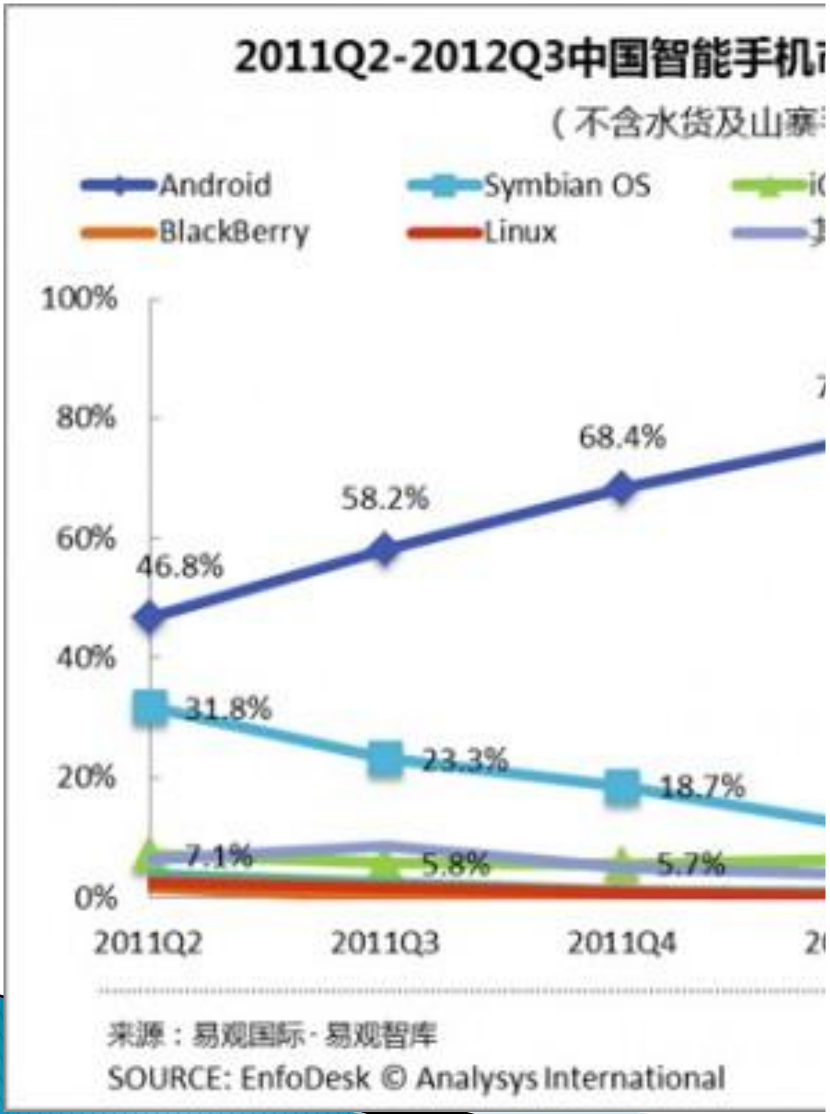
Cloud-based Push-Styled Mobile Botnets: A Case Study of Exploiting the Cloud to Device Messaging Service

S. Zhao, Patrick P. C. Lee, John C.S. Lui, X.H. Guan, X.B. Ma, J. Tao

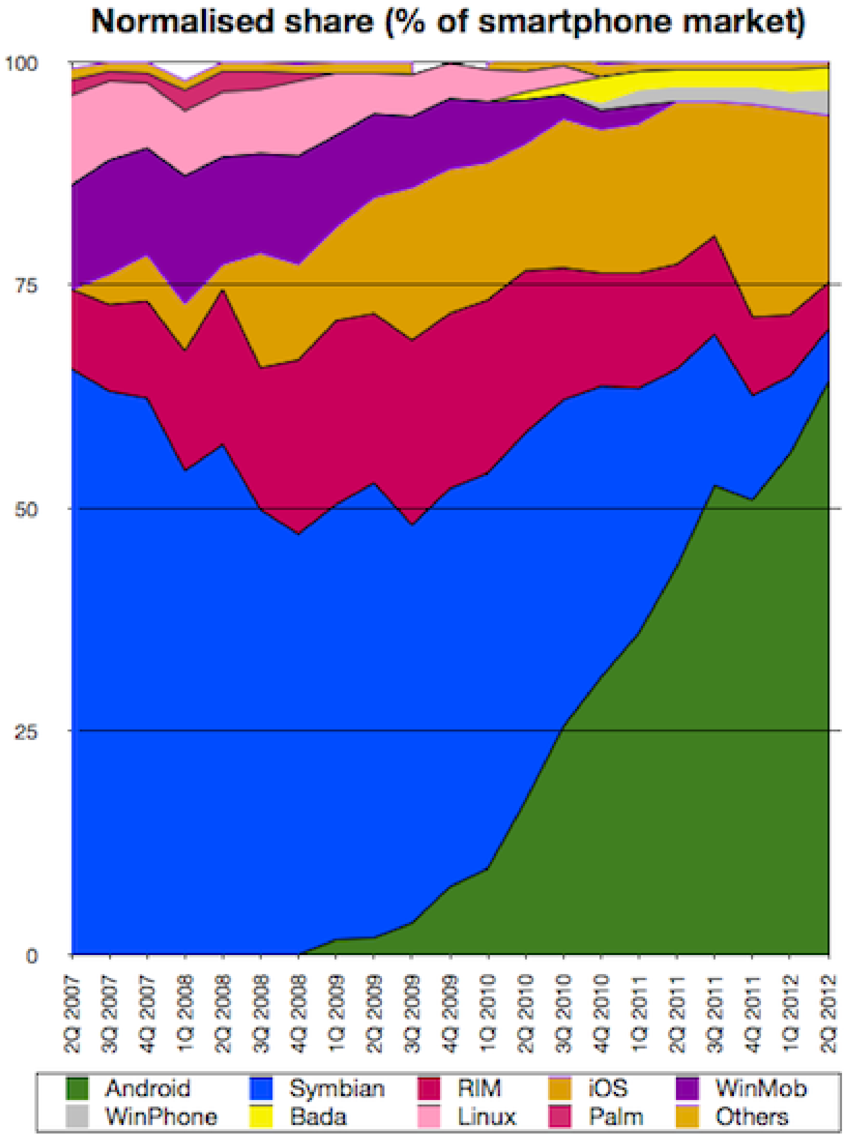
The Chinese University of Hong Kong
XJTU, China

Procceedings of the 28th Annual Computer Security Applications Conference (ACSAC 2012), Orlando, Florida, December 2012.

Motivation



2011Q2-2012Q3中国智能手机
（不含水货及山寨
Android
Symbian OS
BlackBerry
Linux
100%
80%
60%
40%
20%
0%
46.8%
58.2%
68.4%
31.8%
23.3%
18.7%
7.1%
5.8%
5.7%
2011Q2
2011Q3
2011Q4
来源：易观国际·易观智库
SOURCE: EnfoDesk © Analysys International



Normalised share (% of smartphone market)
100
75
50
25
0
2Q 2007
3Q 2007
4Q 2007
1Q 2008
2Q 2008
3Q 2008
4Q 2008
1Q 2009
2Q 2009
3Q 2009
4Q 2009
1Q 2010
2Q 2010
3Q 2010
4Q 2010
1Q 2011
2Q 2011
3Q 2011
4Q 2011
1Q 2012
2Q 2012
Android
Symbian
RIM
iOS
WinMob
WinPhone
Bada
Linux
Palm
Others

My Question

- **How to create your own mobile cloud-based botnet ?**
- **Botnet**
  - ***is a collection of internet-connected computers (devices) whose security defenses have been breached and control ceded to a malicious party. Each such compromised device, known as a "bot", is created when a computer is penetrated by software from a malware distribution. The controller of a botnet is able to direct the activities of these compromised computers through communication channels formed by standards-based network protocols such as IRC (Internet Relay Chat) and HTTP (Hypertext Transfer Protocol)***

Botnet Design Objectives

- **To design a botnet, one needs to consider:**
  - **Scalability:** ***a large population***
  - **Controllability:** ***short response delay for commands***
  - **Stealthiness:** ***hard to be detected***
    - *Keep-Alive Period*
    - *Command Dissemination Period*
    - *Energy Consumption*

Traditional Mobile Botnets

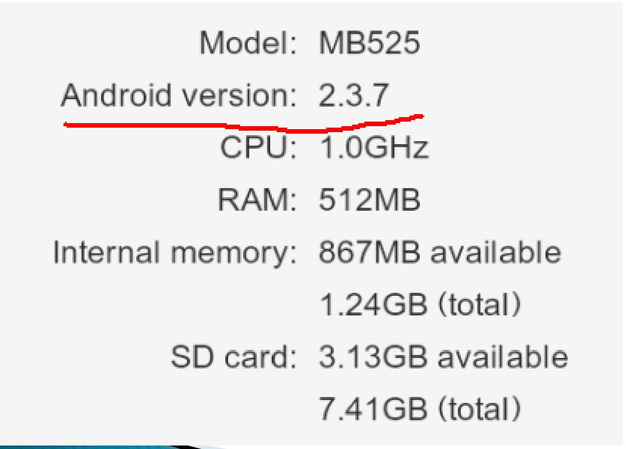
- **SMS Botnet**
  - No keep-alive mechanism between bots and servers
  - High financial cost to send SMS commands
  - Need to use sensitive permissions: Receive_SMS, Send_SMS
- **HTTP Botnet** (facebook, twitter,..etc)
  - Bots pull commands initiatively, which generates unnecessarily traffic for retrieving commands.
  - This may expose C&C server easily
- **TCP Botnet (IRC botnet, etc.)**
  - Connect to C&C server and keep alive
  - This may also expose C&C server easily

Outlines

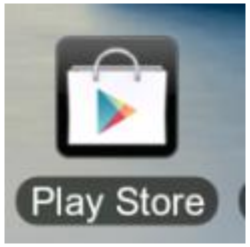
- C2DM/GCM Overview
- C2DM/GCM Botnet
- Evaluation
- Deployment

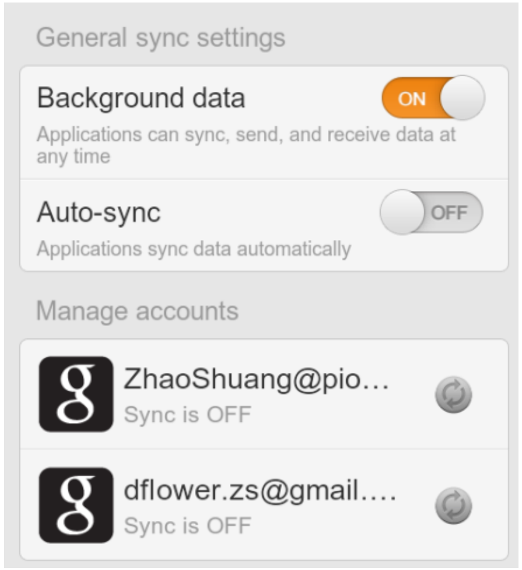
C2DM/GCM Overview

- Why we need such service ?
- Prerequisites
  - Android 2.2+
  - Google Play installed
  - At least one logged-in Google Account and background data enabled



Model:	MB525
Android version:	2.3.7
CPU:	1.0GHz
RAM:	512MB
Internal memory:	867MB available
	1.24GB (total)
SD card:	3.13GB available
	7.41GB (total)





General sync settings

Background data: ON
Applications can sync, send, and receive data at any time

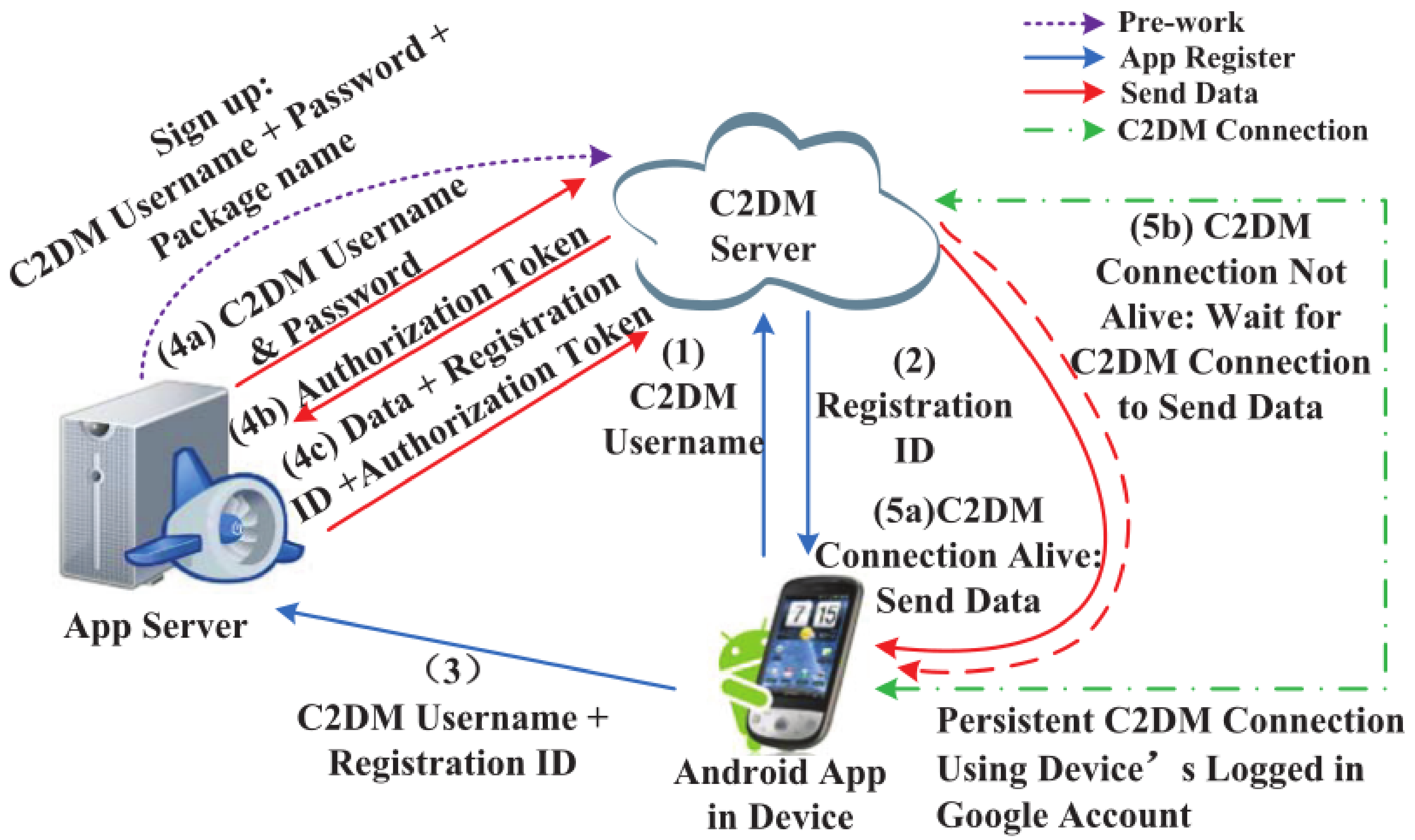
Auto-sync: OFF
Applications sync data automatically

Manage accounts

ZhaoShuang@pio…
Sync is OFF

dflower.zs@gmail.…
Sync is OFF

C2DM/GCM Overview



at

C2DM/GCM Overview

- **C2DM Limitations：**
  - **Unicast Message**
  - **Payload < 1 KB**
  - **Quota: 200,000 notifications per day**
  - **No support for multicast messages**
  - **QPS(Queries Per Second): 0-5, 6-10, 11-100, >100**

C2DM/GCM Overview

- **Important:** C2DM has been officially deprecated as of June 26, 2012. This means that C2DM has stopped accepting new users and quota requests. No new features will be added to C2DM. However, apps using C2DM will continue to work. Existing C2DM developers are encouraged to migrate to the new version of C2DM, called Google Cloud Messaging for Android (GCM). See the C2DM-to-GCM Migration document for more information. Developers must use GCM for new development.

C2DM/GCM Overview

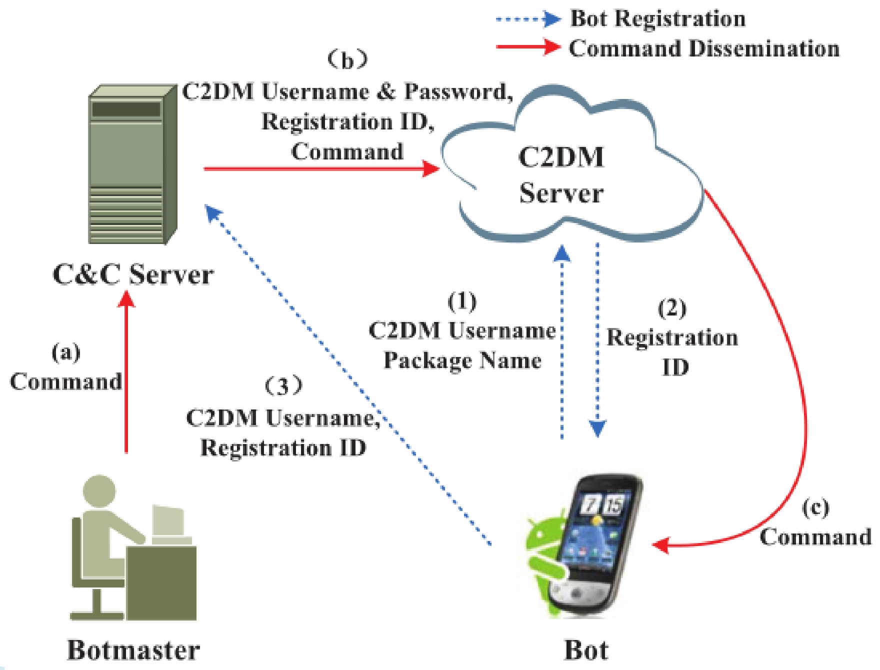
- GCM: New Features to C2DM
  - Simple API Key
  - Sender ID
  - JSON format
  - Multicast messages
  - Multiple senders
  - Time-to-live messages < 4 weeks
  - Message with payload < 4KB
  - Canonical registration ID

Outlines

- C2DM/GCM Overview
- C2DM/GCM Botnet
- Evaluation
- Deployment

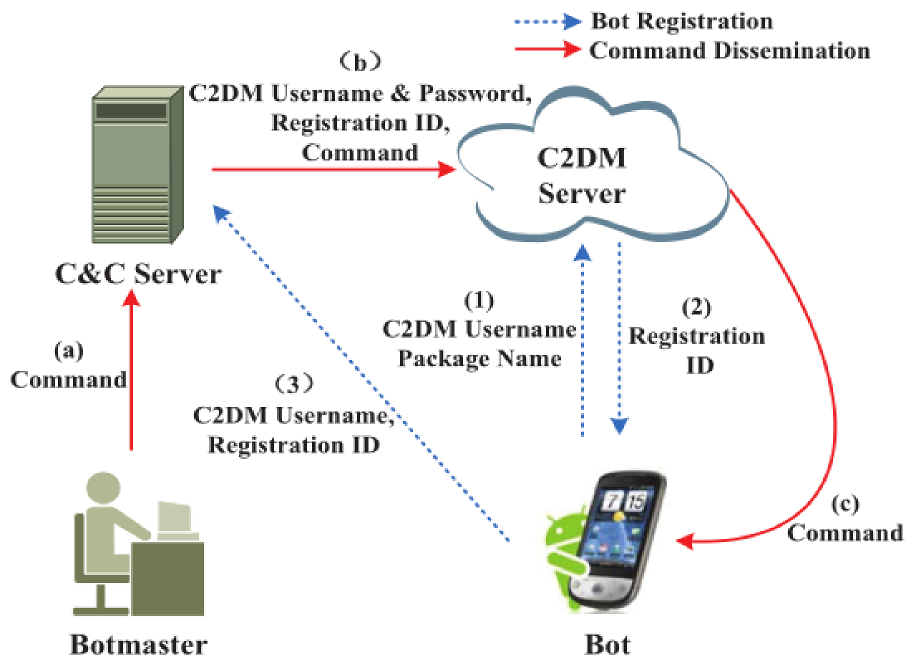
Baseline Architecture

- **Bot Registration:**
  - **(1) Bot sends its C2DM username (hardcoded by botmaster) and package name to C2DM Server to register itself**
  - **(2) C2DM Server returns a unique Registration ID to the bot**
  - **(3)Bot sends the ID to C&C Server**


Bot Registration
Command Dissemination
(b) C2DM Username & Password, Registration ID, Command
C2DM Server
C&C Server
(1) C2DM Username Package Name
(2) Registration ID
(a) Command
(3) C2DM Username, Registration ID
(c) Command
Botmaster
Bot

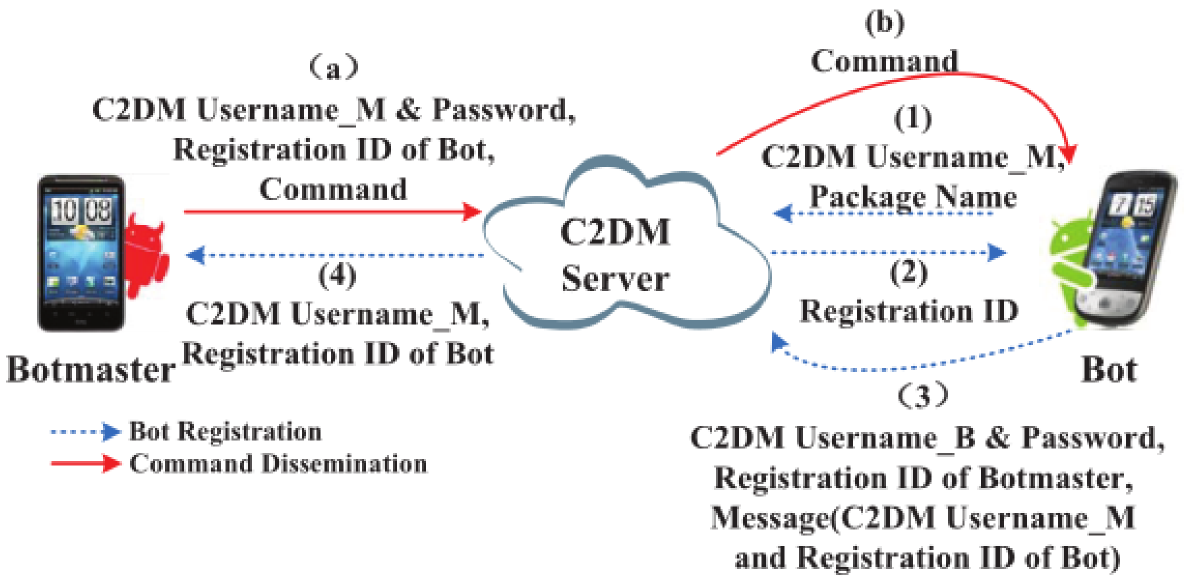
Baseline Architecture

- **Command Dissemination:**
  - **(a) Botmaster sends a command to C&C Server**
  - **(b) C&C Server uses C2DM Username and Password to verify with C2DM Server, then sends a C2DM request containing command and Registration ID of the bot**
  - **(3)C2DM Server forwards the command to Bot**



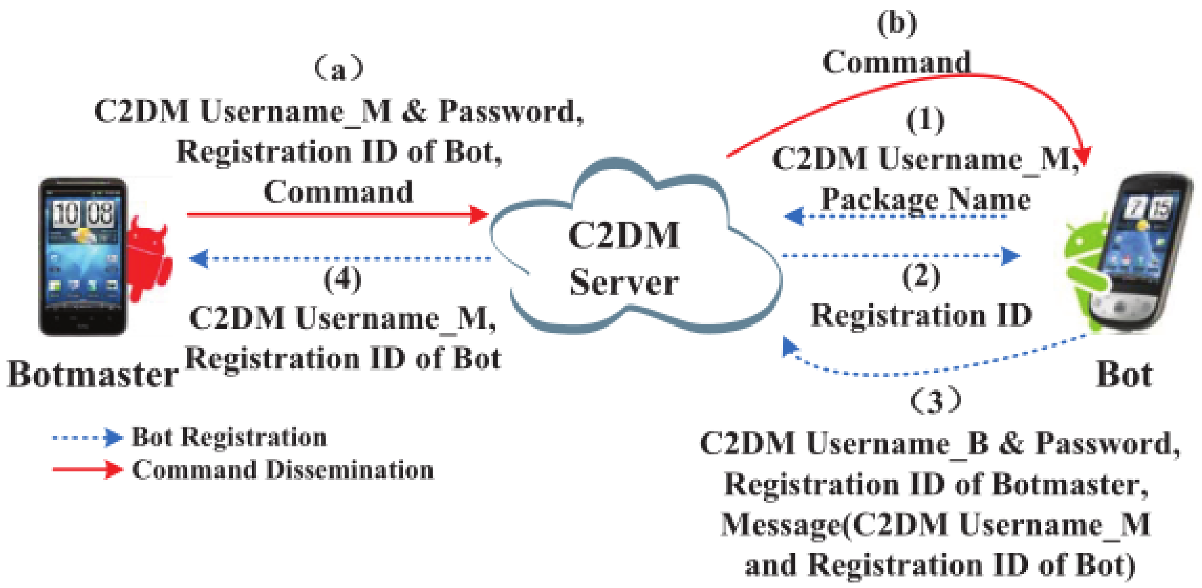
Enhanced Architecture

- **Bot Registration:**
  - **(1) Bot sends its C2DM username(hardcoded by botmaster) and package name to C2DM Server to register itself**
  - **(2) C2DM Server returns a unique Registration ID to the bot**
  - **(3)-(4)Bot sends the ID to C&C Server via C2DM Server**



Enhanced Architecture

- **Command Dissemination**
  - **Similar as that of base architecture**
  - **(a) Botmaster sends a C2DM request containing the command to the C2DM Server**
  - **(b) C2DM Server forwards the command to Bot**



Baseline vs Enhanced Architecture

- **Baseline:**
  - ☹ **Bot sends its Registration ID to C&C Server directly**
  - ☺**Need only one C2DM account**

- **Enhanced:**
  - ☺**Bot sends its Registration ID to C&C Server via C2DM Server**
  - ☹ **Need two C2DM accounts, one is for Bot to send messages to Botmaster, the other is for Botmaster to send messages to Bot**
  - ☹ **The Registration ID of Botmaster may be de-registered or refreshed by Google**

Large Scale Problem(only for C2DM botnet)

- C2DM Limitations:
  - Quota: 200,000 notifications per day
  - QPS < 100 (In order not to draw Google's attention)

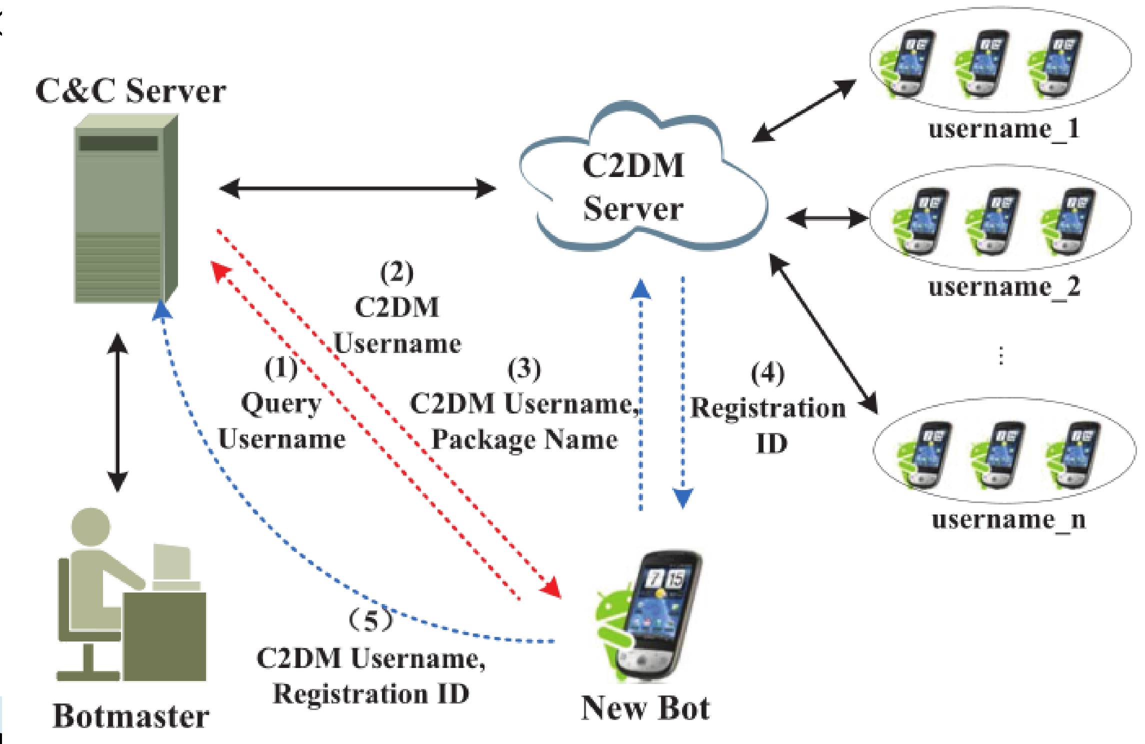
For a small or medium botnet, it is OK.

For a large botnet

- 200,000 notifications maybe not enough.
- It takes to much time to send a command to all of the bots

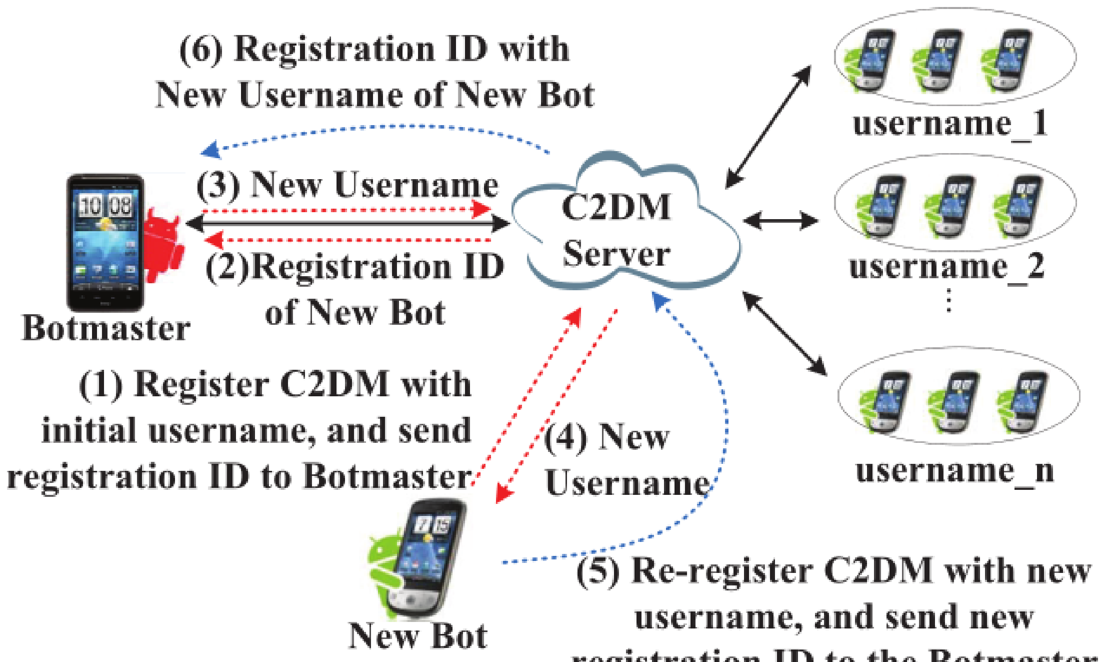
Large Scale Problem(only for C2DM botnet)

- **Use multiple C2DM usernames separate the botnet into sub-nets.**
  - Baseline Architecture: Before registering to C2DM server, the bot communicates with C&C Server to query the C2DM username, and C&C server chooses one from multiple usernames and return to it. Then continue the normal registration stej



Large Scale Problem(only for C2DM botnet)

- **Use multiple C2DM usernames separate the botnet into sub-nets**.
  - **Enhanced Architecture**: The bot firstly registers to C2DM server using hardcoded initial C2DM username, then sends message to Botmaster via C2DM Server to query new C2DM username. Botmaster sends new username to it via C2DM Server. Then continue the normal registration steps.


(6) Registration ID with New Username of New Bot
(3) New Username
(2)Registration ID of New Bot
Botmaster
C2DM Server
username_1
username_2
username_n
(1) Register C2DM with initial username, and send registration ID to Botmaster
(4) New Username
New Bot
(5) Re-register C2DM with new username, and send new registration ID to the Botmaster

SPoF: Account Un-registration

- The C2DM service used by botnet is banned by Google
  - It is more likely that the C2DM username is banned (C2DM account is un-registered)
- In the Enhanced Architecture
  - the Registration ID of the botmaster is unregistered or refreshed

SPoF: Account Un-registration

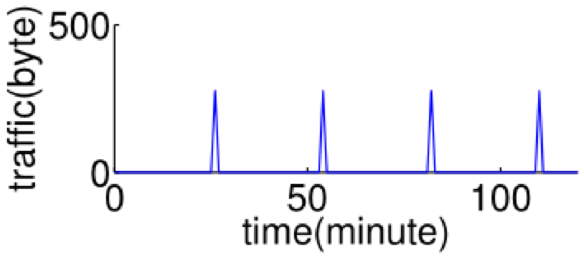
- **Multiple-usernames strategy**
  - **Setup one or several backup C&C servers**
  - **If a bot has not received any C2DM messages for a long time, it will communicate with backup C&C server to query the new C2DM username**
  - **In the enhanced architecture, the bot will also query the new registration ID of the botmaster**

Outlines

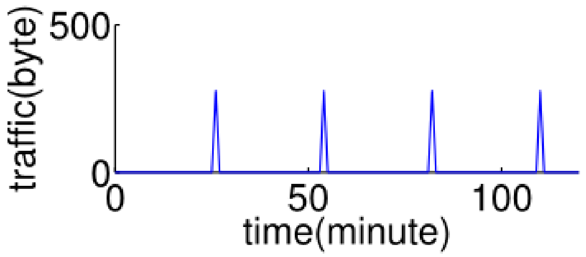
- C2DM/GCM Overview
- C2DM/GCM Botnet
- Evaluation
- Deployment

Stealthiness

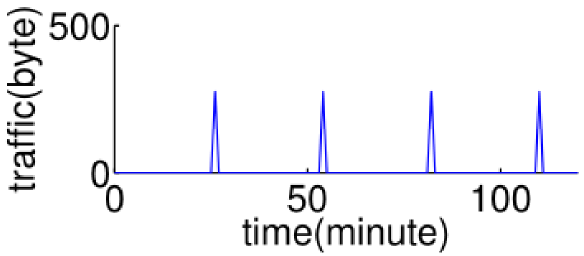
- **Stealthiness of Heartbeat Traffic**
  - **C2DM connection is shared by all C2DM applications(including C2DM**



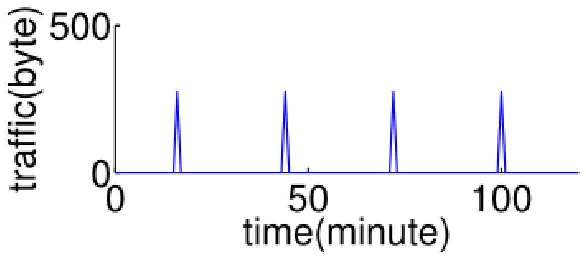
(a) Device with C2DM Bot, Gmail and Google Maps



(b) Clean device with Gmail, Google Maps



(c) Device with C2DM Bot, Gmail, Google Maps, Facebook, LINE and Instagram

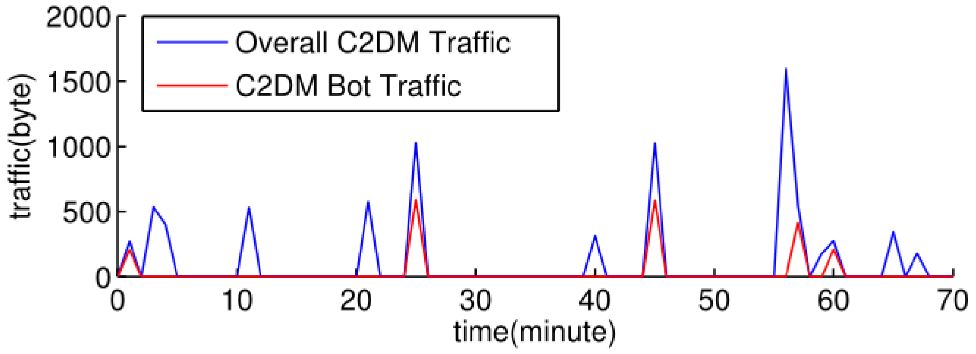


(d) Clean device with Gmail, Google Maps, Facebook, LINE and Instagram

Stealthiness

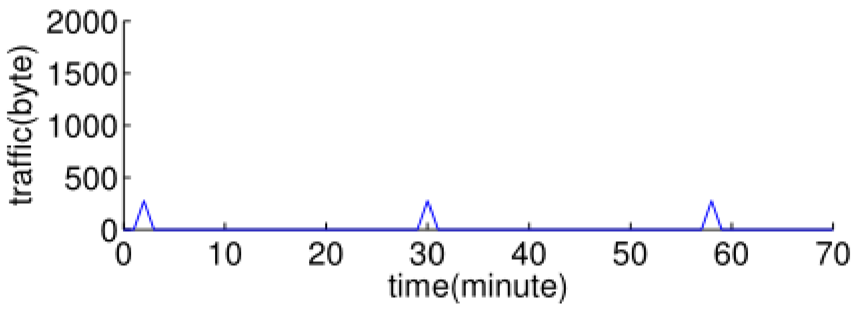
- **Stealthiness of Command Dissemination**
  - In many Android phones, there are some pre-installed or other popular applications which use C2DM service.
  - Botmaster won't send many commands to bots per day.
- **EXPERIMENT(1 hour)**
  - install a C2DM bot into an phone which has **Gmail** and **Google Maps** in it.
  - Send some commands to the bot
  - Meanwhile, some Gmail and Google Map notifications arrives, too.
  - The traffic of C2DM Bot occupies

  less than 20%

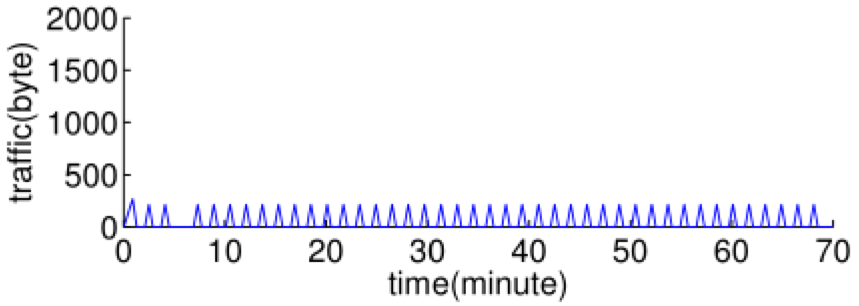


Resource Consumption

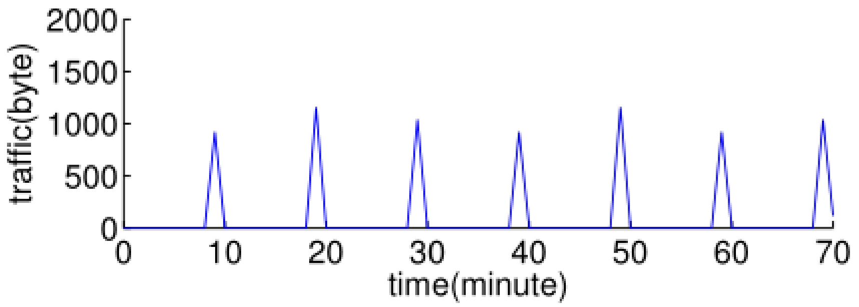
- **Bandwidth Consumption**
  - **C2DM Bot**
    - 900 bytes/hour
  - **IRC Bot (interval 90s)**
    - 5760 bytes/hour
  - **HTTP (interval: 10min):**
    - 7200 bytes/hour



(a) C2DM traffic



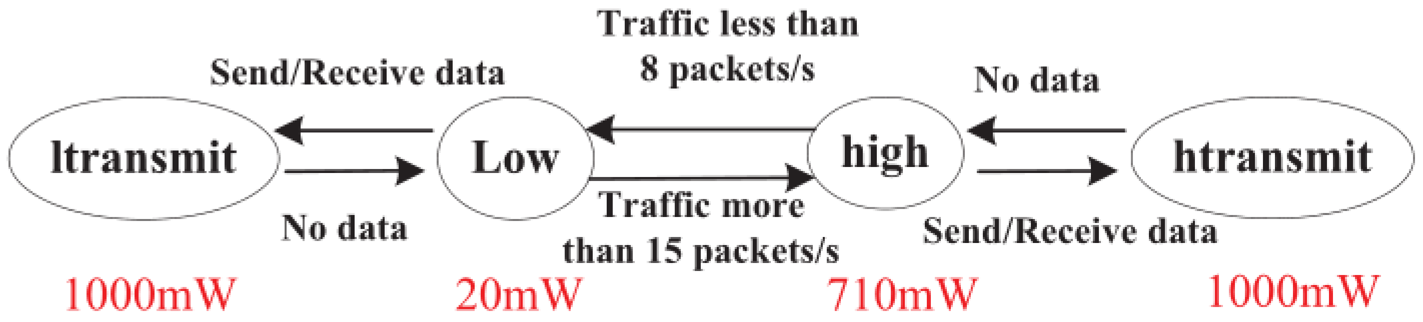
(b) IRC traffic with ping-pong interval being 90s



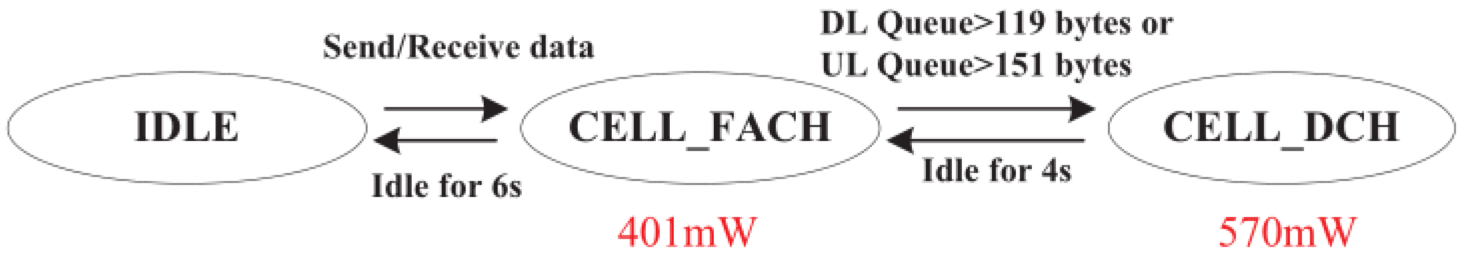
(c) HTTP traffic with pull interval being 10min

Power Consumption

- **WiFi Power States of HTC Dream**

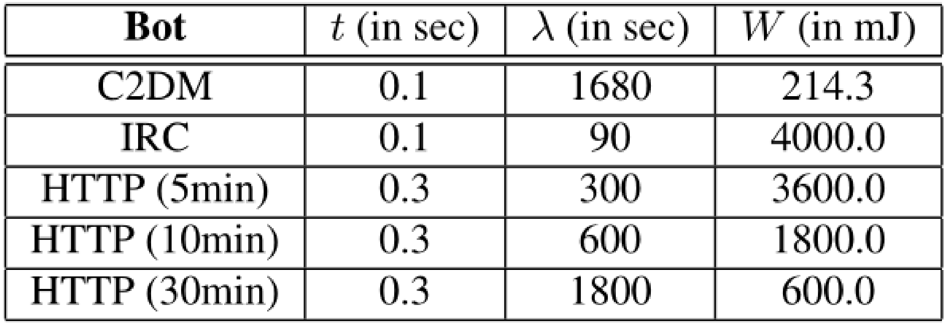


- **3G Power States of HTC Dream**



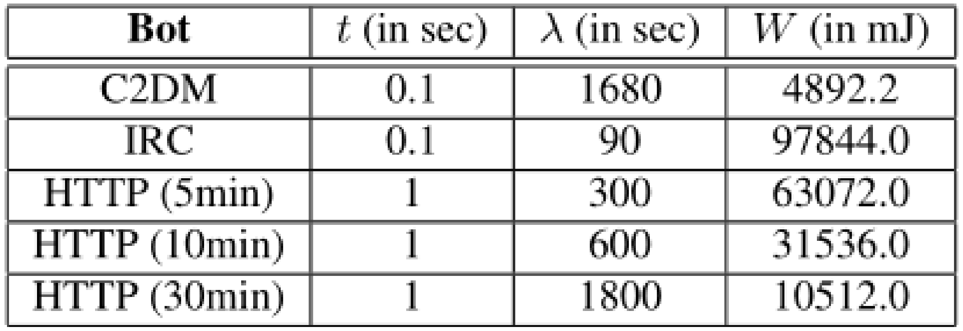
Power Consumption

- **WiFi Power Consumption**



Bot	t (in sec)	λ (in sec)	W (in mJ)
C2DM	0.1	1680	214.3
IRC	0.1	90	4000.0
HTTP (5min)	0.3	300	3600.0
HTTP (10min)	0.3	600	1800.0
HTTP (30min)	0.3	1800	600.0

- **3G Power Consumption**



Bot	t (in sec)	λ (in sec)	W (in mJ)
C2DM	0.1	1680	4892.2
IRC	0.1	90	97844.0
HTTP (5min)	1	300	63072.0
HTTP (10min)	1	600	31536.0
HTTP (30min)	1	1800	10512.0

Controllability

- **Time Delay = Time Delay for Sending Command**

  + Time Delay for Receiving Command

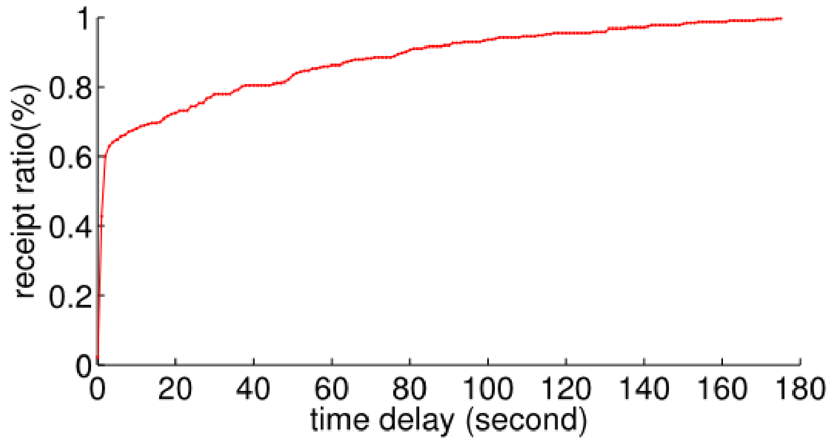
Controllability

- **Time Delay for Sending Commands**
- **C2DM Botnet**:
  - Send commands at a random speed of 0-100/sec
  - Send 10,000 commands will need about 200 seconds.
- **GCM Botnet:**
  - The time delay for sending a multicast command is nearly 0

Controllability

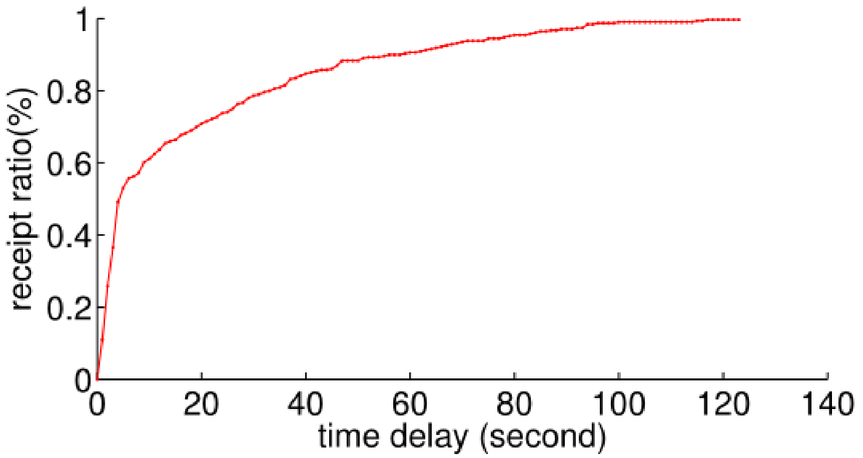
- Time Delay for Receiving Commands (C2DM)
  - **WiFi**

Avg: 21.7s



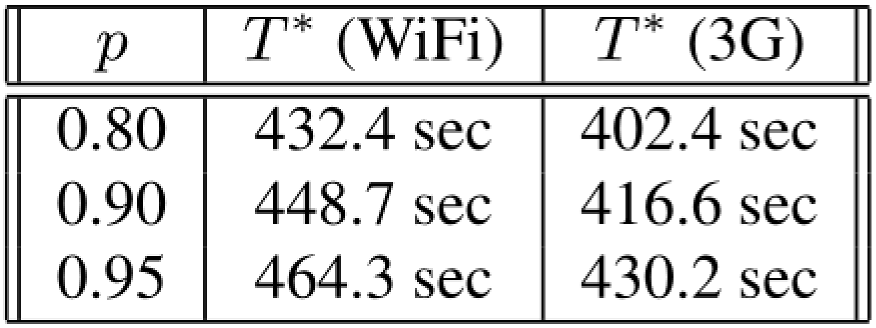
  - **3G**

Avg: 18.9s



Controllability

- p: Probability for all 10,000 bots have received the command
- T*: Time delay for sending command + receiving command



p	T^* (WiFi)	T^* (3G)
0.80	432.4 sec	402.4 sec
0.90	448.7 sec	416.6 sec
0.95	464.3 sec	430.2 sec

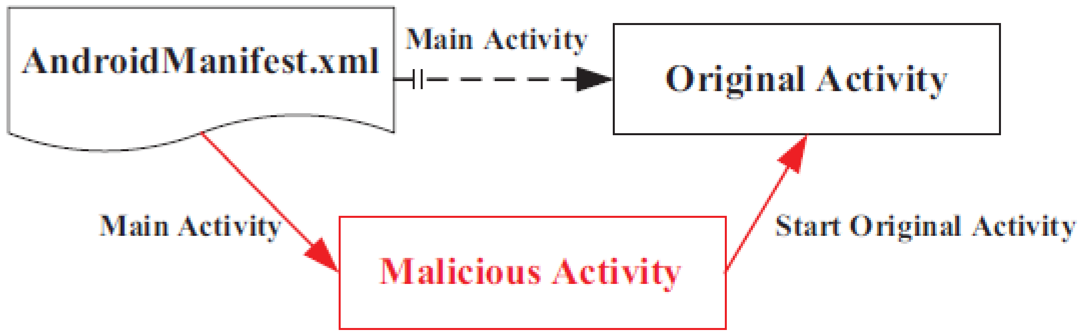
C2DM:less than 8 minutes to reach most of the 10,000 bots.

GCM: less than 5 minutes to reach most of the 10,000 bots

Outlines

- C2DM/GCM Overview
- C2DM/GCM Botnet
- Evaluation
- Deployment

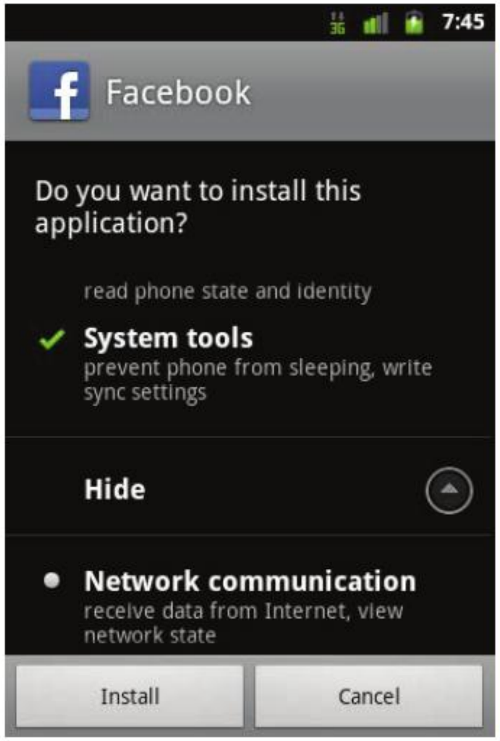
APK Injection and Repackaging



- 1. Disassemble the target app into smali code, modify the main activity in AndroidManifest.xml to malicious activity (the malicious activity will start the original activity after doing bad things)
- 2. Disassemble the malicious app into smali code, copy the smali files into the disassembled folder of the targeted app.
- 3. Repackage and resign the target app.

APK Injection and Repackaging

- Upload the app which is injected with malware to official Android Market or third-party markets
- DEMO: Inject C2DM bot into Facebook for Android


7:45

Facebook

Do you want to install this application?

read phone state and identity

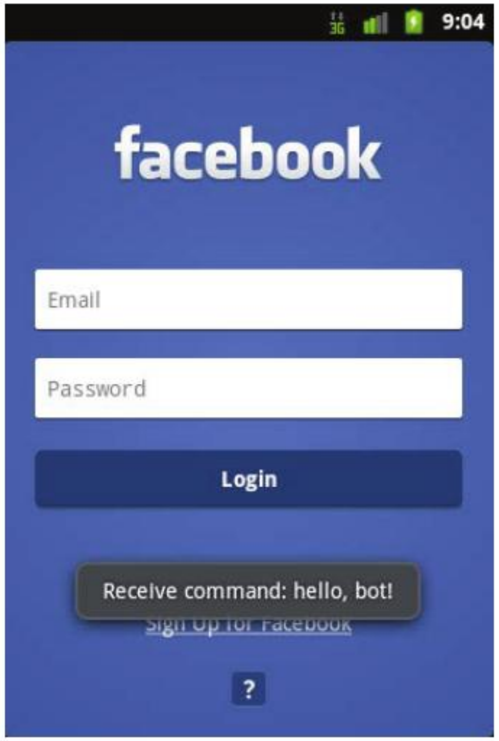
✓ **System tools**
prevent phone from sleeping, write sync settings

Hide

● **Network communication**
receive data from Internet, view network state

Install | Cancel

(a) Installation


9:04

facebook

Email

Password

Login

Receive command: hello, bot!

Sign Up for Facebook

?

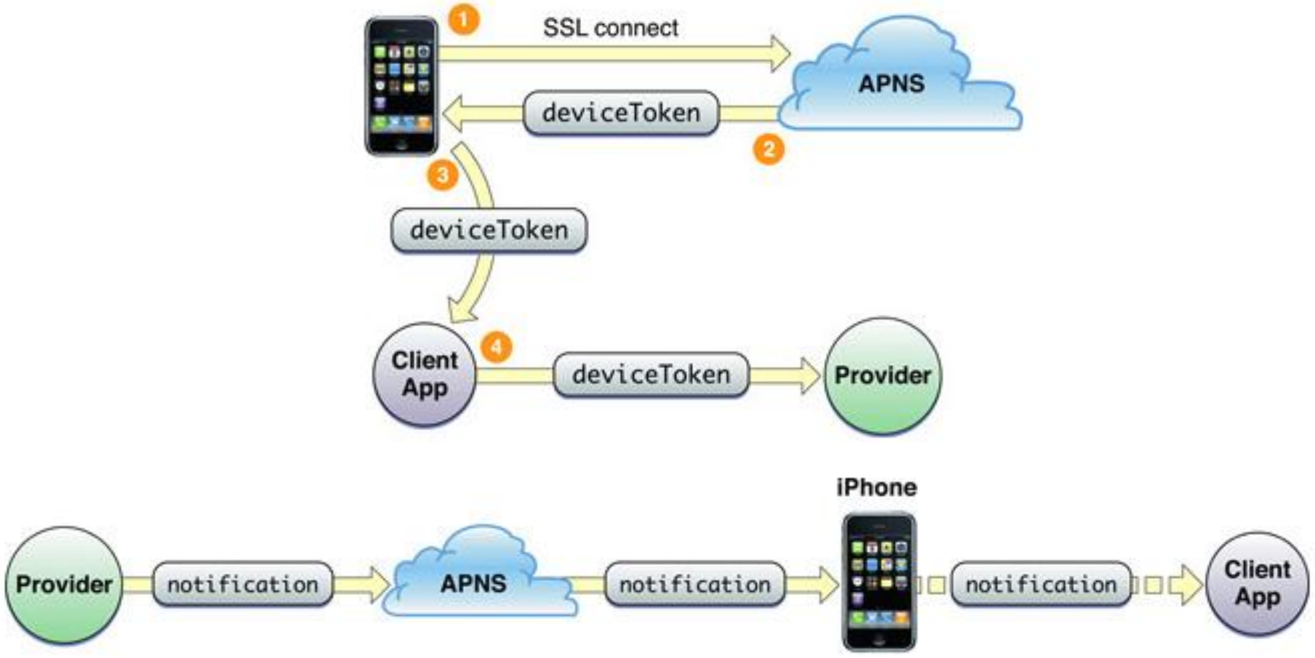
(b) Startup

Push Service for Other Smartphones

- iOS
- Windows
- RIM
- Symbian
- …

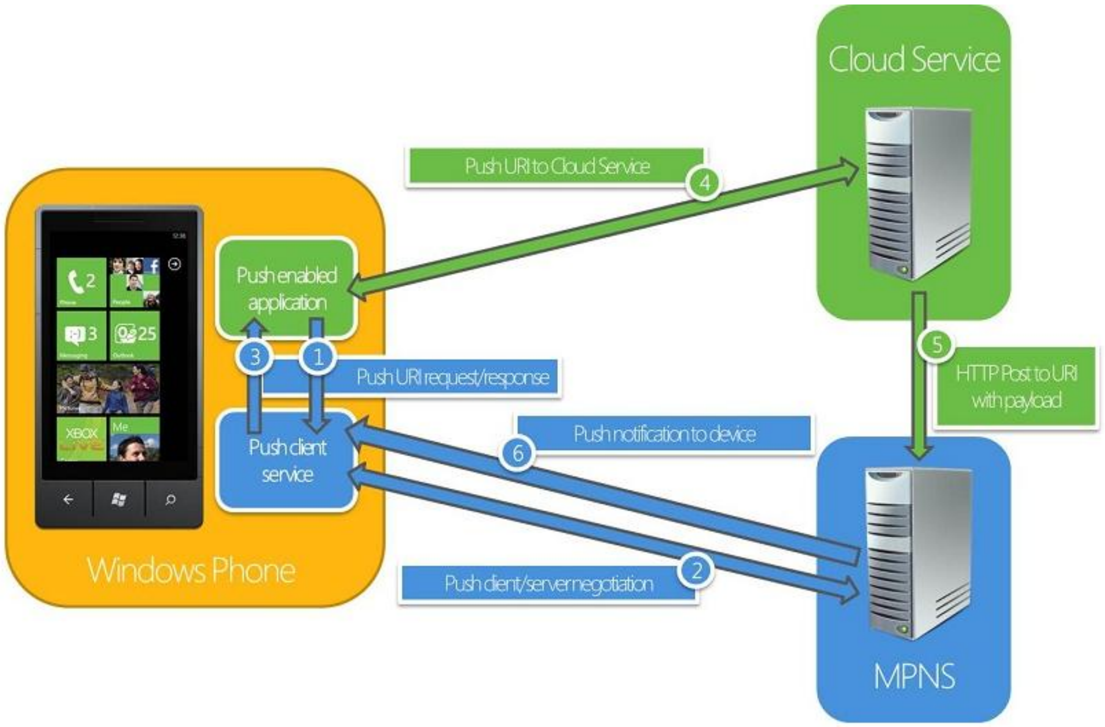
Push Service for iOS(Apple)

- APNS(Apple Push Notification Service)
  - Payload < 256 bytes
  - Quota: Unlimited



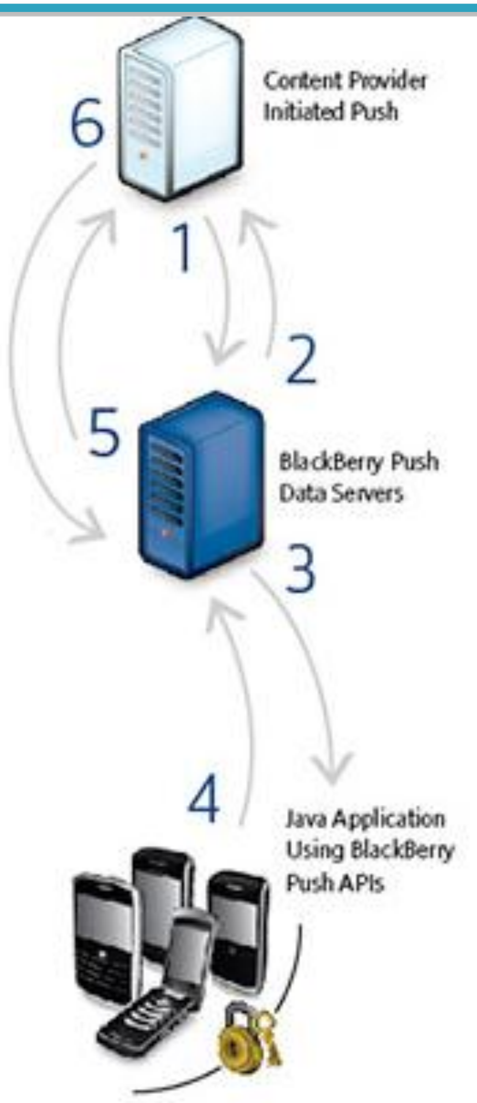
Push Service for Windows(Microsoft)

- MPNS(Microsoft Push Notification Service)
  - Header < 1KB, Payload < 3KB
  - Quota
    - Non-Authorized Web Service: 500 notifications per subscription per day
    - Authorized Web Service: No Limitation


Cloud Service
Push URI to Cloud Service
4
Push enabled application
3
1
Push URI request/response
5
HTTP Post to URI with payload
Push client service
Push notification to device
6
Windows Phone
2
Push client/server negotiation
MPNS

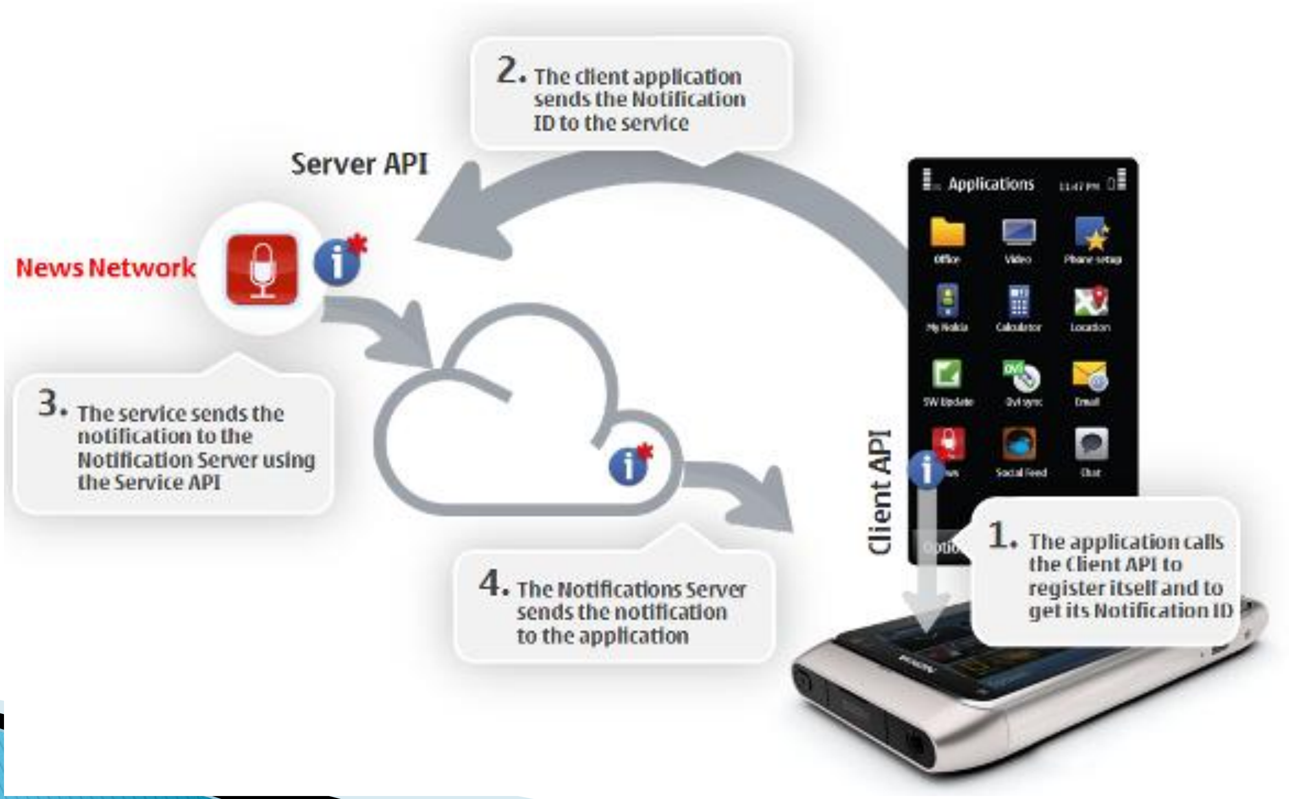
Push Service for RIM(BlackBerry)

- BPS: BlackBerry Push Service(Essentials/Plus)
  - Payload < 8KB
  - Quota
    - Plus(Step 1 – 6): 100,000 messages per day for free, segmented pricing for above 100,000 messages
    - Essentials(Step 1 and 3): Free at all levels


Content Provider Initiated Push
6
1
2
5
BlackBerry Push Data Servers
3
4
Java Application Using BlackBerry Push APIs

Push Service for Symbian & Meego(Nokia)

- NNI(Nokia Notifications API, AKA, Ovi Notifications Service):
  - Payload < 1.5KB
  - Quota: Unlimited


2. The client application sends the Notification ID to the service
Server API
News Network
3. The service sends the notification to the Notification Server using the Service API
4. The Notifications Server sends the notification to the application
Client API
1. The application calls the Client API to register itself and to get its Notification ID
Applications

Conclusion

- Android, though popular, has many security issues
- Did we learn our lesson yet?
- Given the financial importance of the market, we need
  - Better detection
  - Better forensic
  - Better architecture
- The journey just started, where is our destination?

Thanks