Abusing Mobile Games

Ming Chow
mchow@cs.tufts.edu
@0xmchow

Mobile Games: The Not-So Surprising Numbers

- “Worldwide spending on mobile game apps tripled in 2013 to $16B” (http://venturebeat.com/2014/02/19/worldwide-spending-on-mobile-game-apps-tripled-in-2013-to-16b/)
- “Mobile games account for top global apps worldwide in Jan 2014” (http://e27.co/mobile-games-account-for-top-global-apps-worldwide-in-jan-2014/)
- “Gaming apps accounted for around 41% of downloads from the Apple and Google stores, and was 61% on Amazon” (http://blogs.wsj.com/digits/2014/04/10/the-one-thing-mobile-users-can-agree-on-games/)

Mobile: What We Know

- Devices are of high value, can be easily lost or broken
- Has good computational power
- It is “always on” (including networking)
- Features: GPS, accelerometer, compass, bluetooth, Wi-Fi, NFC
- Constraints: battery, screen size, input
- New security model including app distribution
- No authorized method for gaining administrative access by default

What Has Changed for Games

- Development cycle and cost
- *The goal for players*
- *Business model*
- Distribution of game client and content
- Dependency on network connection and third-party systems

Typical Architecture of Mobile Games

- The game client
- Servers
- APIs
- Operating System
- Mobile Carriers

Motivations for Abuse, Unethical Behavior

- Don’t want to grind
- Don’t want to wait for next full meter
- Don’t want to spend money on virtual goods
- Don’t want to lose
- Achievements
- Want to win
- Want to steal data

Abuse Mechanisms

- File modifications and tampering
- Malware and piracy
- Time state attacks
- Faking location and sensor data
- Disconnection and latency
- API abuse

File Modifications and Tampering

- Game data commonly stored in `.plist` or XML files or in SQLite databases on the client side (the app)
- Example: Pocket Trains
  - iOS: Edit the `.plist` file under preferences for the game; risk getting banned
  - Android (assumes rooted device): Edit the file `root/data/data/com.nimblebit.pockettrains/shared_prefs/com.nimblebit.pockettrains.xml`
  - Source: http://www.pockettrainswiki.com/wiki/Cheating

Malware

- Repackage apps on Android
- Example 1: Freedom (Android) - reveals “hackable” applications on device; in-app purchases made free through middle-man, not through Google Play http://www.netnames.com/blog/2013/09/in-appropriate-mobile-behaviour/
- Example 2: Flappy Birds http://nakedsecurity.sophos.com/2014/02/11/flappy-bird-really-is-dead-beware-of-infected-fakes-that-promise-to-keep-him-alive/

Out-of-Band Spam

- Case-in-point, do a search for `candy crush saga cheats`
- Seedy ad networks http://blog.trendmicro.com/leaky-ad-networks-put-mobile-game-players-risk/
- ...leading to potentially malicious websites and networks

Time State Attacks

- Many games are dependent on the actual time
- The idea: (incrementally) change the phone or tablet's internal clock forward or backwards
- “10 year old girl hacker CyFi reveal her first zero-day in Game at #DefCon 19” http://thehackernews.com/2011/08/10-year-old-girl-hacker-cyfi-reveal-her.html
- Works in games such as:
  - Candy Crush Saga: http://forum.xda-developers.com/showthread.php?t=2235910
  - Pocket Trains (jobs finish faster by moving clock forward BUT trains may then have negative fuel)

Disconnection and Latency

- Higher disconnection, higher latency, and high data loss on mobile devices.
- Detecting time state attacks is much harder… (e.g., disconnect device from Wi-Fi)
- Double-edge sword for player:
  - In favor: disconnect on imminent loss, no penalty. Example: FIFA ‘14
  - Not in favor: dead as you know it

Faking Location and Sensor Data

- For augmented reality or location-based games
- Was a big problem for Foursquare
- Easy to spoof gyroscope, accelerometer, compass, geolocation data --especially on a rooted or jailbroken device
- “Most systems currently lack software or hardware checks (e.g., Trusted Platform Module)” (Yahyavi, Pang, and Kemme)

API Abuse

- Use a proxy as middle-man (e.g., mitmproxy)
- iOS allows for system-wide HTTP proxy
- Example (now fixed): Apple Game Center
  - Attack 1: intercept and modify `score-value` field
  - Attack 2: capture email hashes (SHA-1)
  - Source: “Hacking iOS Game Center and Passbook with Proxies” by Karl Fosaaen, NetSPI http://louisvilleinfosec.com/wp-content/uploads/2013/02/KarlFosaaen-iOS-GC-and-PB.pdf

Source: https://www.netspi.com/Portals/0/images/Blog/GameCenter-Blog-Cut-The-Rope.png

Privacy and Information Leakage

- Still way too many apps and developers transmit data HTTP (i.e., plaintext) and not HTTPS
- App permissions hell: http://blog.zscaler.com/angry-apps-saga
- Two years ago, Angry Birds and many other iOS games were calling `ABAddressBookCopyArrayOfAllPeople`. Source: http://blog.veracode.com/2012/02/adios-say-goodbye-to-nosy-iphone-apps/
- Example: QuizUp http://www.theverge.com/2013/11/26/5146998/quizup-security-privacy-issues-fix-on-the-way
- Facebook, Twitter, etc. credentials stored in plaintext in SQLite databases (thanks Joey Peloquin)

Existing Security Mechanisms

- Banning
- Penalties
- CAPTCHAs
- Hash functions and checksums
- (PC games have more glorified mechanisms including PunkBuster, Valve Anti-Cheat, Warden for WoW)
- Mobile Guard: continuous exchange of protection mechanisms between client and server (Grimen, Mönch, Midtstraum)
- Dead reckoning to predict location of player at certain moment

Proposed Solutions (Yahyavi, Pang, Kemme)

- Verify Wi-Fi position by sending nearest SSIDs and their signal strength
- Q&A using local places information (e.g., Google Places API). So we have a game within a game
- Verify unreachable positions (e.g., middle of ocean) via path
- Verify location using picture taken from camera => reverse image search
- Verify network statistics and information, carrier specific (e.g., via AT&T API)
- Facial recognition via camera and biometrics for transaction security

The Good News

- Many complexities (and hence, culprits), are not available in mobile games (Bono, Caselden, Landau, Miller):
  - Third-party plugins
  - User-generated content (e.g., the “nude patch”)
  - Scripting engines
  - Botting (without a lot of difficulties)

What's the Loss (or why do we care)?

1. **Players** - loss of fun; declining resources (including battery life); loss of purchased virtual goods; loss of personal data; spike in cell phone and credit card bills
2. **Game Developers** - loss revenue from in-app purchases or from the app itself; bad data; cost for computing services increase
3. **Carriers and Computing Services**- declining quality of service
4. **Platforms** (e.g., iOS, Android)

To Ponder

- Mobile games is a great arena to see whatcouldpossiblegowrong in the mobile space
- No one wins; everyone's reputation and a lot of money are at stake
- Many stories; a mess as evident by the cases presented
- Question: what's the trust model now?
- *The bigger question: why are we not emphasizing on the security of mobile games?*

Additional References

- G. Grimen, C. Mönch, R. Midtstraum, “Tamper Protection of Online Clients through Random Checksum Algorithms,” in Proceedings of Information Systems Technology and its Applications 5th International Conference (ISTA 2006), May 2006.
- Stephen Bono , Dan Caselden , Gabriel Landau , Charlie Miller, Reducing the Attack Surface in Massively Multiplayer Online Role-Playing Games, IEEE Security and Privacy, v.7 n.3, p.13-19, May 2009.
- A. Yahyavi, J. Pang, B. Kemme, "Towards Providing Security For Mobile Games," in Proceedings of 8th ACM MobiCom Workshop on Mobility in the Evolving Internet Architecture (MobiArch), Miami, USA, 2013, p47-52. http://www.cs.mcgill.ca/~syahya/Yahyavi-MobiArch-MobGameSec.pdf
- “AWS Case Study: Supercell” http://aws.amazon.com/solutions/case-studies/supercell/
- “Building Mobile Games on AWS” http://www.slideshare.net/AmazonWebServices/building-mobile-games-on-aws-gmg301
- http://www.bloomberg.com/news/2014-01-29/nsa-spying-on-apps-shows-perils-of-google-candy-crush-.html