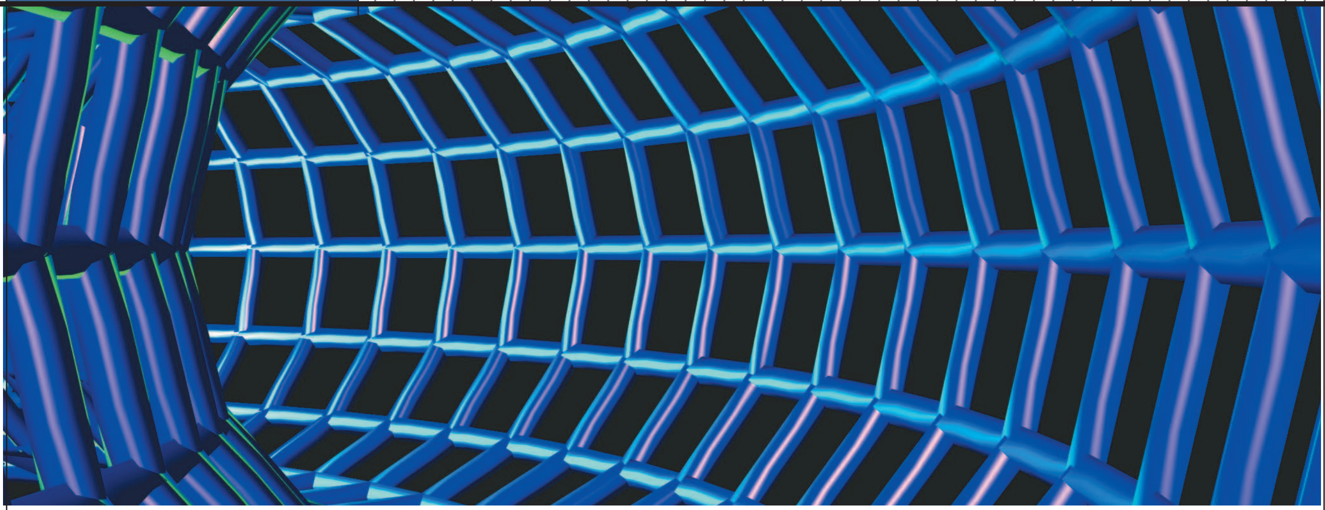


Detection of Mobile Malware in the Wild

Mahinthan Chandramohan and Hee Beng Kuan Tan

Nanyang Technological University, Singapore

New techniques for detecting the presence of mobile malware can help protect smartphones from potential security threats.

Because they're ultraportable and easy to use, smartphones enhance many of our day-to-day tasks, providing instant access to the wealth of information available on the Internet as well as offering various functionalities such as easy credit-card payments, videoconferencing, and voice recognition. According to Gartner, more than 428 million mobile devices were sold worldwide in the second quarter of 2011, accounting for 25 percent of the total number of mobile devices sold to date (http://bit.ly/ru8t30). IMS Research predicts that annual smartphone sales will surpass 1 billion devices by the end of 2016 (http://bit.ly/qwDpiU).

Users increasingly depend on their mobile phones, and the vast amount of personal information these devices now store has made them a prime target for malware attacks. Since researchers at Kaspersky Lab found Cabir, the first network worm for mobile phones, in 2004 (http://bit.ly/uEekX4), malware has continued to increase along with smartphones' popularity.

Surveys provide a better understanding of the motivations behind mobile malware "in the wild"—the malicious applications available in mobile app markets.[1] This information in turn can help mobile-security researchers develop the novel techniques required to protect smartphones from security threats.

WHAT IS MOBILE MALWARE?

Malware—software that exhibits malicious behavior—is broadly categorized to include viruses, botnets, worms, and Trojan horses. Initially, malware merely highlighted a software system's security vulnerabilities, but the motivations behind it gradually changed, and its authors now use malware to gain financial benefits on a larger scale.

Many novel techniques for detecting malware have been proposed in the literature, most of them highlighted in a comprehensive 2011 survey by Manuel Egele and his colleagues.[2] However, mobile malware research is still in its infancy, even as malware authors shift their focus to smartphones.

In addition to malware, the other two major categories of threats to mobile devices are personal spyware and grayware.[1] Spyware collects information such as user location, SMS messages, and call history without the victim's knowledge. Spyware can't be labeled as illegal because it doesn't

Table 1. Behavioral classification overview for mobile malware in the wild.

Malware behavior	Description
Offers novelty and amusement	Primarily developed for fun or to show off the author's technical expertise; less serious; generates purposeless destruction. Example: Android.Walkinwat; www.symantec.com/security_response/writeup.jsp?docid=2011-033008-4831-99.
Sells user information	Secretly collects user details, such as location, installed applications, download history, and contact lists; these details are then sold to advertisers and marketers. Example: DroidDreamLight; http://blog.trendmicro.com/analysis-of-droiddreamlight-android-malware.
Steals user credentials	Captures user credentials, such as bank account details, by secretly snooping on text messages, capturing keystrokes by key logging, scanning documents, and launching phishing attacks. Example: Ikee.B; www.f-secure.com/v-descs/worm_iphoneos_ikee_b.shtml.
Manipulates content delivery	Generates premium-rate phone calls and sends text messages, perhaps to deliver content such as technical support, stock quotes, or adult services. Example: FakePlayer; www.symantec.com/security_response/writeup.jsp?docid=2010-081100-1646-99.
Sends SMS spam	Spams multiple messages to mobile phones that usually contain advertisements and phishing links. Example: Geinimi; www.symantec.com/security_response/writeup.jsp?docid=2011-010111-5403-99.
Manipulates search engine optimization	Improves website rankings in search engine results. Example: Hong Tou Tou; http://androidcommunity.com/android-trojan-alert-hong-tou-tou-20110216.

send information to the application's author, but installing personal spyware on a mobile phone without the device owner's authorization could be considered unethical.

Grayware is annoying, but it's far less serious than malware—for example, instead of disrupting computer operations or gathering sensitive information, it might change the user's font colors or install irritating pop-ups. Grayware applications are common in smartphones and sit just on the edge of legality: grayware authors describe their intentions and the application's behavior in the privacy policy, but users typically just click through and select the "agree" option without paying much attention to what the statement actually says.

Many researchers have categorized malicious mobile applications according to their authors' motivations and the applications' behavioral similarities.[1] Table 1 summarizes the most important examples, which come from http://bit.ly/omLkyJ.

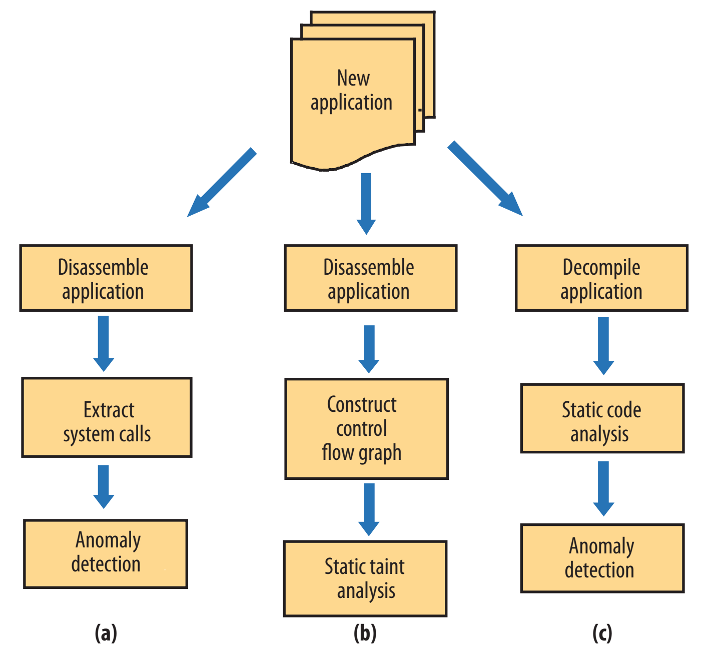


Figure 1. Static analysis techniques based on (a) system calls, (b) taint, and (c) source code.

MALWARE DETECTION TECHNIQUES

The techniques available for detecting mobile malware and other security vulnerabilities have varying strengths and weaknesses.

Static analysis

Static analysis is a quick, inexpensive approach to finding malicious characteristics or bad code segments in an application without executing them. The techniques illustrated in Figure 1 are widely used in a preliminary analysis, when suspicious applications are first evaluated to detect any obvious security threats.

Figure 1a shows a malware detection technique proposed for the Symbian OS.[3] This technique uses IDA Pro (http://bit.ly/p4s5Ap) to disassemble the mobile application and extract system calls (feature extraction). It then use Centroid Machine, a lightweight clustering mechanism, to classify the mobile application as either malicious or benign (anomaly detection). However, for a statistical (machine learning) approach, the current malware sample is relatively small, just 33 malicious and 49 benign mobile

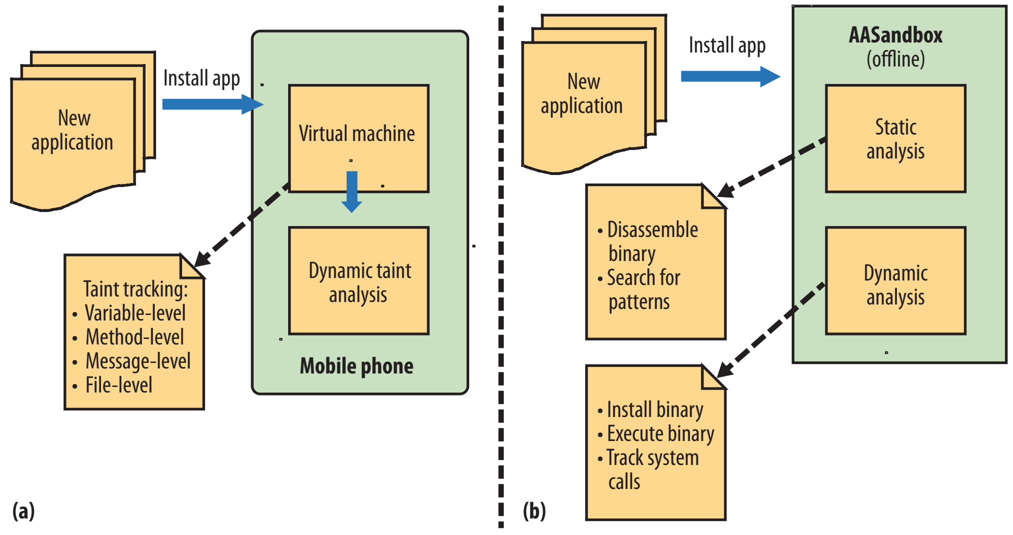


Figure 2. Dynamic analysis: (a) system-wide and (b) sandbox-based.

applications. Furthermore, because the researchers tested this approach only on popular applications, the performance for a normal or less popular application is unclear.

Figure 1b illustrates the use of PiOS, a technique proposed for performing static taint analysis on iOS application binaries.[4] PiOS disassembles the mobile application and constructs a control flow graph (CFG). The analysis considers paths originating from sensitive sources, such as the address book, current GPS coordinates, keyboard cache, unique device ID, and other phone-related information. Dataflow analysis checks for any sensitive data transmitted from the source to synch without notifying the user and thus causing privacy leaks. Tests using PiOS on 825 free applications from Apple's App Store and 582 apps from Cydia's BigBoss repository (http://bit.ly/9omdYr) revealed that more than 50 percent of applications leak the unique device ID without the user's knowledge. However, PiOS can only detect privacy leaks within a single application, and it fails if two or more applications are transitively chained together.

Figure 1c shows a malware detection technique proposed for Android.[5] This approach uses ded, a Dalvik decompiler (http://bit.ly/nWf2so), to generate Java source code from the application's installation image and then uses Fortify SCA (http://bit.ly/pyPajw), a static code analysis suite, to evaluate the recovered source code. Although the researchers tested this technique with 1,100 popular applications, it's generally limited to those that use a relatively small number of permissions and API calls.

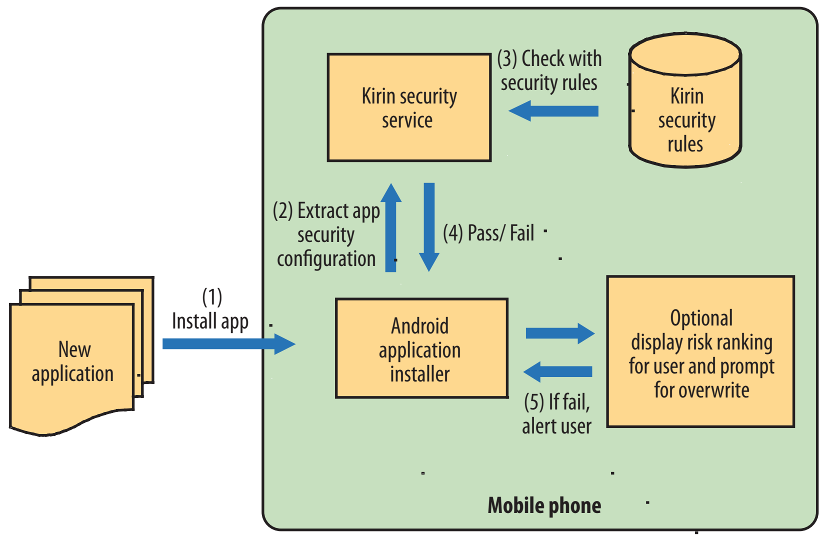
Dynamic analysis

Unlike static analysis, dynamic analysis involves executing the mobile application in an isolated environment, such as a virtual machine or emulator, so that researchers can monitor the application's dynamic behavior. Researchers primarily use dynamic analysis in taint tracking or system call tracing.

TaintDroid provides system-wide dynamic taint tracking for Android.[6] As Figure 2a shows, the mobile application passes to the Dalvik virtual machine to perform four granularities of taint propagation: variable, method, message, and file-level. Taint tracking marks any ambiguous data that originates from sensitive sources, such as location, microphone, camera, and other phone identifiers. This technique modifies the native library loader to ensure that all the native libraries are called from the virtual machine, thus preventing untrusted applications from executing native methods directly. Finally, dynamic analysis screens impacted data for any potentially sensitive data leaks before it leaves the system at the network interface—a taint sink.

Testing using TaintDroid with 30 popular third-party Android applications revealed that 15 of them share user location with advertisers and seven share phone identifiers with remote servers without the user's knowledge.[6] However, TaintDroid might suffer from false negative and false positive results; in addition, it focuses solely on dataflow and doesn't consider other vulnerabilities.

The Android Application Sandbox (AASandbox) systems offers two-step analysis for Android applications.[7] As Figure 2b shows, the mobile application passes to AASandbox, where it performs static and dynamic analyses in offline mode. Static analysis disassembles the application image binary and uses the disassembled code to search for any suspicious patterns (http://bit.ly/171MnI). Dynamic analysis executes the binary in an Android emulator and logs the system calls. Although the researchers used Android Monkey (ADB Monkey) to generate inputs, this isn't



Kirin security service
(3) Check with security rules
Kirin security rules
(2) Extract app security configuration
(4) Pass/ Fail
(1) Install app
New application
Android application installer
Optional display risk ranking for user and prompt for overwrite
(5) If fail, alert user
Mobile phone

Figure 3. Kirin Android application certification.

as effective as testing with real users.[7] Furthermore, this approach hasn't been tested against malware that exhibits polymorphic behavior or code fragment encryption.

Application permission analysis

Permissions play a key role in mobile applications: they convey the application's intentions and back-end activities to the user. In smartphones, permissions are clearly defined, so application authors must acquire appropriate permissions. However, some authors purposely hide the permissions they use in the application, leading to application vulnerability.

Figure 3 illustrates Kirin, an application certification for Android.[8] Kirin performs a permission check on the application during installation. When a user installs an application, Kirin extracts its security configurations and checks them against the security policy rule that it already has. If an application fails to pass all the security policy rules, Kirin can either delete it or alert the user.

The following is a sample policy rule extracted from Kirin:

```
An application must not have PHONE_STATE,
RECORD_AUDIO, and INTERNET permission labels'
```

This rule ensures that an application doesn't record audio or access the Internet while the user is talking on the phone, which protects against device-side call eavesdropping.

Testing using Kirin with 311 popular applications revealed that the rules flagged 10 applications for which the behavior of five was found to be questionable.[8] Kirin solely checks the application author's permission requests and doesn't examine how the application uses these permissions.

David Barrera and colleagues performed permission checks on 1,100 Android applications and used self-organizing maps (SOMs) to visualize the relationship between the applications and the permissions requested.[9] However, SOMs also focus solely on the application author's permission requests and don't examine how the application uses them.

In a study of Android application permission requests that included 100 paid and 856 free applications, approximately 93 percent of the free and 82 percent of the paid applications had at least one dangerous permission request.[10] Internet permission is the most common dangerous request. However, solely analyzing the permission request isn't sufficient for mobile malware detection; this must be done in parallel with static or dynamic analysis.

Cloud-based detection

Because of limited computational power and energy sources, smartphones don't carry fully featured security mechanisms. Running a simple file scanner on an Android HTC G1 device takes nearly 30 minutes and reduces the battery life by 2 percent.[11] A scanning application reportedly runs 11.8 times slower on an HTC G1 than on a desktop PC, highlighting the need for new mobile malware analysis techniques.[12]

Figure 4a shows Paranoid Android (PA), a cloud-based malware protection technique that moves security analysis and computations to a remote server that hosts multiple replicas of mobile phones running on emulators.[11] A tracer, located in the smartphone, records all the necessary information required to replay the mobile application's execution. The tracer transmits the recorded information to the cloud-based replayer, which replays the execution in the emulator. The replayer can deploy several security checks, such as dynamic malware analysis, memory scanners, system call anomaly detection, and commercial antivirus scanning, from the cloud's ample resources.

PA uses a proxy to temporarily store inbound network traffic information so that the phone can save energy by not sending this data back to the server. Instead, the server can directly contact the proxy to get the network traffic information needed to successfully replay the execution. However, PA incurs some significant overhead, increases the CPU load by 15 percent, and consumes 30 percent more energy during heavyweight tasks. Furthermore, because tracing systems implement the tracer module in the user

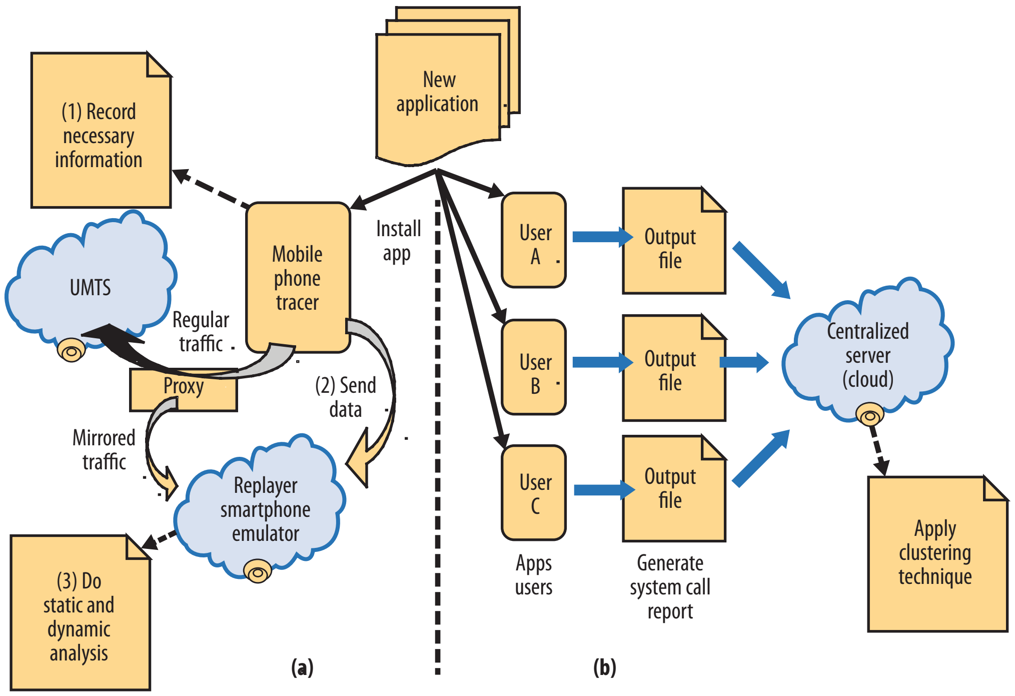

(1) Record necessary information
New application
UMTS
Regular traffic
Mobile phone tracer
Install app
Proxy
(2) Send data
Mirrored traffic
Replayer smartphone emulator
(3) Do static and dynamic analysis
(a)
User A
User B
User C
Apps users
Output file
Output file
Output file
Generate system call report
Centralized server (cloud)
Apply clustering technique
(b)

Figure 4. Cloud-based malware protection techniques: (a) Paranoid Android and (b) Crowdroid.

space, the calls incur heavy overhead compared to native execution. For example, tracing a single system call such as read() takes 0.7 milliseconds in the user space, but less than 0.1 millisecond in the kernel.

Figure 4b depicts Crowdroid, a behavior-based mobile malware detection technique for Android.[13] Crowdroid is a lightweight client application that monitors system calls invoked by the target mobile application, preprocesses the calls, and sends them to the cloud, where a clustering technique helps determine whether the application is benign or malicious. Increased use of Crowdroid will result in improved malware detection, but using the approach initially might cause from false positives as the sample size is still very small. Moreover, it isn't clear how users will react when they're asked to send application behavior to a third party, and total dependence on user behavior might not produce accurate results.

CloudAV is a cloud-based antivirus file scanning mechanism, but it lacks the features required to detect zero-days attacks, remote exploits, and memory-resident attacks.[14]

Battery life monitoring

Because smartphones have limited battery capacity, observing energy consumption sometimes identifies malicious applications, which consume more energy than benign ones. If normal user behavior, current battery state, and other domain-specific details such as signal strength and network traffic are known, it's possible to detect hidden malicious activities more precisely.

Researchers have tested the VirusMeter implementation for the Symbian OS with relatively small malware samples.[15] However, because VirusMeter relies completely on user-centric power models, unpredictable user behavior and malware-injected fake events can affect power model accuracy. Furthermore, this approach isn't applicable for smartphones with multitasking features.

SMARTPHONE PROTECTION TIPS

Although various tools and techniques are available for detecting malware attacks and protecting smartphones, users must be aware of potential security threats and their consequences. It's widely accepted that user negligence and lack of awareness of potential threats contribute to the success of security attacks.

Following a few good practices can help protect smartphones from potential threats:

- Install a good mobile security application that can protect the smartphone from attacks and alert the user when a suspicious event occurs.
- Download all mobile applications from trusted, official application providers. Avoid downloading anything from untrusted third-party app stores.
- Before downloading an app, read the reviews and the ratings, even if the application author is well-known.
- During installation, always read the permissions requested by the application. If something seems suspicious, don't install the application. It's better to leave

a comment on the site, which may help others in the future.

- Turn off Wi-Fi, Bluetooth, and infrared when they aren't in use. Be cautious when connecting to unsecured public Wi-Fi networks. This includes enabling the firewall, disabling sharing, and using SSL or a virtual private network.
- Always keep applications up to date and make sure that firmware is updated immediately when it becomes available for the mobile phone.
- Encrypt all confidential data stored in the mobile phone and back it up regularly. Make sure sensitive information isn't cached locally.
- Whenever possible, set a password for confidential files and applications.

Bringing the user into the smartphone security ecosystem could be the key to achieving a dramatic reduction in successful malware attacks.

- Don't click on Internet links that seem suspicious or untrustworthy. If absolutely necessary, visit the website by typing its URL—don't copy and paste links into the browser. This helps protect mobile phones from drive-by download attacks.
- Always monitor the battery life, SMS, and call charges. Any unusual behavior should prompt a thorough check on recently installed applications. There's a high possibility that the mobile phone is under a security attack.
- Finally, if the mobile phone is stolen, delete all the apps, contacts, and confidential data remotely, and use the unique device ID to block the stolen mobile phone.

It's highly unlikely that an efficient mobile malware protection tool would have zero false positives. Thus, following these good practices will help to protect smartphones from the vast majority of malware threats in the wild.

Mobile malware is more than a piece of malicious software; it's growing rapidly and is strongly linked to the underground economy. Thus, preventing mobile malware attacks has become critical, and smartphone security research is focused on both detecting and preventing malicious applications from infecting mobile devices.

In smartphones with constrained resources, the measures that can be taken to detect mobile security attacks are limited. To address the resource constraint issue, future mobile security mechanisms will leverage the power of cloud computing and distributed computing.

To prevent users from downloading mobile apps from untrusted third-party markets, smartphone manufacturers and platform developers should ensure that mobile devices are fully protected. New smartphone features like near-field communication (NFC) payment services might be the next major target for malware authors as they have the potential to replace credit cards and physical currency notes. NFC-based payment apps can be reverse-engineered to access stored credit-card credentials or even to develop malicious apps that can emulate a legitimate one. These threats can be mitigated by deploying strong encryption mechanisms to authenticate access to stored secret data and restricting unauthorized developers from accessing the NFC card.

Ultimately, bringing the user into the smartphone security ecosystem could be the key to achieving a dramatic reduction in successful malware attacks. C

References

1. A.P. Felt et al., "A Survey of Mobile Malware in the Wild," *Proc. ACM Workshop Security and Privacy in Mobile Devices* (SPMD 11), ACM, 2011, pp. 3-14.
2. M. Egele et al., "A Survey on Automated Dynamic Malware Analysis Techniques and Tools," *ACM Computing Surveys*, 2012; https://www.seclab.tuwien.ac.at/papers/malware_survey.pdf.
3. A.D. Schmidt et al., "Detecting Symbian OS Malware through Static Function Call Analysis," *Proc. 4th Int'l Conf. Malicious and Unwanted Software* (Malware 09), IEEE, 2009, pp. 15-22.
4. M. Egele et al., "PiOS: Detecting Privacy Leaks in iOS Applications," *Proc. ISOC Network and Distributed System Security Symp.* (NDSS 11), ISOC, 2011; www.iseclab.org/papers/egele-ndss11.pdf.
5. W. Enck et al., "A Study of Android Application Security," *Proc. 20th Usenix Security Symp.*, Usenix, 2011; http://static.usenix.org/events/sec11/tech/full_papers/Enck.pdf.
6. W. Enck, "TaintDroid: An Information-Flow Tracking System for Real-Time Privacy Monitoring on Smartphones," *Proc. 9th Usenix Symp. Operating Systems Design and Implementation* (OSDI 10), Usenix, 2010; http://static.usenix.org/events/osdi10/tech/full_papers/Enck.pdf.
7. T. Blasing et al., "An Android Application Sandbox System for Suspicious Software Detection," *Proc. 5th Int'l Conf. Malicious and Unwanted Software* (Malware 10), ACM, 2010, pp. 55-62.
8. W. Enck, M. Ongtang, and P. McDaniel, "On Lightweight Mobile Phone Application Certification," *Proc. 16th ACM Conf. Computer and Communications Security* (CCS 09), ACM, 2009, pp. 235-245.
9. D. Barrera et al., "A Methodology for Empirical Analysis of Permission-Based Security Models and Its Application to Android," *Proc. 17th ACM Conf. Computer and Communications Security* (CCS 10), ACM, 2010, pp. 73-84.
10. A.P. Felt, K. Greenwood, and D. Wagner, "The Effectiveness of Application Permissions," *Proc. 2nd Usenix Conf. Web Application Development* (WebApps 11), Usenix 2011; http://static.usenix.org/events/webapps11/tech/final_files/Felt.pdf.

11. G. Portokalidis et al., "Paranoid Android: Versatile Protection for Smartphones," *Proc. Ann. Computer Security Applications Conf.* (ACSAC 10) ACM, 2010, pp. 347-356.
12. B.-G. Chun and P. Maniatis, "Augmented Smartphone Applications Through Clone Cloud Execution," *Proc. 12th Conf. Hot Topics in Operating Systems* (HotOS 09), ACM, 2009, pp. 8-8.
13. I. Burguera, U. Zurutuza, and S. Nadjm-Tehrani, "Crowdroid: Behavior-Based Malware Detection System for Android," *Proc. ACM Workshop Security and Privacy in Mobile Devices* (SPMD 11), ACM, 2011, pp. 15-26.
14. J. Oberheide, E. Cooke, and F. Jahanian, "CloudAV: N-Version Antivirus in the Network Cloud," *Proc. 17th Conf. Security Symp.*, Usenix, 2008, pp. 91-106.
15. L.G. Liu, Y. Zhang, and S. Chen, "VirusMeter: Preventing Your Cellphone from Spies," *Proc. Int'l Symp. Research in Attacks, Intrusions, and Defenses* (RAID 09), LNCS 5758, Springer, 2009, pp. 244-264.

***Mahinthan Chandramohan** is a research student in the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore. His research focuses on behavior-based malware detection and analysis. Chandramohan received a BEng in computer engineering from the Nanyang Technological University. Contact him at mahintha001@ntu.edu.sg.*

***Hee Beng Kuan Tan** is an associate professor in the Division of Information Engineering in the School of Electrical and Electronic Engineering, Nanyang Technological University. His research focuses on software security and software testing and analysis. Tan received a PhD in computer science from the National University of Singapore. Contact him at ibktan@ntu.edu.sg.*

cn **Selected CS articles and columns are available for free at http://ComputingNow.computer.org.**

IEEE computer society NETWORK

IEEE Computer Society Network Webinar Series

Stay current and learn from leading industry experts as they explore important technology developments and trends Interact with presenters through a live Q&A following each live webinar.

Wizardry at Your Fingertips--Making Tough Decisions Easier with Predictive Analytics

Join us in this webcast to hear Murray Cantor and Walker Royce, two of IBM Rational's technical wizards, share some wizardry.

Date: Thursday, August 30, 2012
2:00 PM ET / 11:00 AM PT / 18:00 GMT
(Duration: 1 hour)
computer.org/webinars/08302012

IEEE computer society

IBM

Unifying Requirements and Design with a Simplified Requirements Analysis Workflow

Change is inevitable. Learn how utilizing a new workflow provided by the Rational Solution for Systems and Software Engineering simplifies the way that systems engineers and software developers can trace requirements to design information.

Date: Thursday, September 20, 2012
2:00 PM ET / 11:00 AM PT / 18:00 GMT
(Duration: 1 hour)
computer.org/webinars/09202012

Adopting Agile Methods for Safety-Critical Systems Development

Join IBM expert Bruce Douglass in this webcast to learn how agile methods can be used in the development of safety critical systems to deliver high quality results with improved predictability.

Date: Thursday, September 27, 2012
2:00 PM ET / 11:00 AM PT / 18:00 GMT
(Duration: 1 hour)
computer.org/webinars/09272012