%
7. Promiscuous Privileges	31%



Parting Thoughts



RSA CONFERENCE 2012

What Questions to Ask?

- What do the apps do and for whom?
- What platform(s) do your apps support and how?
- Who develops your apps and where?
- Is there an existing SDL for other development?
- Do you rely on platform providers or app distributors for any security assurance?
- Are mobile apps prompting back-end changes?
- Are your apps appropriately permissioned?

