GEORGIA TECH INFORMATION SECURITY CENTER

*Safeguarding Digital Infomation Through Innovative Research and Education*

# On the Feasibility of Large-Scale Infections of iOS Devices

**Tielei Wang**, Yeongjin Jang, Yizheng Chen, Simon Chung, Billy Lau, and Wenke Lee

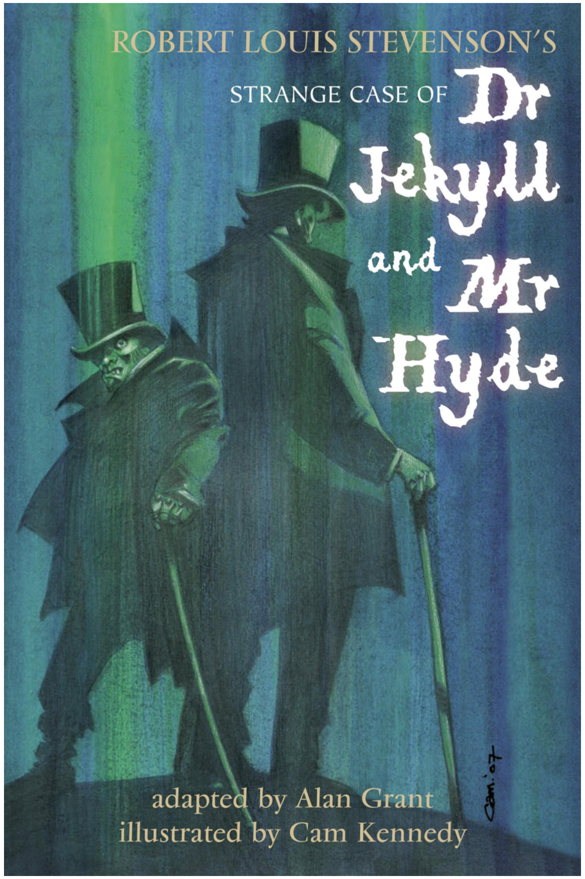
Georgia Institute of Technology

# Outline

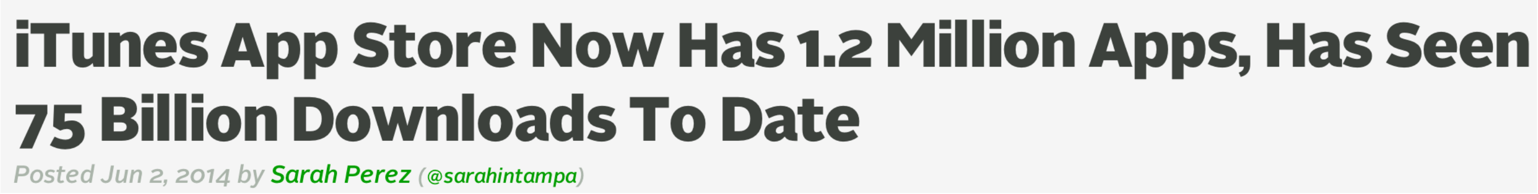
- Background and Motivation
- Security Risks in Connecting iOS Devices to Compromised PCs
- Measurement Results
- Conclusion

# Jekyll on iOS [USENIX Security'13]

- We created  a seemingly benign app named Jekyll and published it on the Apple App Store
- Jekyll can be instructed to carry out malicious tasks by reordering and rearranging benign functionalities
- Conclusion: Apple's vetting process cannot prevent malicious apps



# Key Limitation of Jekyll Apps



**iTunes App Store Now Has 1.2 Million Apps, Has Seen 75 Billion Downloads To Date**

Posted Jun 2, 2014 by Sarah Perez (@sarahintampa)

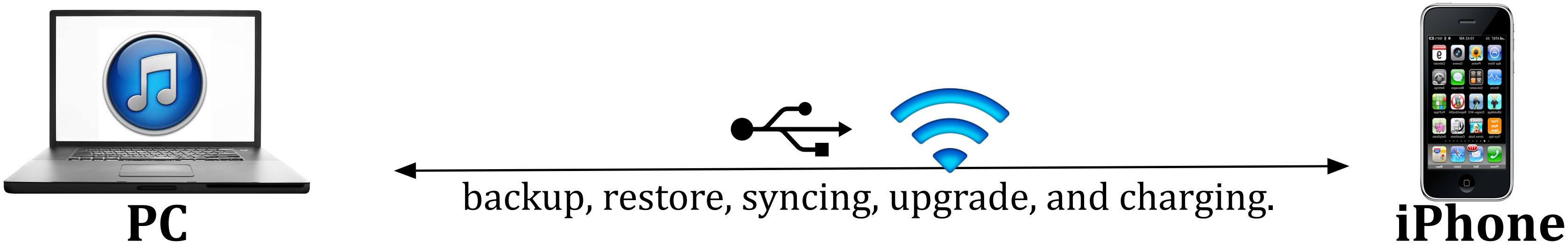
- Jekyll apps did not get a lot of downloads
  - Malicious apps, like any other apps, have the challenge of attracting attention from users
  - Such apps can only affect a limited number of iOS users who *accidentally* download and run them

# Motivation

Is it feasible to proactively deliver malicious apps to iOS devices at scale?

# Attack Vector

- We reviewed the iOS app distribution channels, and confirmed that PCs become a new attack vector to iOS devices



PC

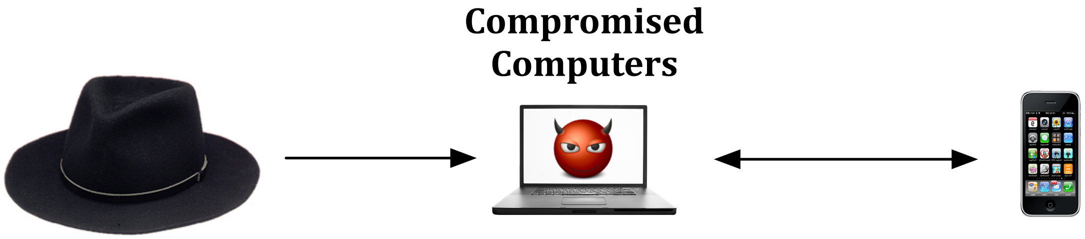
backup, restore, syncing, upgrade, and charging.

iPhone

- Install or remove apps
- Access data in mobile devices

# Contributions

- Demonstrated security risks in connecting iOS devices to compromised computers



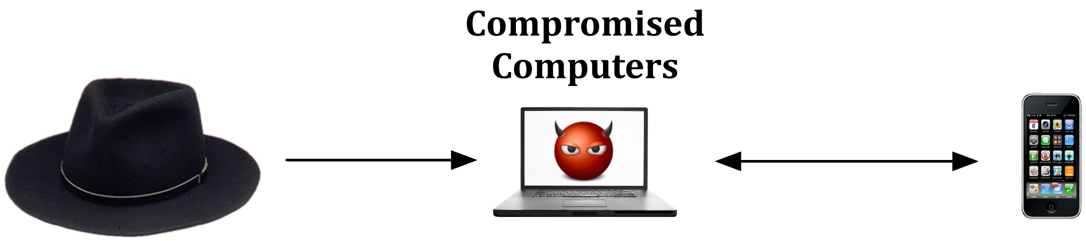
- Measured the overlap between iOS devices and compromised Windows PCs

# Outline

- Background and Motivation
- Security Risks in Connecting iOS Devices to Compromised PCs
- Measurement Results
- Conclusion

# Attack I: Delivery of Jekyll Apps

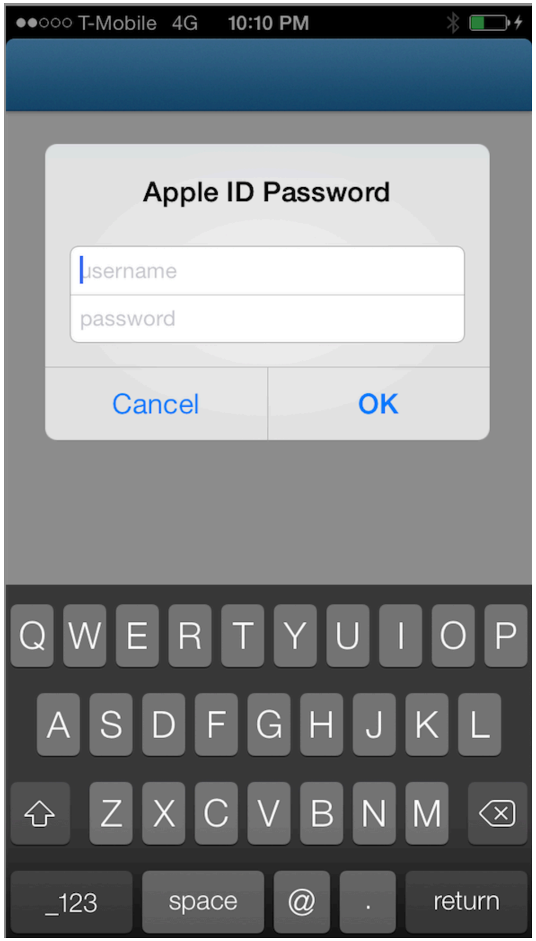
- Intuition: Attacker downloads a Jekyll app, and then injects the Jekyll app to plugged-in iOS devices



- Challenge: Digital Rights Management (DRM) technology in iOS prevents users from sharing apps among arbitrary iOS devices

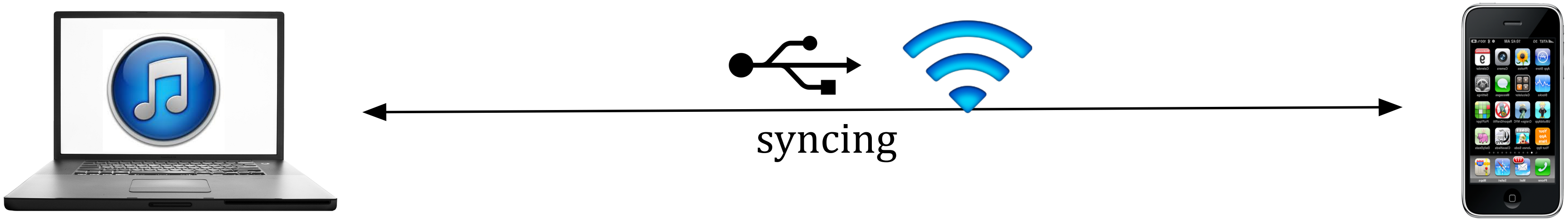
# FairPlay DRM

- Downloading apps from the App Store requires an Apple ID
- User attempting to run an app downloaded by a different Apple ID on his iOS device needs to first enter the correct Apple ID and password



*How to bypass the DRM protection?*

# Key Observation: iTunes Syncing



iTunes with **Apple ID A**

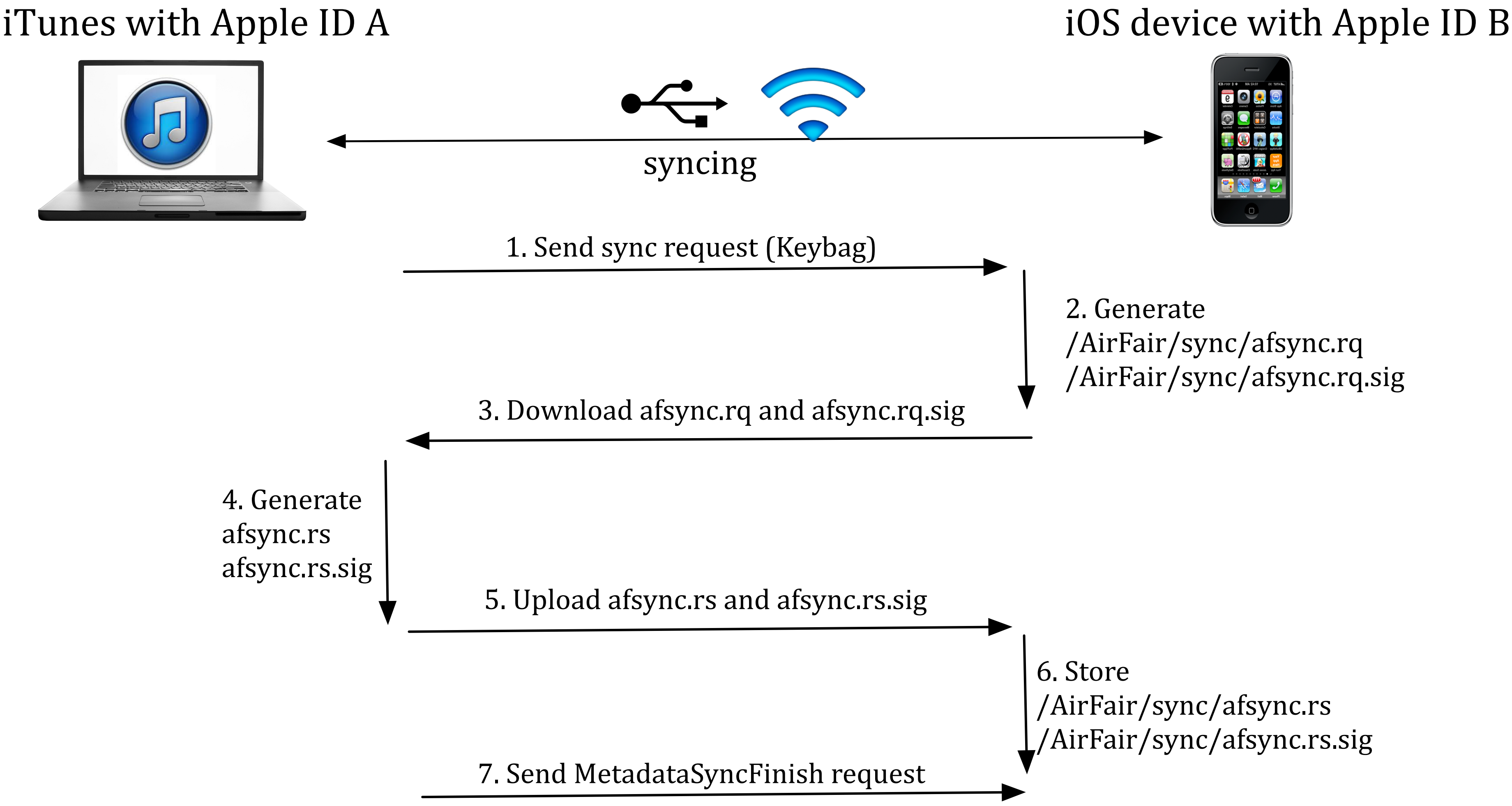
iOS device with Apple ID B

syncing

iTunes can authorize an iOS device with a different Apple ID to run its apps

After syncing, apps purchased by Apple ID A can also run on the iOS device

# The Detailed Process



iTunes with Apple ID A

iOS device with Apple ID B

syncing

1. Send sync request (Keybag)

2. Generate
/AirFair/sync/afsync.rq
/AirFair/sync/afsync.rq.sig

3. Download afsync.rq and afsync.rq.sig

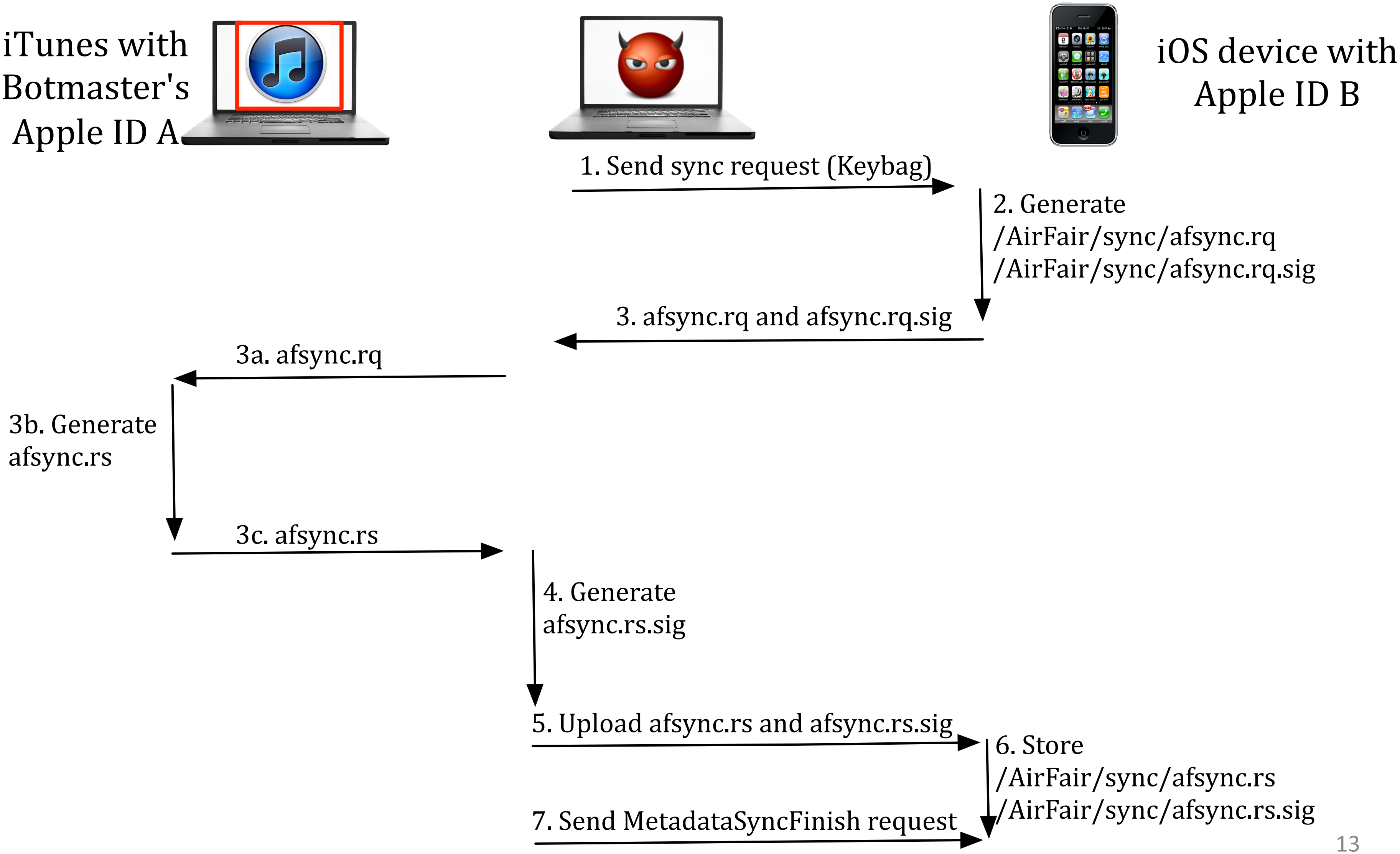
4. Generate
afsync.rs
afsync.rs.sig

5. Upload afsync.rs and afsync.rs.sig

6. Store
/AirFair/sync/afsync.rs
/AirFair/sync/afsync.rs.sig

7. Send MetadataSyncFinish request

# The Man-in-the-Middle syncing



iTunes with Botmaster's Apple ID A

iOS device with Apple ID B

1. Send sync request (Keybag)

2. Generate
/AirFair/sync/afsync.rq
/AirFair/sync/afsync.rq.sig

3. afsync.rq and afsync.rq.sig

3a. afsync.rq

3b. Generate
afsync.rs

3c. afsync.rs

4. Generate
afsync.rs.sig

5. Upload afsync.rs and afsync.rs.sig

6. Store
/AirFair/sync/afsync.rs
/AirFair/sync/afsync.rs.sig

7. Send MetadataSyncFinish request

# Attack I Summary

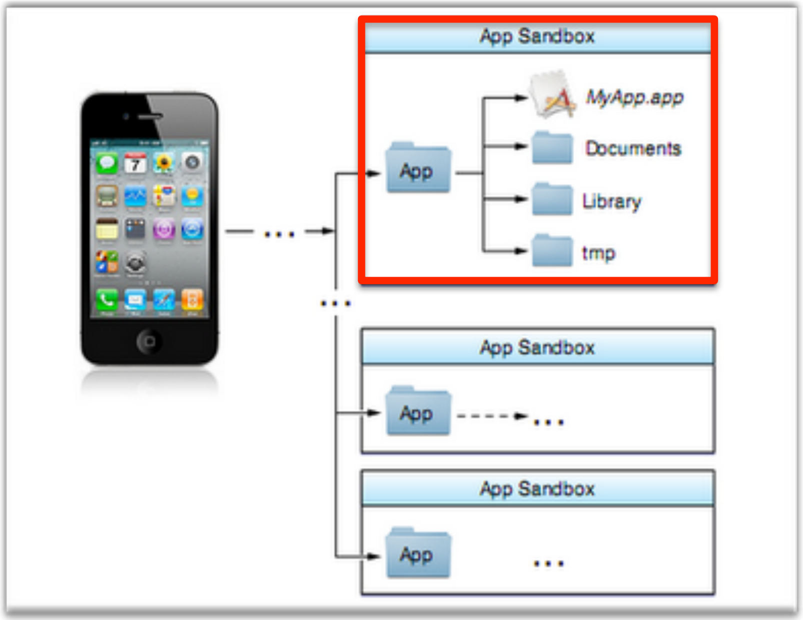
- Attackers can remotely instruct an already compromised computer to install apps on a connected iOS device, completely bypassing DRM checks
- Even if an app has been removed from the App Store, attackers can still distribute their own copies to iOS users
- Although Apple has absolute control of the App Store, attackers can leverage MitM to build a covert distribution channel of iOS apps

# Attack II: Delivery of Attacker-Signed Apps

- Apple allows developers to test their apps on iOS devices through a process called device provisioning
- A compromised computer can be instructed to provision a plugged-in iOS device without user knowledge
- It allows the computer to further install any app signed by the attacker

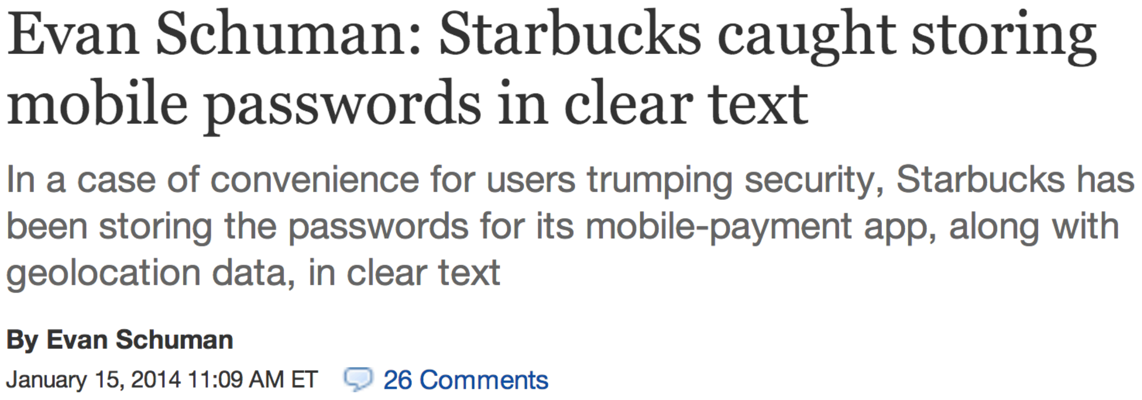
# Attack III: Stealing Credentials

- iOS Sandboxing
  - Each app has a unique home directory for its file
  - Apps are restricted from accessing files stored by other apps or from making changes to the device



# Attack III: Stealing Credentials

- Many iOS apps store credentials in plaintext, because the developers presume that the iOS sandbox can prevent other apps from accessing files in their apps' home directories



## Evan Schuman: Starbucks caught storing mobile passwords in clear text

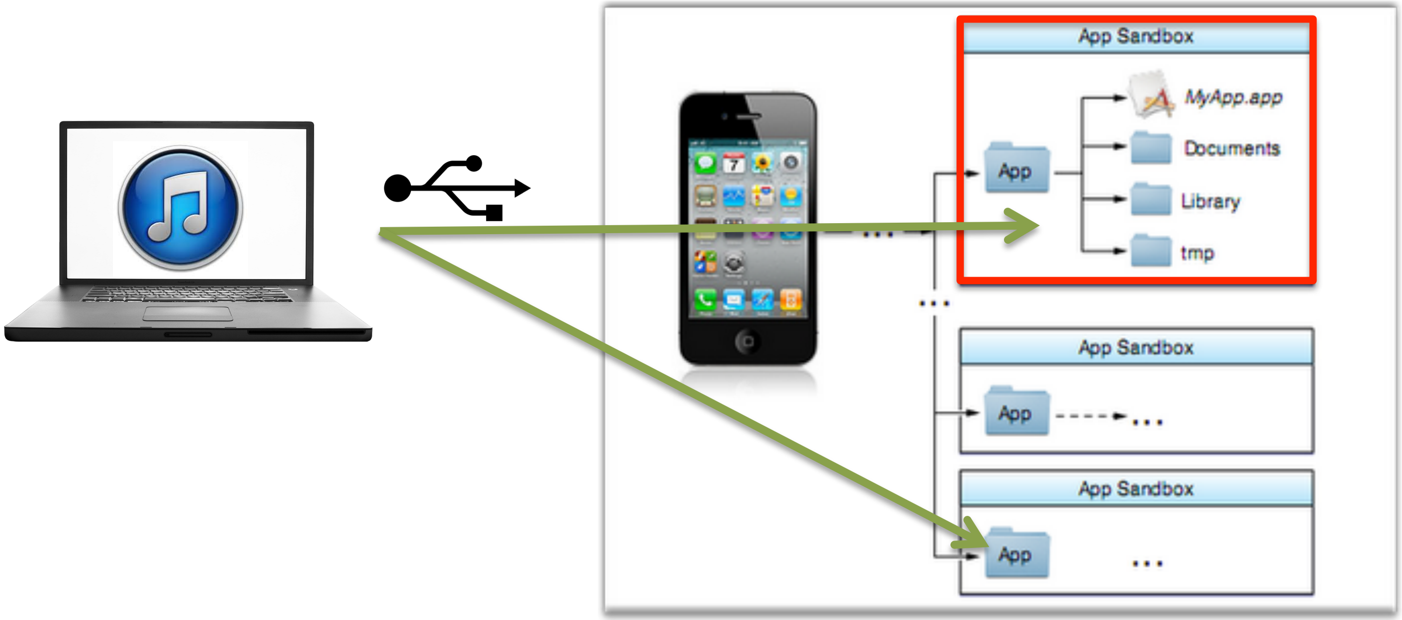
In a case of convenience for users trumping security, Starbucks has been storing the passwords for its mobile-payment app, along with geolocation data, in clear text

**By Evan Schuman**

January 15, 2014 11:09 AM ET 26 Comments

# Attack III: Stealing Credentials

- However, from a USB connection, a host computer has access to the contents of all apps



# Attack III: Stealing Credentials

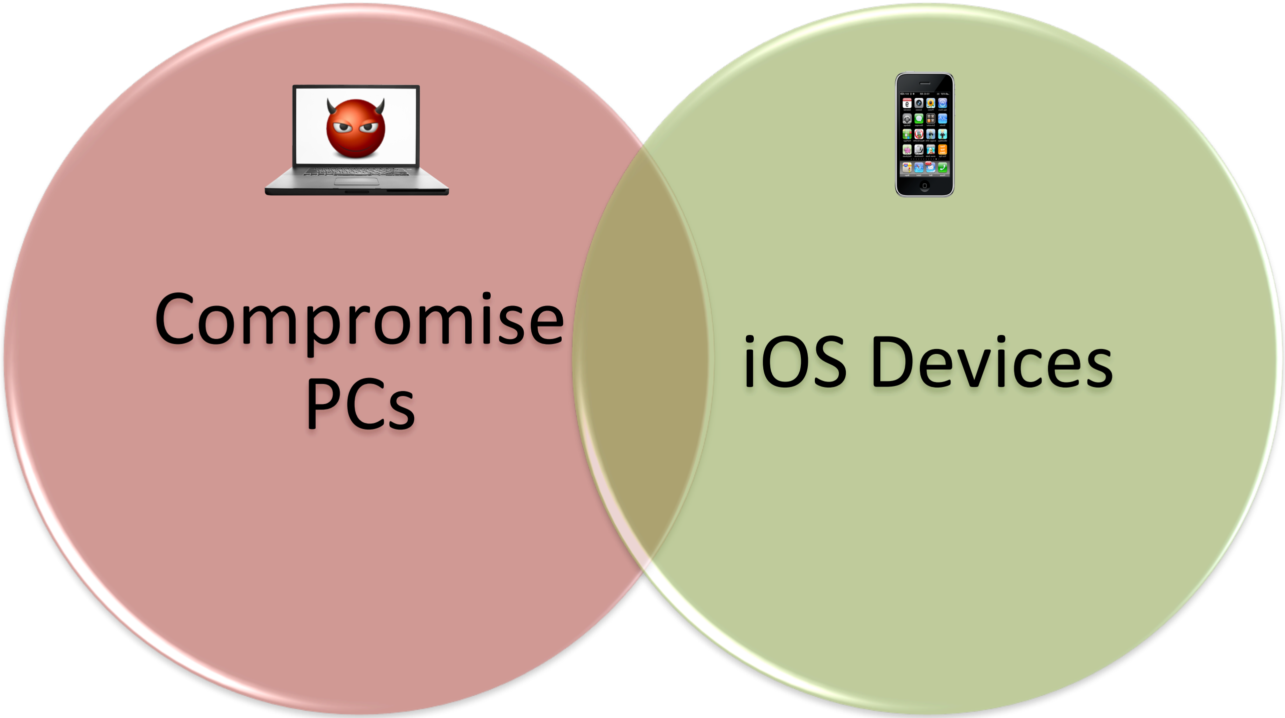
- As a proof of concept, we implemented a tool that can retrieve the cookies of Facebook and Gmail apps from a USB-connected iOS device
- By reusing the cookies, we successfully logged in as the iOS user via the web services for both Facebook and Gmail

# Outline

- Background and Motivation
- Security Risks of Connecting iOS Devices to Compromised PCs
- Measurement Results
- Conclusion

# The Goal of the Measurement

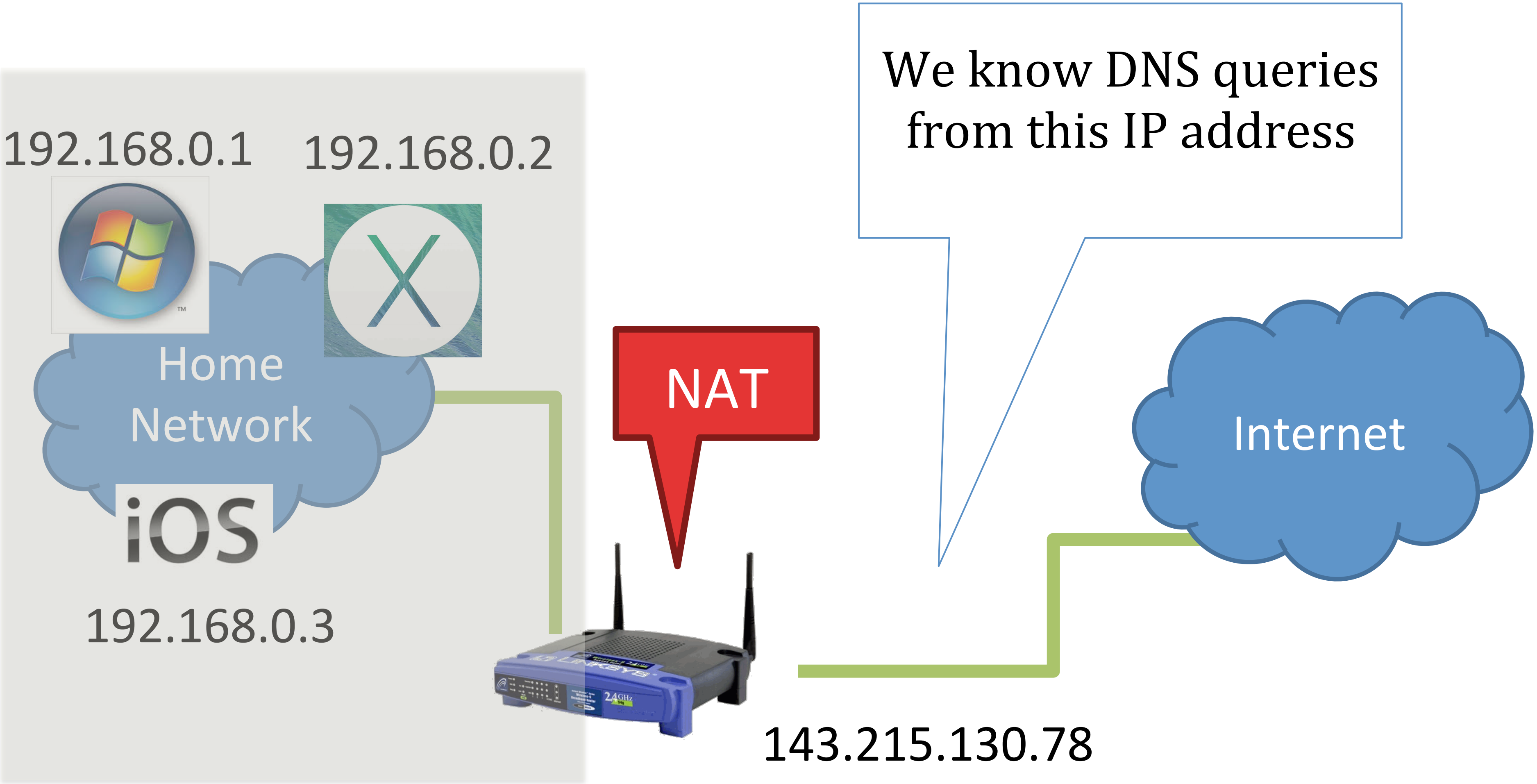
- To quantitatively estimate how many users would connect iOS devices to compromised PCs



# Two Main Datasets

- DNS Query Dataset
  - Obtained from two large ISPs in the US, collected in 13 cities for five days in Oct 2013
  - 54 million client IDs, 62 million queries, and 12 billion records daily from 13 sensors in total
- Labeled C&C Domains
  - Obtained command and control (C&C) domain names for botnets that Damballa is tracking

# Basic Information



We know DNS queries from this IP address

192.168.0.1

192.168.0.2

Home Network

iOS

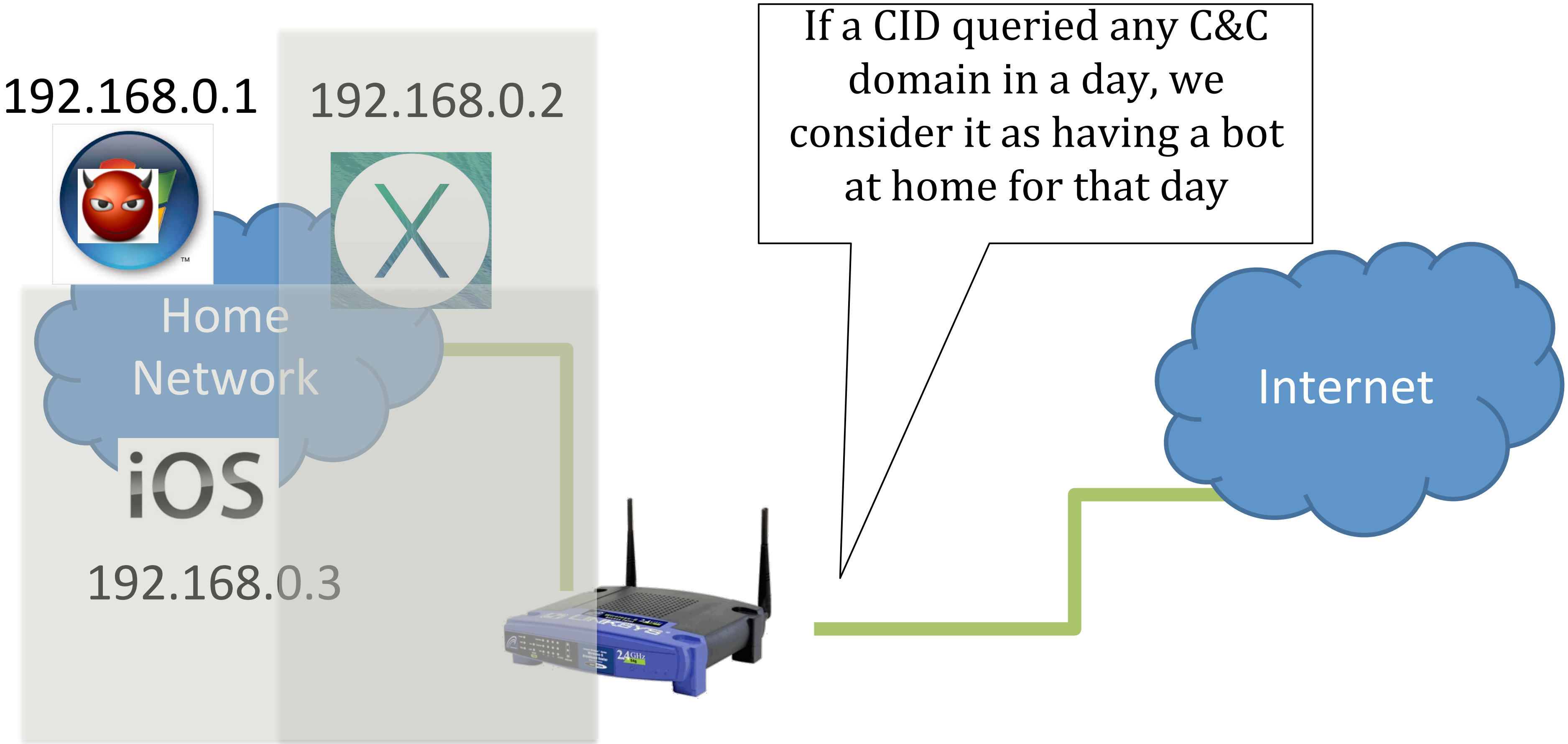
192.168.0.3

NAT

Internet

143.215.130.78

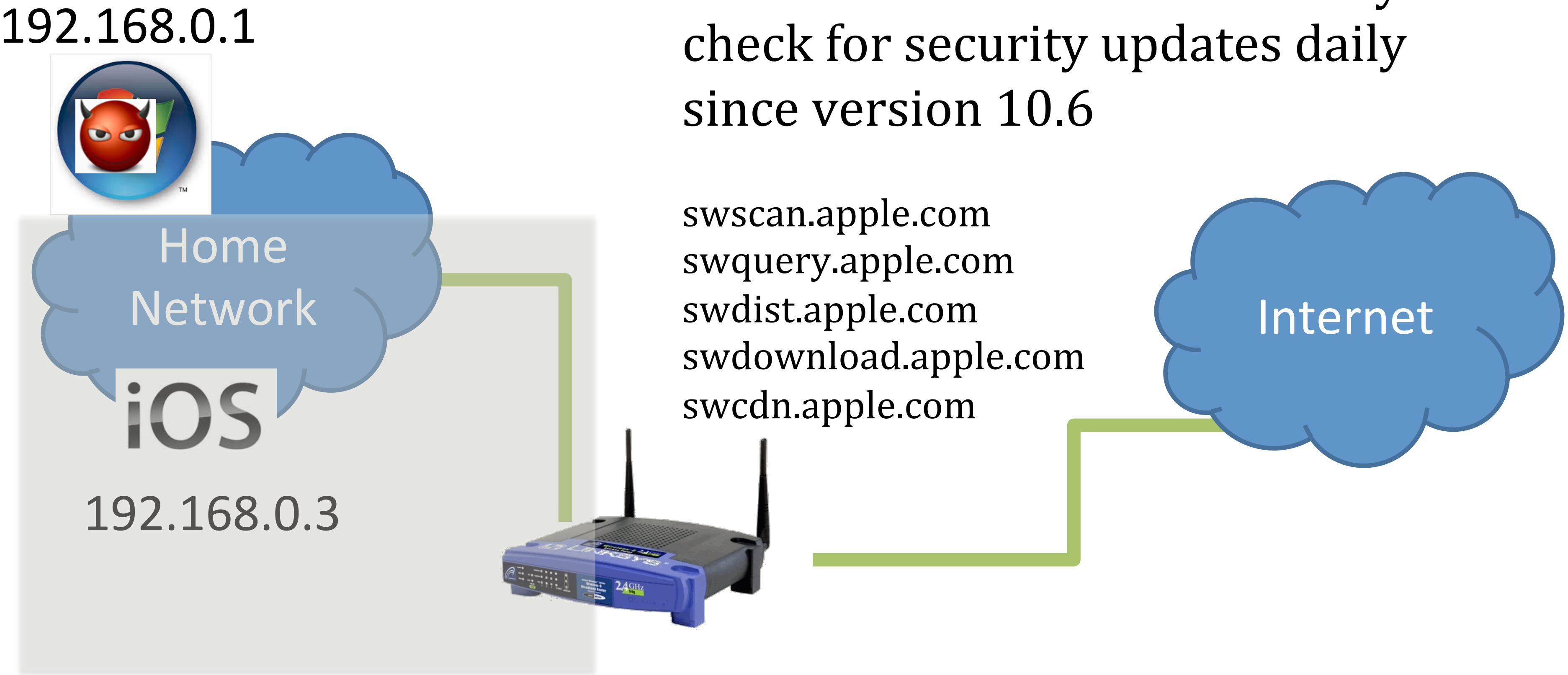
# Step1: Determine Bot Population



If a CID queried any C&C domain in a day, we consider it as having a bot at home for that day

473,506 infected CIDs on 10/12/2013.

# Step2: Exclude Mac OS X



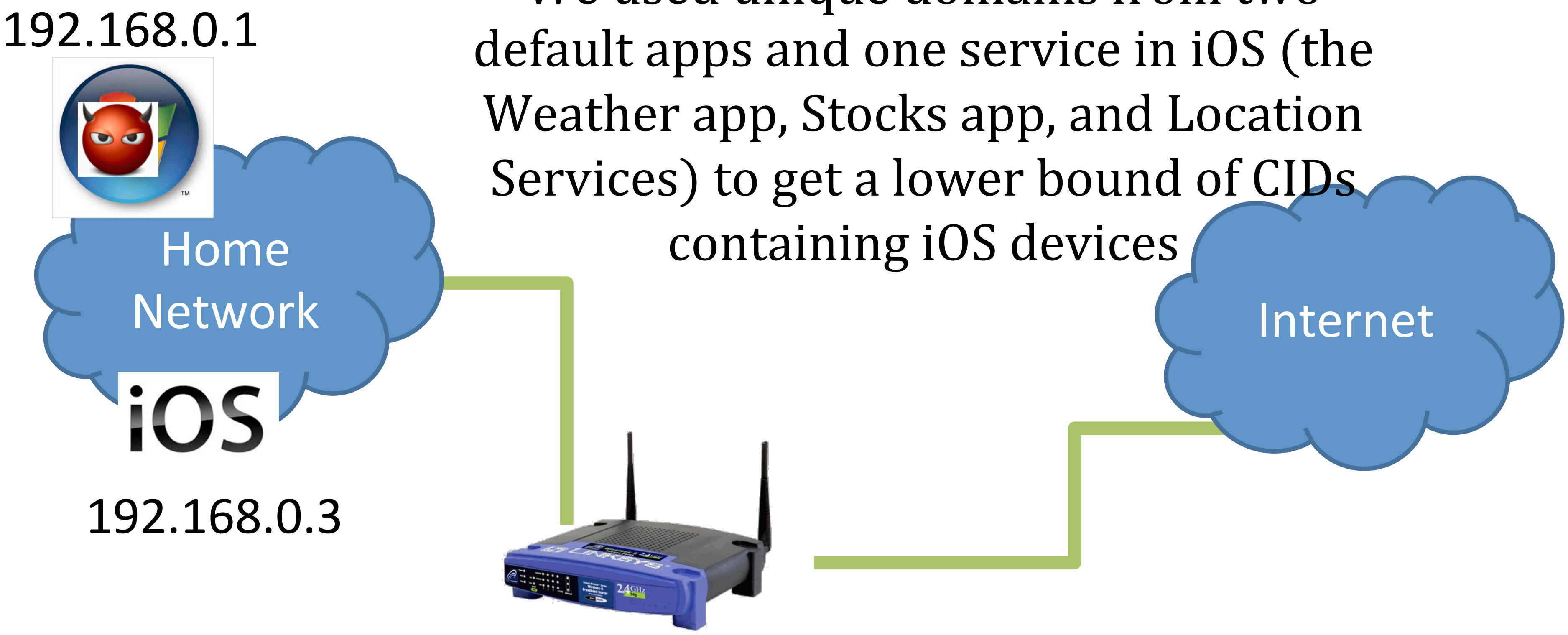
Mac OS X is set to automatically check for security updates daily since version 10.6

swscan.apple.com
swquery.apple.com
swdist.apple.com
swdownload.apple.com
swcdn.apple.com

After excluding Mac OS X, we have 466,540 bot CIDs

# Step3: Determine coexistence of iOS

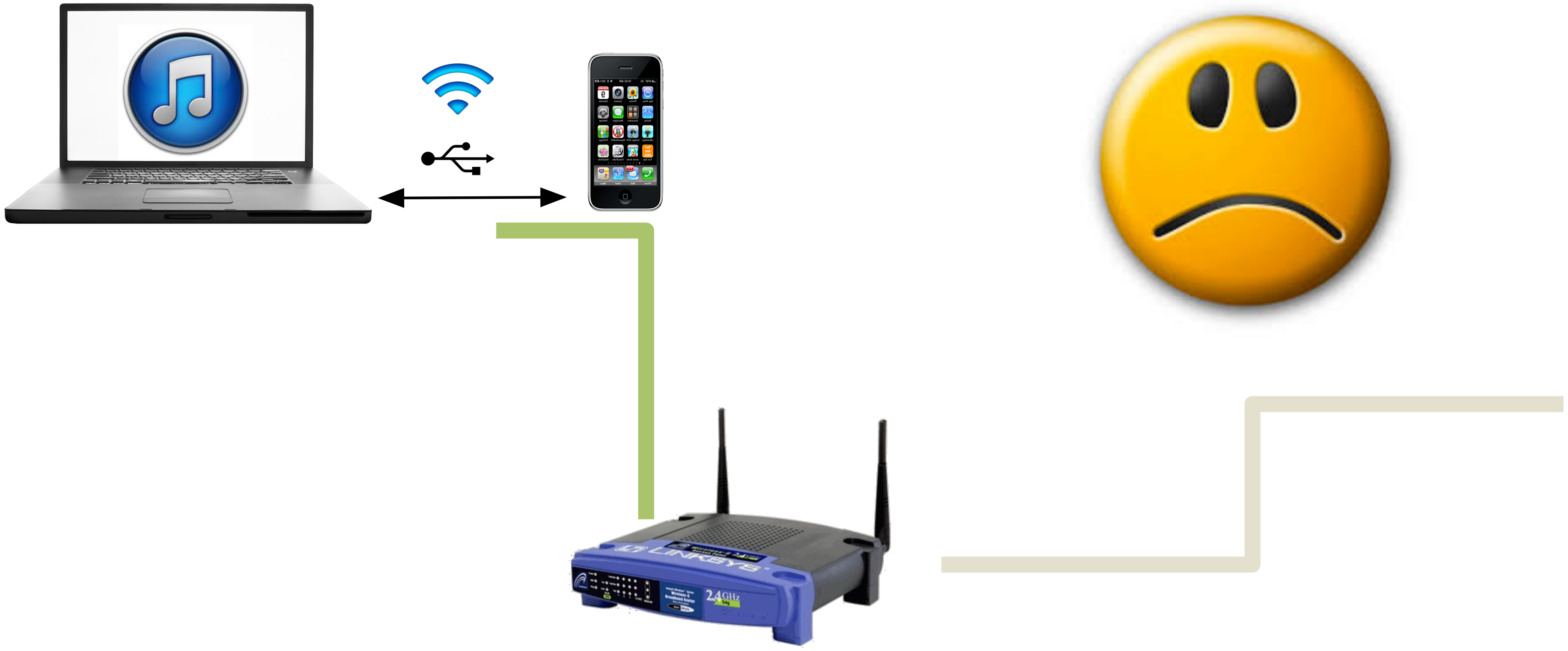
We used unique domains from two default apps and one service in iOS (the Weather app, Stocks app, and Location Services) to get a lower bound of CIDs containing iOS devices



Of 466,540 CIDs without Mac OS X traffic, 142,907 queried these domains

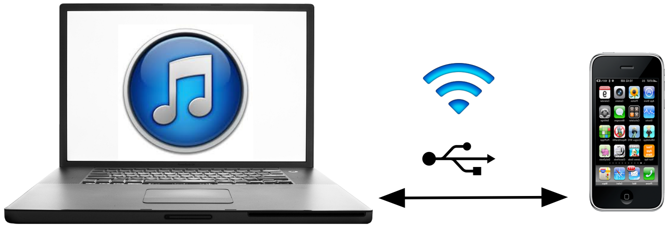
# Step4: Determine Windows iTunes Purchases

- Connecting iOS devices to a PC does not generate observable network traffic



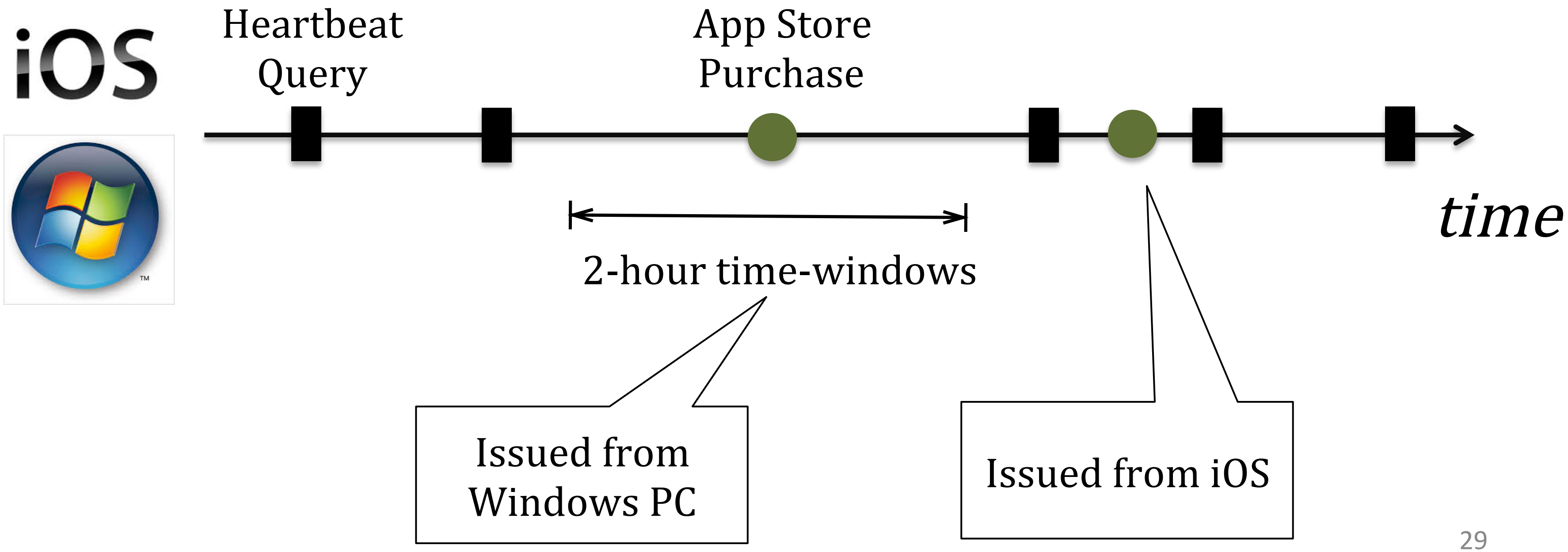
# Step4: Determine Windows iTunes Purchases

- More evidences
  - If iTunes is installed on a user's PC and is also used to purchase some items from the App Store, the user will eventually connect her iOS devices to the PC



# Step4: Determine Windows iTunes Purchases

- *Solution*: Leverage iOS heartbeat DNS queries
  - iOS devices must send an HTTP request to init-p01st.push.apple.com to get push server configurations at least every 1,800s



Heartbeat Query

App Store Purchase

*time*

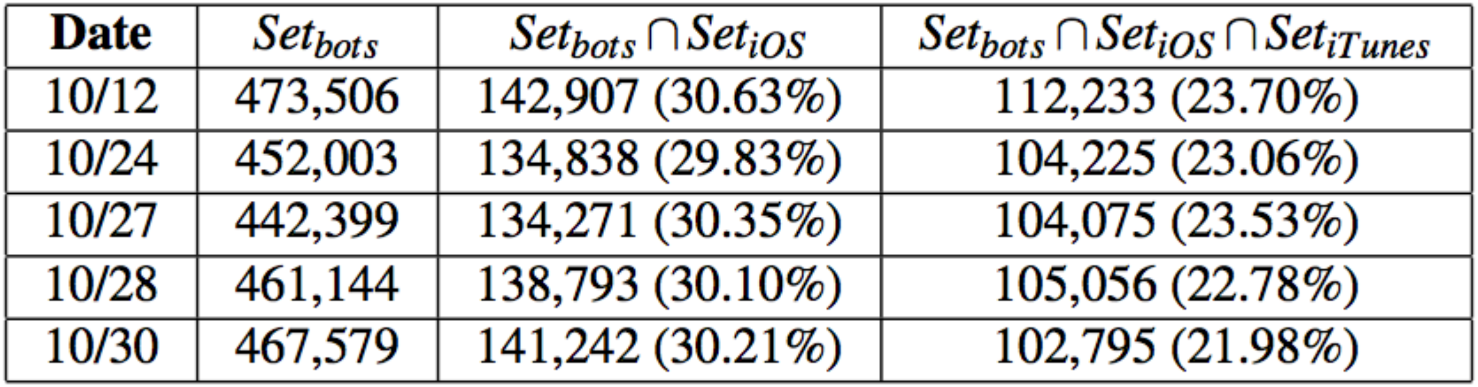
2-hour time-windows

Issued from Windows PC

Issued from iOS

# Measurement Summary

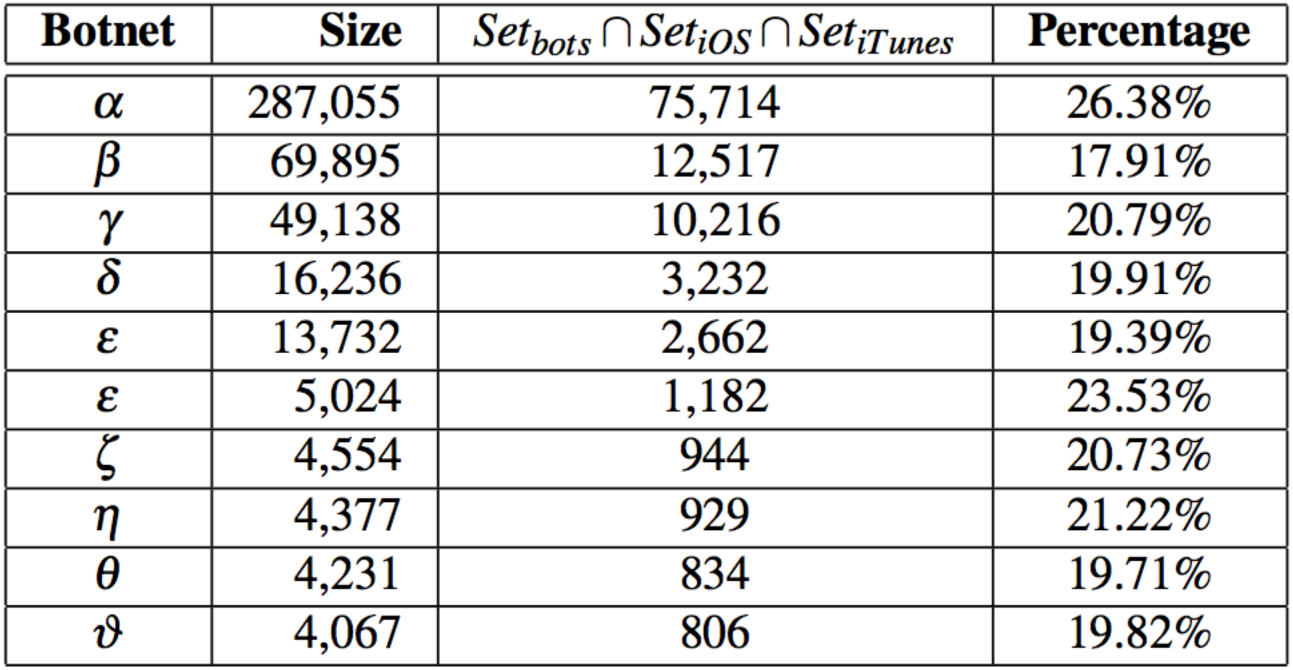
- 23% of all bot population have connections with iOS devices



| Date | $Set_{bots}$ | $Set_{bots} \cap Set_{iOS}$ | $Set_{bots} \cap Set_{iOS} \cap Set_{iTunes}$ |
|---|---|---|---|
| 10/12 | 473,506 | 142,907 (30.63%) | 112,233 (23.70%) |
| 10/24 | 452,003 | 134,838 (29.83%) | 104,225 (23.06%) |
| 10/27 | 442,399 | 134,271 (30.35%) | 104,075 (23.53%) |
| 10/28 | 461,144 | 138,793 (30.10%) | 105,056 (22.78%) |
| 10/30 | 467,579 | 141,242 (30.21%) | 102,795 (21.98%) |

# Measurement Summary

- 23% of all bot population have connections with iOS devices



| Botnet | Size | $Set_{bots} \cap Set_{iOS} \cap Set_{iTunes}$ | Percentage |
|---|---|---|---|
| $\alpha$ | 287,055 | 75,714 | 26.38% |
| $\beta$ | 69,895 | 12,517 | 17.91% |
| $\gamma$ | 49,138 | 10,216 | 20.79% |
| $\delta$ | 16,236 | 3,232 | 19.91% |
| $\varepsilon$ | 13,732 | 2,662 | 19.39% |
| $\varepsilon$ | 5,024 | 1,182 | 23.53% |
| $\zeta$ | 4,554 | 944 | 20.73% |
| $\eta$ | 4,377 | 929 | 21.22% |
| $\theta$ | 4,231 | 834 | 19.71% |
| $\vartheta$ | 4,067 | 806 | 19.82% |

# Outline

- Background and Motivation
- Security Risks of Connecting iOS Devices to Compromised PCs
- Measurement Results
- Conclusion

# Conclusion

- Demonstrated attacks:
  - Bypass Apple DRM and install Apple-signed malicious apps
  - Stealthily provision the devices and install attacker-signed malicious apps
  - Obtain app credentials (e.g., Gmail and Facebook cookies)
- Measurement Results: 23% of all bot population have connections with iOS devices

# QUESTIONS?