



Whack-A-Mobile II

Mobile Penetration Testing with MobiSec



Tony DeLaGrange,
& Kevin Johnson
Senior Security Consultants
info@secureideas.net
Office - 904-639-6709
Twitter - @secureideasllc

Tony DeLaGrange

- Security Consultant at Secure Ideas
- Info Sec related roles for past 12 years
- Co-Author of SEC571 Mobile Device Security
- Project Lead for the MobiSec Live Environment
- Co-Chair of the SANS Mobile Device Summit
- Avid Sailor - RC-27 "Daddio"



Kevin Johnson

- Security Consultant at Secure Ideas
- Author of SEC542/642/571
 - Web App PenTesting/Adv Web PenTesting/Mobile Security
- SANS Senior Instructor
- Open Source Project Lead
 - SamuraiWTF, Laudanum, Yokoso, WeaponizedFlash etc.
- Co-Chair of the SANS Mobile Device Summit



Thank You Chris Cuevas!

- Security Consultant at Secure Ideas
- Contributor to SamuraiWTF and MobiSec
- Co-Author of Sec571
- SANS Mentor
- Thanks for all the help on building & testing MobiSec
 - and for dressing up for this pic!



Let's Talk About...

- Overview of the MobiSec Live Environment
- MobiSec Structure & Testing Tools
- ADB is Your Friend for Talking Android
- Finding Data Nuggets on an Android Device
- Sniffing Traffic from an Android Emulator
- Capturing & Manipulating Web Requests
- Hooking Mobile Devices with BeEF
- What's New with MobiSec v1.1
- OWASP Mobile Security Project

MobiSec Live Environment

- What is it? Why did we do this?
- Similar to
 - SamuraiWTF
 - BackTrack
- DARPA CFT Project
- Open Source project
 - Version 1.0 released Feb 2012



MobiSec Design Objectives

- Live testing environment on Intel computers
- Based on an OS *everyone* is familiar with
- Open source and distributable
- Structure aligned to testing methodology
- Easy to find & use tools
- Include development kits and emulators
- Customizable
- Updateable
- Cool name and logo - "catch them all!" ☺

MobiSec Build

- Run as Live Environment from DVD/USB/VM
- Hardware or VM Settings Specs:
 - Single 32-bit processor / Two processors preferred
 - 1GB Memory / More is preferred
 - 15GB HD / More if you want to customize
 - USB (for Ubertooth and USB connect to devices)
 - 802.11 (for WiFi analysis)
- Download available at:
<http://sourceforge.net/p/mobisec>

Mobile Testing Methodology

- We aligned the pen testing tools to a well known pen testing methodology

- ✦ Reconnaissance
- ✦ Mapping
- ✦ Discovery
- ✦ Exploitation



- If you're not using a testing methodology, then adopt a good one and USE IT!

MobiSec Structure

- MobiSec is organized to categorize tools:
 - Development Tools
 - Device Forensics
 - Penetration Testing
 - Reverse Engineering
 - Wireless Analyzers
- Menu and directory structure
 - Similar to other testing environments you're already use to ☺

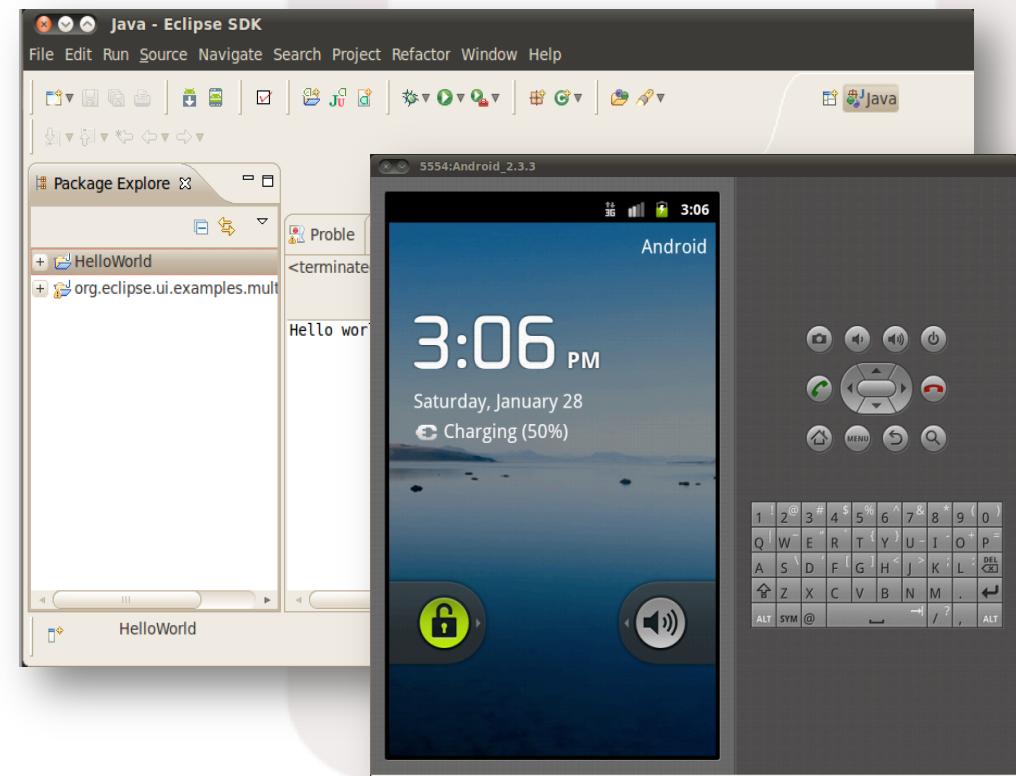


Development Tools

- Includes mobile device development environments, emulators and simulators
 - Android SDK
 - Android Emulators
 - Eclipse IDE



AndroidLabs



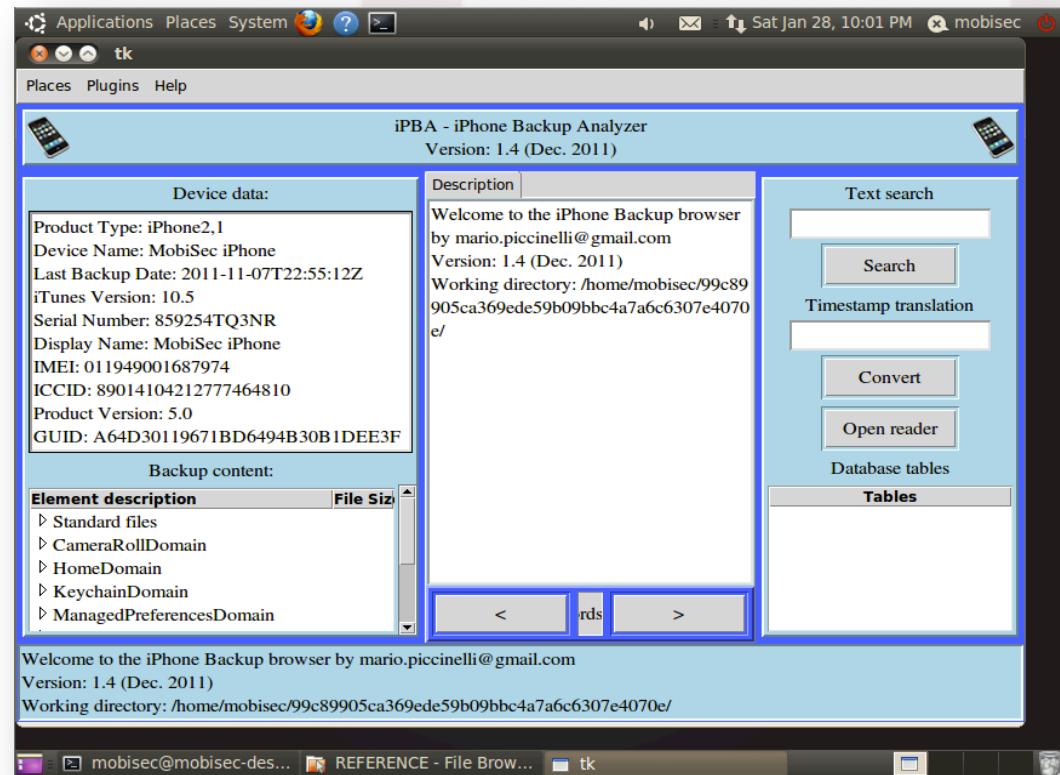
Forensics Tools

- Includes tools that provide the ability to perform forensics on mobile devices

- BitPim
- Foremost
- iPhone Backup Analyzer
- The Sleuth Kit

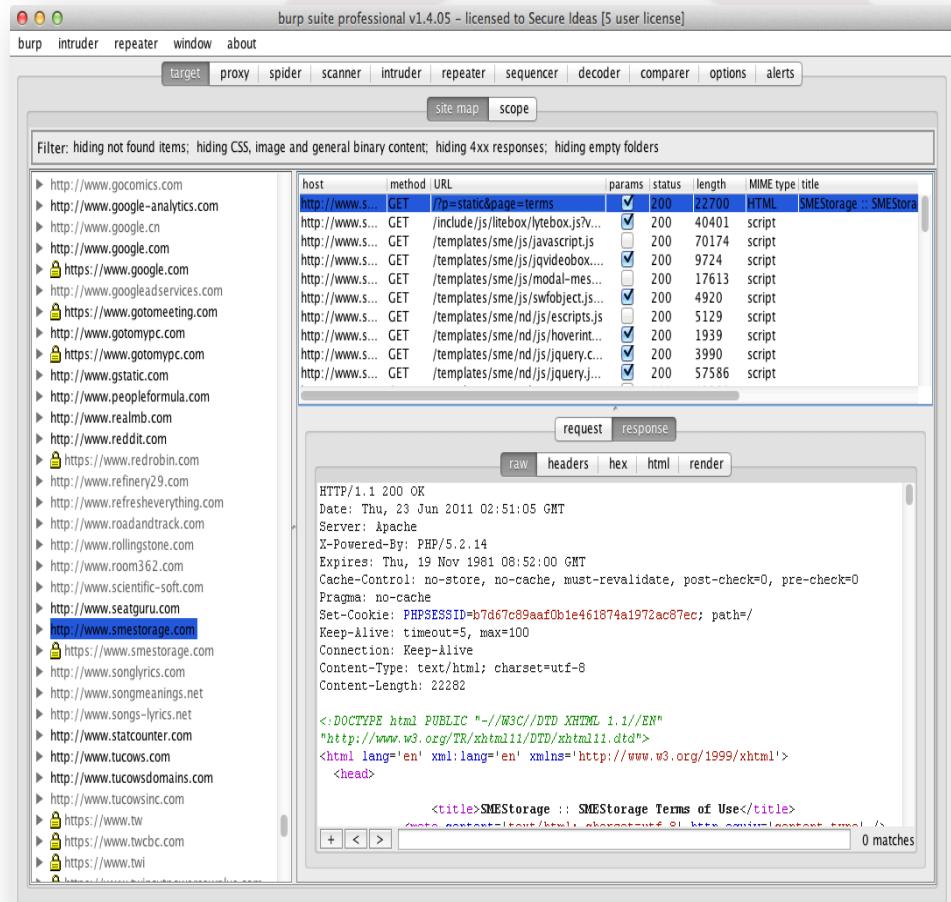


SQLiteSpy



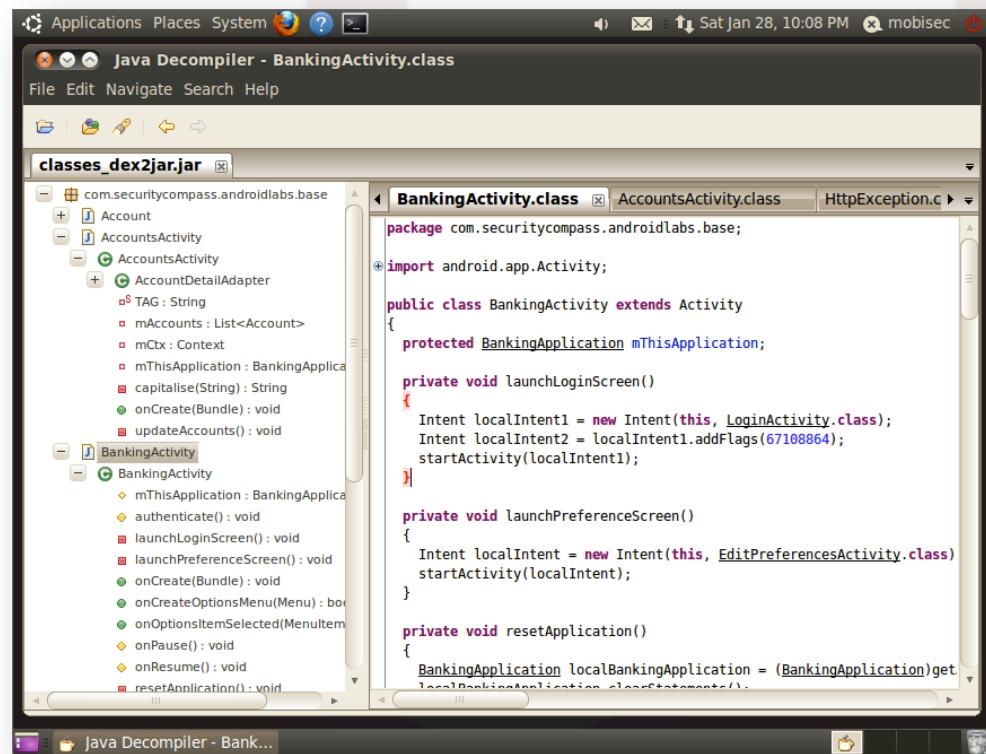
Penetration Testing Tools

- Reconnaissance
 - Maltego CE, SEAT
 - Mapping
 - CeWL, DirBuster, Fierce, Nikto, nmap
 - Discovery
 - Burp, w3af, ZAP
 - Exploitation
 - BeEF, Metasploit, SET
-  Ettercap, iSniff, NetSed, SQLMap, SSLStrip



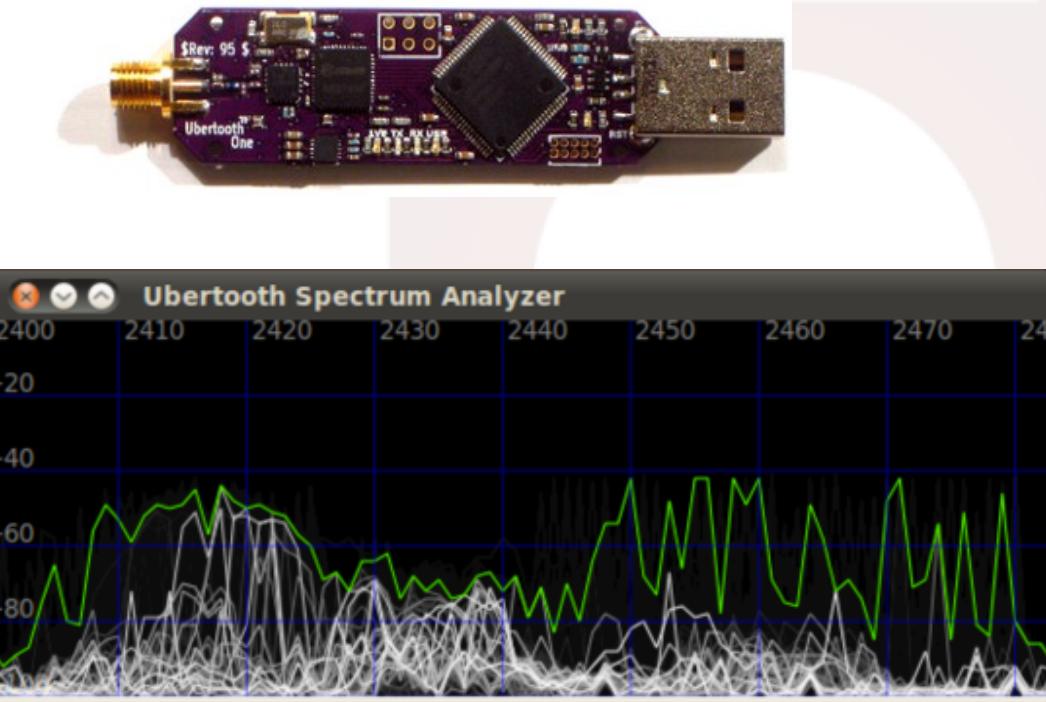
Reverse Engineering Tools

- Includes tools used for performing reverse engineering of mobile apps
 - APK Tool
 - Dex2Jar
 - Flawfinder
 - Java Decompiler
 - Strace



Wireless Analysis Tools

- Drivers and wireless tools for capturing and analyzing wireless traffic
 - Kismet
 - Ubertooth
 - Wireshark
-  Aircrack-ng



Mobile Attack Vectors

- From SmartBombs talk earlier today: there are three major attack vectors for mobile testing:
 - **File System**
What are apps writing to the file system?
How is data stored?
 - **Transport Layer**
How are apps communicating over the network?
TCP and Third-party APIs
 - **Application Layer**
How are apps communicating via HTTP and Web Services?
- Let's take a look at how MobiSec can be used...

Connect to Android Device via USB

- Connect android device via USB, and list with adb, but...
- Enable USB debugging on the Android device
 - Settings -> Applications
- List connected USB devices
 - Is VM connected to USB devices?
- Create /etc/udev/rules.d/51-android.rules

```
SUBSYSTEMS=="usb", ATTRS{idVendor}=="0955", ATTRS{idProduct}=="7100", MODE="0666"
```

- Restart udev and adb server
- Try again...

```
$ adb devices  
List of devices attached  
1714404641614517 device
```

```
$ adb devices  
List of devices attached  
???????????? no permissions
```

```
$ lsusb  
...  
Bus 001 Device 002: ID 0955:7100
```

```
$ sudo restart udev  
$ adb kill-server  
$ adb start-server
```

Getting shell on an Android Device

- adb shell to open shell on the device
 - defaults to connected device
- Uses shell account, can su to root, but prompted on the device
 - Can set default to always accept! ☺
- Use uname -a to get system info
- Use find to look for interesting database files
 - find / -name *.db | grep account
 - find / -name *.db | grep email

```
mobisec@mobisec-desktop: ~
File Edit View Terminal Help
mobisec@mobisec-desktop:~$ adb shell
$ whoami
shell
$ su
# whoami
root
# uname -a
Linux localhost 2.6.32.9-00000-10.8.2-dirty #13 SMP PREEMPT Mon Nov 15
20:14:21 EST 2010 armv7l GNU/Linux
# pwd
/
# find / -name *.db | grep email
/data/data/com.android.email/databases/EmailProvider.db
/data/data/com.android.email/databases/EmailProviderBody.db
/data/data/com.android.email/databases.EmailProvider.db
/data/data/com.android.email/databases.EMailProvider.db
#
```

Using SQLite3 to Find Data

- Let's take a closer look at that Email database
 - sqlite3 /data/data/com.android.email/databases/EmailProvider.db
- SQLite3 provides simple SQL commands
 - sqlite> .databases (list attached databases)
 - sqlite> .tables (list tables)
 - sqlite> .dump <table> (dump table contents)
- Let's find the email account configurations
 - .dump HostAuth
 - notice the passwords in cleartext?

```
mobisec@mobisec-desktop: ~
File Edit View Terminal Help
# sqlite3 /data/data/com.android.email/databases/EmailProvider.db
SQLite version 3.6.22
Enter ".help" for instructions
Enter SQL statements terminated with a ";"
sqlite> .tables
Account          HostAuth          Message          Message_Updates
Attachment       Mailbox           Message_Deletes  android_metadata
sqlite> .dump HostAuth
PRAGMA foreign_keys=OFF;
BEGIN TRANSACTION;
CREATE TABLE HostAuth (_id integer primary key autoincrement, protocol text, address text, port integer, flags integer, login text, password text, domain text, accountKey integer);
INSERT INTO "HostAuth" VALUES(1,'pop3','pop.gmail.com',995,13,'mobiseclive@gmail.com','mobisec11',NULL,0);
INSERT INTO "HostAuth" VALUES(2,'smtp','smtp.gmail.com',465,13,'mobiseclive@gmail.com','mobisec11',NULL,0);
COMMIT;
sqlite>
```

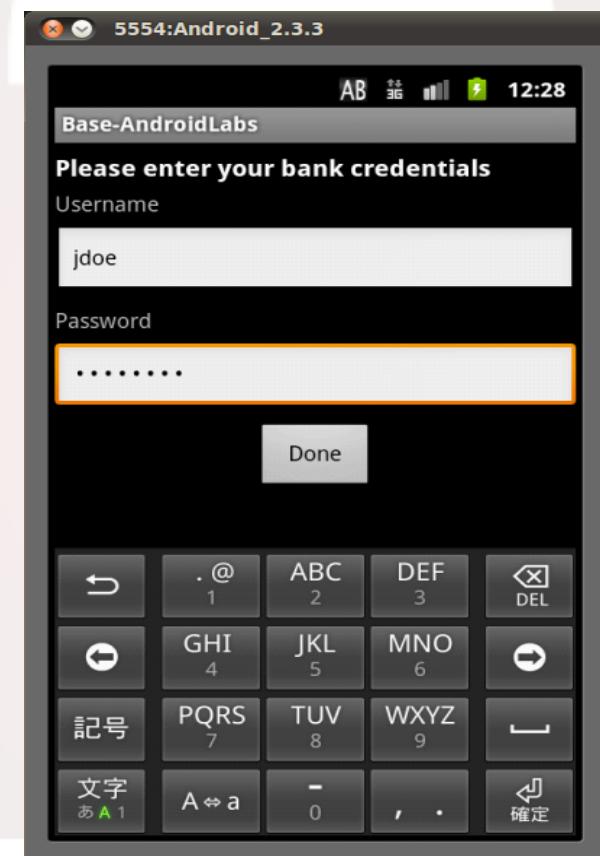
Android Emulators

- Android SDK with Emulators
 - Android 2.1 (DroidBox), 2.3.3, 3.2, 4.03
 - Launch from menu under Emulators & Simulators
 - Launch from command line:
`android-emu.sh <21/233/32/403>`
- Security Compass Lab Server 
 - Simulates very poorly developed "banking" app
 - Already installed on the emulators ☺
 - Launch from menu or commandline:
`sc-labserver-http.sh` or `sc-labserver-https.sh`

Let's Capture Some Packets

- Start emulator manually to capture tcp packets to .cap file
 - emulator –avd Android_2.3.3 –scale 0.75 –tcpdump ~/lab.cap
 - menu/script doesn't include -tcpdump arg
- Start Security Compass Lab Server (http)
- Launch Base-AndroidLabs app
 - Login to the app (jdoe/password)
- Launch Wireshark to view packets
 - wireshark ~/lab.cap

```
Terminal
File Edit View Terminal Help
=====
SecurityCompass Lab Server =====
Launch Android Emulator and use Base-AndroidLabs mobile
app to connect to lab server.
Go to the Security Compass site for info and tutorials:
http://securitycompass.github.com/AndroidLabs/index.html
=====
Starting Lab Server [Ctrl-C to Quit]
Serving HTTP on port 8080
[*] 127.0.0.1 [2012-04-02 00:29:55] "POST http://10.0.2.2:8080/login"
[*] 127.0.0.1 [2012-04-02 00:29:59] "POST http://10.0.2.2:8080/login"
[*] 127.0.0.1 [2012-04-02 00:30:00] "POST http://10.0.2.2:8080/login"
```



lab.cap - Wireshark

File Edit View Go Capture Analyze Statistics Telephony Tools Help

Filter: Expression... Clear Apply

No.	Time	Source	Destination	Protocol	Info
18	335.016103	10.0.2.15	10.0.2.2	TCP	[TCP segment of a reassembled PDU]
19	335.019282	10.0.2.2	10.0.2.15	TCP	http-alt > 55946 [ACK] Seq=1 Ack=231 Win=8760 Len=0
20	335.019812	10.0.2.15	10.0.2.2	HTTP	POST /login HTTP/1.1 (application/x-www-form-urlencoded)
21	335.019838	10.0.2.2	10.0.2.15	TCP	http-alt > 55946 [ACK] Seq=1 Ack=262 Win=8760 Len=0
22	335.077636	10.0.2.2	10.0.2.15	TCP	[TCP segment of a reassembled PDU]
23	335.078			TCP	http-alt > 55946 [ACK] Seq=1 Ack=231 Win=8760 Len=0
24	335.082			HTTP	POST /login HTTP/1.1 (application/x-www-form-urlencoded)

+ Frame 20

+ Ethernet II, Src: RealtekU_12:34:56 (52:54:00:12:34:56), Dst: RealtekU_12:35:02 (52:54:00:12:35:02)

+ Internet Protocol, Src: 10.0.2.15 (10.0.2.15), Dst: 10.0.2.2 (10.0.2.2)

+ Transmission Control Protocol, Src Port: 55946 (55946), Dst Port: http-alt (8080), Seq: 231, Ack: 1, Len: 31

+ [Reassembled TCP Segments (261 bytes): #18(230), #20(31)]

- Hypertext Transfer Protocol

+ POST /login HTTP/1.1\r\nContent-Type: application/x-www-form-urlencoded\r\nUser-Agent: Dalvik/1.4.0 (Linux; U; Android 2.3.3; sdk Build/GRI34)\r\nHost: 10.0.2.2:8080\r\nConnection: Keep-Alive\r\n+ Content-Length: 31\r\nAccept-Encoding: gzip\r\n\r\n

- Line-based text data: application/x-www-form-urlencoded

password=password&username=jdoe

0000 52 54 00 12 35 02 5
0010 00 47 e8 a7 40 00 4
0020 02 02 da 8a 1f 90 4
0030 16 d0 a8 b6 00 00 7
0040 61 73 73 77 6f 72 6
0050 3d 6a 64 6f 65 20

Frame (86 bytes) Reassembled TCP (261 bytes)

Frame (frame), 86 bytes Packets: 37 Displayed: 37 Marked: 0 Profile: Default

Line-based text data: application/x-www-form-urlencoded
password=password&username=jdoe =jdoe

Intercepting Web Requests

- Start emulator manually to route traffic through Burp
 - emulator –avd Android_2.3.3 –scale 0.75 –proxy localhost:8008
- Start AndroidLabs Lab Server ([https](https://))
 - sc-labserver-https.sh
- Configure Burp to intercept and forward traffic
 - Intercept port 8008
 - Forward to port 8443 (AndroidLabs SSL listen port)
 - Support invisible proxying
- Configure AndroidLabs mobile app on emulator
 - IP address of MobiSec (ethx)
 - Enable HTTPS

Mobile App & Burp Settings

The image displays two side-by-side screenshots. On the left is a mobile application interface titled "Base-AndroidLabs". It has four fields: "Bank Service Address" (dropdown), "HTTP Port" (dropdown), "HTTPS Port" (dropdown), and "HTTPS Enabled" (checkbox checked). On the right is the "proxy listeners" configuration screen of Burp Suite free edition v1.4.01. The table shows one listener:

running	port	loopback only	support invis...	redirect	cert
<input checked="" type="checkbox"/>	8008	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	10.10.1....	per-host

Below the table, there are configuration options:

- local listener port: 8008 (with an "update" button)
- listen on loopback interface only
- support invisible proxying for non-proxy-aware clients
- redirect to host: 10.10.1.139
- redirect to port: 8443
- server SSL certificate:
 - use a self-signed certificate
 - generate CA-signed per-host certificates
 - generate a CA-signed certificate with a specific hostname:
- help button

Authenticate & Intercept

Base-AndroidLabs
Please enter your bank credentials

Username: jdoe

Password:

Done

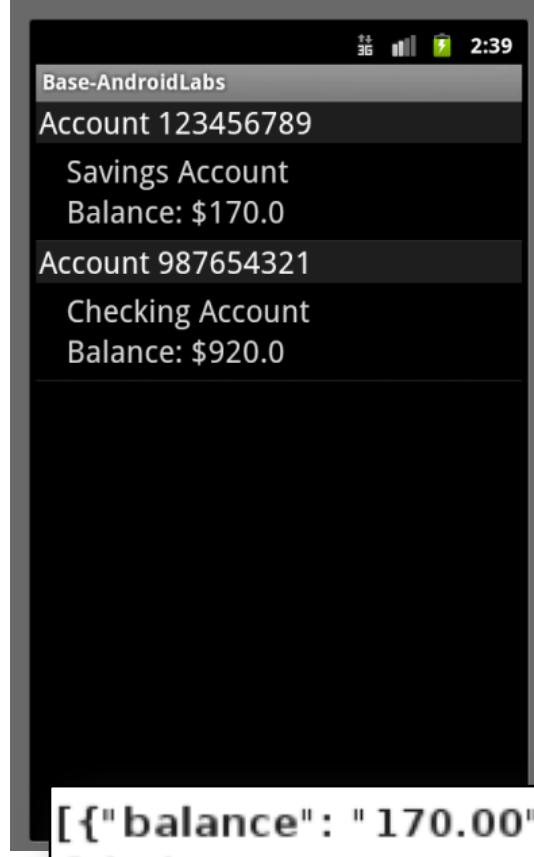
POST /login HTTP/1.1
Content-Type: application/x-www-form-urlencoded
User-Agent: Dalvik/1.4.0 (Linux; U; Android 2.3.3; sdk Build/GRI34)
Host: 10.10.1.139:8443
Connection: Keep-Alive
Content-Length: 31
Accept-Encoding: gzip

password=password&username=jdoe

password=password&username=jdoe|

0 matches

Intercept Account Balances



The screenshot shows the Burp Suite proxy tool. The top menu bar includes "burp", "intruder", "repeater", "window", and "about". Below the menu is a navigation bar with tabs: "intruder", "repeater", "sequencer", "decoder", "comparer", "options", "alerts", "target", "proxy", "spider", "scanner", "intercept" (which is selected), "options", and "history". A filter bar below the navigation bar says "Filter: hiding CSS, image and general binary content".

The main pane displays a list of network requests:

#	host	method	URL	params	mod	sta
1	https://10.10.1.139:8...	POST	/login	<input checked="" type="checkbox"/>	<input type="checkbox"/>	200
2	https://10.10.1.139:8...	POST	/login	<input checked="" type="checkbox"/>	<input type="checkbox"/>	200
3	https://10.10.1.139:8...	POST	/login	<input checked="" type="checkbox"/>	<input type="checkbox"/>	200
4	https://10.10.1.139:8...	GET	/accounts?session_key=iRNak77LuzPmQLi6zb...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	200
5	https://10.10.1.139:8...	GET	/accounts?session key=iRNak77LuzPmQLi6zb...	<input checked="" type="checkbox"/>	<input type="checkbox"/>	200

Below the requests, there are tabs for "request" and "response". The "response" tab is selected, showing the following details:

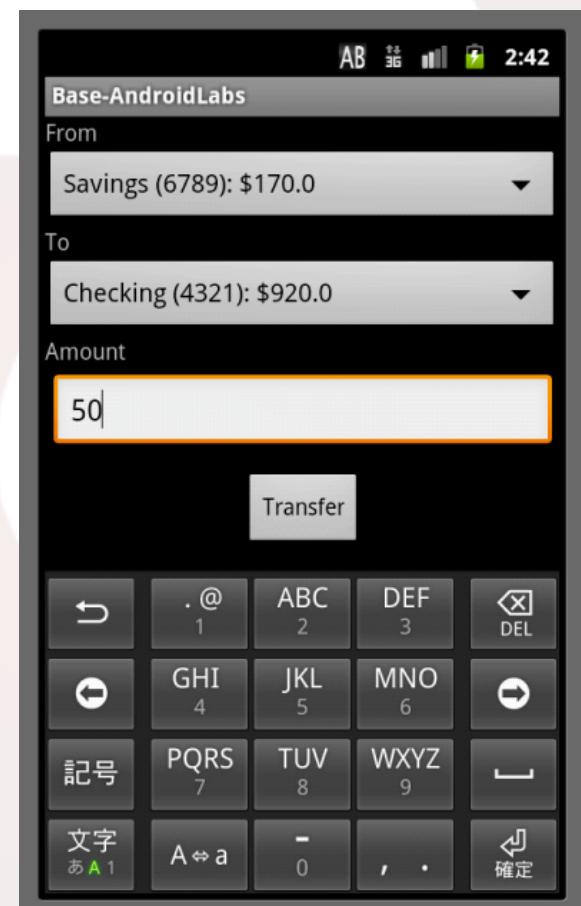
HTTP/1.1 200 OK
Content-Type: text/html; charset=utf-8
Content-Length: 143
Date: Mon, 02 Apr 2012 06:36:54 GMT
Server: mobisec-desktop

[{"balance": "170.00", "type": "savings", "account_number": 123456789},
 {"balance": "920.00", "type": "checking", "account_number": 987654321}]

A large callout box highlights the JSON response body: [{"balance": "170.00", "type": "savings", "account_number": 123456789}, {"balance": "920.00", "type": "checking", "account_number": 987654321}]

Manipulating Web Requests

- Select Transfer from AndroidLabs mobile app
 - Transfer \$50 from Savings to Checking
- Manipulate request in Burp
 - Change "amount=50" to "amount=100"
 - Forward the request to LabServer
- Check the Balances



Change the Amount and Forward

The screenshot shows the Burp Suite interface. The top navigation bar includes 'burp', 'intruder', 'repeater', 'window', and 'about'. Below the navigation is a tab bar with 'repeater', 'sequencer', 'decoder', 'comparer', 'options', 'alerts', 'target', 'proxy', 'spider', 'scanner', and 'intruder'. The 'proxy' tab is selected. Under 'proxy', there are buttons for 'intercept' (which is active), 'options', and 'history'. Below these are buttons for 'forward', 'drop', 'intercept is on', and 'action'. A sub-menu bar at the bottom of the proxy section includes 'raw', 'params', 'headers', and 'hex', with 'raw' being the active tab.

The main content area displays a POST request to `https://10.10.1.139:8443`. The request headers are:

```
POST /transfer?session_key=2eBlwMF8HLDHq%2FWlSKJgZohDSGPNCUe3 HTTP/1.1
Content-Type: application/x-www-form-urlencoded
User-Agent: Dalvik/1.4.0 (Linux; U; Android 2.3.3; sdk Build/GRI34)
Host: 10.10.1.139:8443
Connection: Keep-Alive
Content-Length: 55
Accept-Encoding: gzip
```

The request body contains the following URL-encoded parameters:

```
amount=100.0&from_account=123456789&to_account=987654321
```

To the right of the Burp Suite window, a mobile application screen is displayed. The app shows account information for 'Base-AndroidLabs':

- Account 123456789: Savings Account, Balance: \$70.0
- Account 987654321: Checking Account, Balance: \$1020.0

Using BeEF to Hook Mobile Devices

- Browser Exploitation Framework
- Social Engineer users to click on links
 - No one does that, right? ☺
- Hooked browser appears in BeEF console
 - Displays lots of details of the connected device
 - Commands send javascript to hook browser
 - Browser then responds back to BeEF

iPad hooked by BeEF

The screenshot shows the BeEF Control Panel interface. On the left, there's a sidebar titled "Hooked Browsers" listing "Online Browsers" and "Offline Browsers". Under "Offline Browsers", it shows "172.20.10.7" which further lists "172.20.10.9" and "172.20.10.4". To the right of this is a main panel with tabs for "Getting Started", "Logs", and "172.20.10.4". The "172.20.10.4" tab is active, showing a table of "Category: Browser Hook Initialisation (19 Items)". The table contains several entries with "Initialisation" status. Below this table, specific details about the hooked iPad browser are displayed in a large box:

OS Name: iPad
Browser Name: Safari
Browser Version: 5
Browser UA String: Mozilla/5.0 (iPad; CPU OS 5_1 like Mac OS X) AppleWebKit/534.46 (KHTML, like Gecko) Version/5.1 Mobile/9B176 Safari/7534.48.3
Cookies: BEEFHOOK=iWYxzjnm8Uz6lSSCQn8c5ASwlmz7OBQdSQeEjU8yO2Me1cbLVny3RZxUXVXF1wt77Q8HBCwGANZFq1
Browser Plugins: QuickTime Plug-in YouTube Plug-in
System Platform: iPad
Screen Params: Width: 768, Height: 1024, Colour Depth: 32
Window Size: Width: 981, Height: 644
Java Enabled: No
VBScript Enabled: No
Has Flash: No
Has GoogleGears: No
Has WebSockets: Yes
Has ActiveX: No
Session Cookies: Yes
Persistent Cookies: Yes

Lot's of Meaty Goodness

The screenshot shows a web-based interface for a penetration testing tool, likely Metasploit, running at 127.0.0.1:3000/ui/panel. The interface includes a navigation bar with links for Home, Logout, and other tools like Google and XSSRays.

The main area is divided into several panels:

- Left Panel (Hooked Browsers):** Shows a tree view of hooked browsers. Under "Online Browsers", "172.20.10.7" is expanded. Under "Offline Browsers", "172.20.10.4" is selected.
- Top Center Panel (Module Tree):** Displays a hierarchical list of modules:
 - Browser (17)
 - Chrome Extensions (4)
 - Debug (2)
 - Exploits (5)
 - Host (9)
 - Get Physical Location (selected)
 - Get System Info
 - Hook Default Browser
 - Make Skype Call (Skype)
 - Make Skype Call (Tel)
 - Get Geolocation
 - Get Clipboard
 - Get Protocol Handlers
 - Get Registry Keys
 - IPEC (4)
 - Metasploit (0)
 - Misc (2)
 - Network (7)
 - Persistence (3)
 - Router (3)
- Top Right Panel (Logs):** Shows logs for the target host **172.20.10.4**. Sub-tabs include Details, Logs, Commands (selected), Requester, and XSSRays.
- Bottom Center Panel (Module Results History):** A table with columns id..., date, and label. It displays the message: "The results from executed command modules will be listed here."
- Right Panel (Get Physical Location):** Describes the module's function: "This module will retrieve geolocation information based on the neighboring access points using commands encapsulated within an signed Java Applet." It also lists details: "The details will include:
 - GPS Coordinates details
 - Street Address details"

At the bottom left, there are "Sort by: domain" and "external ip" buttons. At the bottom right, there is an "Execute" button.

What's in MobiSec 1.1

- Updates and added some new tools
 - Metasploit, SET, and Android SDK
 - Ettercap with GUI
 - SQLMap & SQLiteSpy
 - SSLStrip
 - iSniff & dsniff
 - A bunch of FireFox plug-ins
 - Changed the idle-time lockout to 30 mins ☺
 - And more...
- Look for MobiSec v1.1 release next week



OWASP
The Open Web Application Security Project

- The OWASP Mobile Security project was announced in Q3 2010
 - Currently very active
- The project lead is Jack Mannino
 - https://www.owasp.org/index.php/OWASP_Mobile_Security_Project
- It is geared toward providing resources for developers and security teams
 - Tools, guidelines and standards
 - Mobile Security Top Ten

Questions?

- Follow @MobiSecLive on Twitter
- Kevin Johnson & Tony DeLaGrange
Secure Ideas LLC
Web: www.secureideas.net
Email: mobisec@secureideas.net
OR info@secureideas.net
Phone: 904.639.6709
Twitter: @secureideasllc