Are these Ads Safe: Detecting Hidden Attacks through Mobile App-Web Interfaces

Vaibhav Rastogi\textsuperscript{1}, Rui Shao\textsuperscript{2}, Yan Chen\textsuperscript{3}, Xiang Pan\textsuperscript{3}, Shihong Zou\textsuperscript{4}, and Ryan Riley\textsuperscript{5}

\textsuperscript{1}University of Wisconsin and Pennsylvania State University
\textsuperscript{2}Zhejiang University
\textsuperscript{3}Northwestern University
\textsuperscript{4}Beijing University of Posts and Telecommunications
\textsuperscript{5}Qatar University
Consider This...
Consider This...
The Problem

• Enormous effort toward analyzing malicious applications
• App may itself be benign
  • But may lead to malicious content through links
• **App-web interface**
  • Links inside the app leading to web-content
  • Not well-explored
• Types
  • Advertisements
  • Other links in app
Outline

- App-Web Interface Characteristics
- Solution
- Results
- Conclusion
Outline

- App-Web Interface Characteristics
- Solution
- Results
- Conclusion
App-Web Interface Characteristics

• Can be highly dynamic
• A link may recursively redirect to another before leading to a final web page
• Links embedded in apps
  • Can be dynamically generated
  • Can lead to dynamic websites
• Advertisements
  • Ad libraries create links dynamically
  • Ad economics can lead to complex redirection chains
Advertising Overview

AdColony
admob
millennialmedia
UMENG
InMobi

Advertisers  Ad networks  Apps / Developers  Users
Ad Networks

- Ad libraries act as the interface between apps and ad network servers

- Ad networks may interface with each other
  - Syndication – One network asks another to fill ad space
  - Ad exchange – Real-time auction of ad space

- App or original ad network may not have control on ads served
Outline

App-Web Interface Characteristics
Solution
Results
Conclusion
Solution Components

• **Triggering**: Interact with app to launch web links

• **Detection**: Process the results to identify malicious content

• **Provenance**: Identify the origin of a detected malicious activity
  • Attribute malicious content to domains and ad networks
Solution Architecture
Triggering

• Use AppsPlayground\(^1\)
  • A gray box tool for app UI exploration
  • Extracts features from displayed UI and iteratively generates a UI model

• A novel computer graphics-based algorithm for identifying buttons
  • See widgets and buttons as a human would

Detection

- Automatically download content from landing pages
- Use VirusTotal for detecting malicious files and URLs
Provenance

• How did the user come across an attack?
 • Code-level attribution
   • App code
   • Ad libraries
     • Identified 201 ad libraries
 • Redirection chain-level attribution
   • Which URLs led to attack page or content
Results

• Deployments in US and China

• 600 K apps from Google Play and Chinese stores

• 1.4 M app-web links triggered

• 2,423 malicious URLs

• 706 malicious files
Case Study: Fake AV Scam

• Multiple apps, one ad network: Tapcontext
• Ad network solely serving this scam campaign
• Phishing webpages detected by Google and other URL blacklists about 20 days after we detected first instance
Case Study: Free iPad Scam

• Asked to give personal information without any return
• New email address receiving spam ever since
• Origins at Mobclix and Tapfortap
  • Ad exchanges
  • Neither developers nor the primary ad networks likely aware of this
Case Study: iPad Scam from static link

- Another Scam, this time through a static link embedded in app
- Link target opens in browser and redirects to scam
- Not affiliated with Facebook
Case Study: SMS Trojan Video Player

- Ad from nobot.co.jp leads to download a movie player
- Player sends SMS messages to a premium number without user consent

Click on ad
Outline

- App-Web Interface Characteristics
- Solution
- Results
- Conclusion
Limitations

• Incomplete detection
  • Antiviruses and URL blacklists are not perfect
  • Our work DroidChameleon\(^2\) shows this

• Incomplete triggering
  • App UI can be very complex
  • May still be sufficient to capture advertisements

Conclusion

• Benign apps can lead to malicious content
• Provenance makes it possible to identify responsible parties
• Can provide a safer landscape for users
  • Screening offending applications
  • Holding ad networks accountable for content
• Working with CNCERT to improve the situation
Future Work

• Speeding up collection of ads

• Goals of analyzing an order of magnitude more ads in shorter time
Software and Dataset

• Dataset of 201 ad libraries: http://bit.ly/adlibset

Thank you!
Backup
Related Work

• Web Malvertising

• Other ad security and Privacy

• UI exploration

• Malware analysis and detection
Comparison with Web Malvertising

• Focus on mobile applications

• Triggering component for web malvertising is trivial

• Different malware propagation mechanisms: drive-by-downloads vs. trojans