A Survey on Security for Mobile Devices

Mariantonietta La Polla, Fabio Martinelli, and Daniele Sgandurra

***Abstract*—Nowadays, mobile devices are an important part of our everyday lives since they enable us to access a large variety of ubiquitous services. In recent years, the availability of these ubiquitous and mobile services has significantly increased due to the different form of connectivity provided by mobile devices, such as GSM, GPRS, Bluetooth and Wi-Fi. In the same trend, the number and typologies of vulnerabilities exploiting these services and communication channels have increased as well. Therefore, smartphones may now represent an ideal target for malware writers. As the number of vulnerabilities and, hence, of attacks increase, there has been a corresponding rise of security solutions proposed by researchers. Due to the fact that this research field is immature and still unexplored in depth, with this paper we aim to provide a structured and comprehensive overview of the research on security solutions for mobile devices.**

This paper surveys the state of the art on threats, vulnerabilities and security solutions over the period 2004-2011, by focusing on high-level attacks, such those to user applications. We group existing approaches aimed at protecting mobile devices against these classes of attacks into different categories, based upon the detection principles, architectures, collected data and operating systems, especially focusing on IDS-based models and tools. With this categorization we aim to provide an easy and concise view of the underlying model adopted by each approach.

***Index Terms*—Mobile Security, Intrusion Detection, Mobile Malware, Trusted Mobile.**

I. Introduction

CURRENT mobile devices (henceforth, called *smartphones*) provide lots of the capabilities of traditional personal computers (PCs) and, in addition, offer a large selection of connectivity options, such as IEEE 802.11, Bluetooth, GSM, GPRS, UMTS, and HSPA. This plethora of appealing features has led to a widespread diffusion of smartphones that, as a result, are now an ideal target for attackers. In the beginning, smartphones came packaged with standardized Operating System (OS): less heterogeneity in OS allowed attackers to exploit just a single vulnerability to attack a large number of different kinds of devices by causing major security outbreaks [1]. Recently, the number of OSes for smartphones (Symbian OS, Windows Mobile, Android and iPhone OS) has increased [2]: as shown in Table I, each mobile OS has now gained a significant market share.

Even if global sales of smartphones will pass 420 million devices in 2011 (according to a recent report by IMS research [3]), the number of mobile malware is still small compared to that of PC malware [4]. Nonetheless, we can expect malware for smartphones to evolve in the same trend as malware for PCs: hence, in the next incoming years we will face a growing number of malware. As an example, as more users download and install third-party applications for smartphones, the chances of installing malicious programs increases as well. Furthermore, since users increasingly exploit smartphones for sensitive transactions, such as online shopping and banking, there are likely to be more threats designed to generate profits for the attackers. As a proof that attackers are starting to focus their efforts on mobile platforms, there has been a sharp rise in the number of reported new mobile OS vulnerabilities [5]: from 115 in 2009 to 163 in 2010 (42% more vulnerabilities). In the same trend, there has been an increase in attention to the security issues from security researchers.

To help understanding the current security problems affecting smartphones, we review threats, vulnerabilities and attacks specific to smartphones and examine several security solutions to protect them. In particular, we survey the literature over the period 2004-2011, by focusing our attention on high-level attacks.

The paper is organized as follows. Section II introduces some background notions on mobile technologies, both for wireless telecommunication and networking standards. Section III describes different types of mobile malware, along with some predictions on future threats, and outlines the differences among security solutions for smartphones and traditional PCs. Section IV discusses current threats targeting smartphones: firstly, it analyzes the different methodologies to perform an attack in a mobile environment; then, it investigates how these methodologies can be exploited to reach different goals. In Sec. V we present security solutions, focusing on those that exploit intrusion detection systems and trusted platform technologies. Finally, Sec. VI draws some conclusions.

II. Mobile Technologies

In this section, we briefly recall some background notions on wireless and networking technologies that, even if not originally created for a mobile environment, have favored the increasing usage of smartphones.

A. Wireless Telecommunication Technologies

The most important wireless technologies targeted at mobile communications are GSM, GPRS, EDGE and UMTS.

1) GSM: Global System for Mobile communications (GSM) is the first and most popular standard in Europe for mobile telecommunication system and is part of the second-generation (2G) wireless telephone technology. Developed in 1990 by Group Special Mobile, a group created in 1982 by Conférence Européenne des administrations des Postes et des Télécommunications (CEPT), this standard enables the creation of cellular networks where mobile phones (called *mobile station* in the standard) communicate with each other

Manuscript received 17 February 2011; revised 7 November 2011.

The authors are with Istituto di Informatica e Telematica Consiglio Nazionale delle Ricerche, Pisa, Italy (e-mail: {firstname.secondname}@iit.cnr.it).

Digital Object Identifier 10.1109/SURV.2012.013012.00028

TABLE I
WORLDWIDE SMARTPHONE SALES TO END USERS BY OPERATING SYSTEM IN 3Q10 [2]

Company	3Q10 Units/1k	3Q10 Market Share (%)	3Q09 Units/1k	3Q09 Market Share (%)
Symbian	29,480.1	36.6	18,314.8	44.6
Android	20,500.0	25.5	1,424.5	3.5
iOS	13,484.4	16.7	7,040.4	17.1
Research In Motion	11,908.3	14.8	8,522.7	20.7
Microsoft Windows Mobile	2,247.9	2.8	3,259.9	7.9
Linux	1,697.1	2.1	1,918.5	4.7
Other OS	1,214.8	1.5	612.5	1.5
Total	80,532.6	100.0	41,093.3	100.0

through base stations, networks and switching subsystems. Compared to its predecessor (TACS standard), telecommunication operators can offer new services by exploiting these technologies: for instance, data transmission, digital fax, e-mail, call forwarding, teleconferencing service and Short Message Service (SMS).

2) GPRS and EDGE: These standards stem as an evolution of GSM; *General Packet Radio Service* (GPRS), also referred as 2.5 generation, was developed to improve performances of GSM network to enable users to achieve higher data rates and lower access time compared with previous GSM standard. GPRS uses packet switching mechanism (as in IP protocol) to enable the exchange of data between users. Moreover, services such as Wireless Application Protocol (WAP) and Multimedia Messaging Service (MMS) are also introduced. In this way, a variety of packet-oriented multimedia applications and services can be offered to mobile users.

Enhanced Data rates for GSM Evolution (EDGE) standard was developed in 2000 to improve the features offered by GPRS by supporting higher transmission rate and higher reliability.

3) UMTS: The *Universal Mobile Telecommunications System* (UMTS) was introduced in Europe in 2002. This standard represents the third-generation (3G) on cellular system. The transmission rate is higher than 2G and 2.5G by providing a transmission speed up to 2Mbps. Circuit switching connections are supported simultaneously with packet switching connections and users can exploit multiple services and different classes of services, such as conversational, streaming, interactive and background.

B. Networking Technologies

During the last few years, due to ease of installation and the increasing popularity of laptop computers, *Wireless Local Area Network* (WLAN) has become very popular. This technology enables devices to be linked together through wireless distribution methods and allows users to move in a local coverage area without losing their connection to the network. There are different standards that regulate communications in a WLAN. In the mobile environment, the most popular are Bluetooth and IEEE 802.11.

1) Bluetooth: Bluetooth is a standard that enables devices to exchange data over a small area through short wavelength radio transmissions. Bluetooth is a personal networking technology that enables the creation of Personal Area Networks with high levels of security. This standard, developed by Bluetooth Special Interest Group (SIG) in 1999, is aimed at providing communication between devices having these features:

- lower consumptions;
- short range of communications (1-100 meters);
- small production costs.

TABLE II
BLUETOOTH CLASSES

Class	Power (dBm)	Distance (m)
Class 1	20	100
Class 2	4	10
Class 3	0	1

As shown in Table II, there are three different classes of Bluetooth devices according to the power consumption and range of communication.

SIG defines several *profiles* to indicate different services (e.g. Generic Access Profile, GAP, or Headset Profile, HSP) and to describe the service's implementation.

2) Wireless LAN IEEE 802.11: IEEE 802.11 is a family of standards for WLAN that includes several protocols for communicating at different frequencies (2.4, 3.6 and 5 GHz). These standards can be used in two operation mode:

1) in the *infrastructure mode*, a device, referred as *Access Point* (AP), plays the role of the referee: an AP regulates the network access and coordinates the devices that are part of the network;
2) in the infrastructure-less mode (*ad hoc* mode), no referee exists and devices monitor the spectrum to gain network access.

TABLE III
802.11B AND 802.11G PROTOCOLS

Technology	Bandwidth (GHz)	Bitrate (Mbit/s)	Modulation
802.11b	2.4	5.5, 11	CCK
802.11g	2.4	6, 9, 12, 18, 24, 26, 48, 54	OFDM

The most popular protocols included in this standard are defined by the 802.11b and 802.11g protocols. As shown in Table III, the differences between these protocols are related to bandwidth, bit-rate and type of modulation (Complementary Code Keying, CCK, for 802.11b, Orthogonal Frequency-Division Multiplexing, OFDM, for 802.11g).

III. MOBILE MALWARE

This section provides a comprehensive overview of mobile malware and some predictions on future threats. Moreover, it describes the differences among security solutions targeting smartphones and PCs.

Malware is any kind of hostile, intrusive, or annoying software or program code (e.g. Trojan, rootkit, backdoor) designed to use a device without the owner's consent. Malware is often distributed as a spam within a malicious attachment or a link in an infected websites. Malware can be grouped in the following main categories, according to its features (e.g., the vector that is used to carry the payload):

- virus;
- worm;
- Trojan;
- rootkits;
- botnet.

A *virus* is a piece of code that can replicate itself. Different replica of a virus can infect other programs, boot sector, or files by inserting or attaching itself to them.

A *worm* is a program that makes copies of itself, typically from one device to another one, using different transport mechanisms through an existing network without any user intervention. Usually, a worm does not attach to existing programs of the infected host but it may damage and compromise the security of the device or consume network bandwidth.

Malware can also come packaged as a *Trojan*, a software that appears to provide some functionalities but, instead, contains a malicious program.

Rootkits achieve their malicious goal by infecting the OS: usually, they hide malicious user-space processes and files or install Trojans, disable firewalls and anti-virus. Rootkits can operate stealthily since they directly apply changes to the OS and, hence, can retain longer control over the infected devices.

Finally, a *botnet* is a set of devices that are infected by a virus that gives an attacker the ability to remotely control them. Botnets represent a serious security threat on the Internet and most of them are developed for organized crime doing attacks to gain money. Example of such attacks are sending spam, Denial-of-Service (DoS) or collecting information that can be exploited for illegal purposes (DoS attacks targeting smartphones are described in detail in Sec. IV-B3).

Mobile malware can spread through several and distinct vectors, such as an SMS containing a link to a site where a user can download the malicious code, an MMS with infected attachments, or infected programs received via Bluetooth. The main goals of malware targeted at smartphones include theft of personal data stored in the phone or the user's credit.

Examples: [6] details a Trojan for Android smartphones, named Trojan-SMS.AndroidOS.FakePlayer.b, which masquerades as a media player and requires the user to manually install it. This fake application is downloaded from an infected webpage in order to view adult content videos. The installation file is very small in size and during installation the application asks the user permissions to send SMS messages. Once the installation has finished, if the user launches the fake application, the Trojan begins sending SMS messages to a premium rate number without the user's knowledge. These messages result in costly sums being transferred from the user's account to that of the cybercriminals.

[7] develops a kernel-level Android rootkit in the form of a loadable kernel module that can open a shell for the attacker (using a reverse TCP connection over 3G/Wi-Fi) upon the reception of an incoming call from a trigger number. This results in full root access on the Android device. In this way, an attacker can read all SMS messages on the device, incur the owner with long-distance costs or even potentially pinpoint the mobile device's exact GPS location.

[8] analyzes three sample rootkits to show how smartphones are as vulnerable as traditional computers to rootkits. In fact, smartphone rootkits can access several distinctive interfaces and information that are unique to smartphones, such as GPS, battery, voice and messaging, which provide rootkits writers with new attack vectors to compromise either the privacy or the security of end users. The first proposed sample rootkit allows a remote attacker to stealthily listen into (or record) confidential GSM conversation using the user's infected smartphone. The second attack aims at compromising the victim's location privacy by requiring the infected smartphone to send a text message to the remote attacker including the user's current GPS location. The final sample attack exploits power-intensive smartphone services, such as those offered by GPS and Bluetooth, to exhaust the battery on the smartphone.

As an example of smart malware, recently a multifarious malware for iOS devices has been designed and implemented by [9] (*iSAM*). iSAM incorporates six different features of malware:

1) propagation logic;
2) botnet control logic;
3) collect confidential data stealthily;
4) send a large number of malicious SMS;
5) denial of application services;
6) denial of network services.

Moreover, iSAM is able to connect back to its botmaster server to update the programming logic, to implement commands, and to perform a synchronized, distributed, attack.

Table IV reports some notable examples of mobile malware.

A. Evolution of Mobile Malware

Several papers discuss the evolution of mobile malware: for instance, [10] describes the evolution of malware on smartphones from 2004 to 2006. For an overview on the state of the art of mobiles viruses and worms up to 2006, see Hypponen [11]. In the period 2004-2008, the number of types of mobile malware has increased significantly: as of March 2008, F-Secure has categorized 401 distinct types of mobile malware worldwide, whereas McAfee has counted 457 kinds of mobile malware [12]. In the period 2004-2010, 517 families of mobile viruses, worms and Trojans have been categorized by F-Secure [13]. For a complete list of mobile malware in the period 2000-2008 see [14]; see [15] for mobile malware that spread from January 2009 to June 2011.

The first virus (a Trojan) for mobile phones, developed for Palm devices [16], was discovered in 2000 by F-Secure [17]. In June 2004, the first worm that could spread through mobile phones with Symbian OS appeared: this worm, called Cabir [18], was only a prototype developed by the 29A Eastern European hacker group. Cabir is considered the first example of malicious code that can spread itself exploiting the networking technologies on mobile devices (in this case, Bluetooth) to infect other devices.

TABLE IV
MOBILE MALWARE EXAMPLES

Name	Time	Type	Method of Infection	Effects	OS
Liberty Crack	2000	Trojan	Pretend to be a hack	Remove third-party software	Palm OS
Cabir	2004	Worm	Bluetooth connection and copies itself	Continuous scan of Bluetooth, drain phone's battery	Symbian OS
Dust	2004	Virus	File Infector	Infect all executables in root DIR	Windows Mobile
Brador	2004	Trojan	Copy itself in to the startup folder	Open a backdoor	Windows Mobile
Mosquitos	2004	Trojan	Embedded in a game	Send SMS to premium-rate numbers	Symbian OS
Skulls	2004	Trojan	Vulnerability in overwriting system files	DoS	Symbian OS
MetalGear	2004	Trojan	Vulnerability in overwriting system files	Disable virus scanner	Symbian OS
CommWarrior	2005	Worm	Replicates via Bluetooth and MMS	MMS charging	Symbian OS
Doomboot	2005	Trojan horse	Doom 2 video game	Prevents booting and installs Cabir and CommWarrior	Symbian OS
Lasco	2005	Virus	File infection	Add itself to install packages	Symbian OS
Locknut	2005	Trojan	Vulnerability in OS	Create entries for a new application	Symbian OS
Feakk	2005	Worm	SMS message	Send SMS to all contacts	Symbian OS
Cardblock	2005	Virus	Fake SIS application	Encrypt memory card with a random password	Symbian OS
CardTrap	2005	Cross-Platform Virus	Auto-start of removable storage	Copy Wukill on the phone	Symbian/Windows OS
Blankfont	2005	Trojan	Replace font files	Fonts not displayed	Symbian OS
Crossover	2006	Cross-Platform Virus	CIL vulnerabilities	Copy to/from mobile/PC	Windows/Mobile OS
Letum	2006	Worm	E-Mail spreading	Infect registry	Windows Mobile
Fontal	2006	Trojan	Vulnerability in overwriting system files	Device not restart after reboot	Symbian OS
Mobler	2006	Cross-Platform Worm	Dropping Mechanisms	Disable antivirus and infect removable storage	Symbian/Windows OS
Redbrowser	2006	Trojan	Fake Browser	Send SMS continuously	OS-Independent (J2ME)
Wesber	2006	Trojan	Fake Browser	Send SMS to premium-rate numbers (Russia only)	OS-Independent (J2ME)
Acallno	2006	Spyware	Fake Commercial Software	Gather and send information about user's activities	Symbian OS
Lasco	2007	Worm	A worm that spreads over Bluetooth networks	Searching and infecting other phones	Symbian OS
Feak	2007	Worm	Proof-of-concept worm	Sending SMS to contact list with URL	Symbian OS
Flocker	2007	Trojan	It claims to be an ICQ application to trick the user	Sending SMS to a hard coded phone number	Symbian OS
Beselo	2008	Worm	Via MMS and Bluetooth fake application	MMS charging	Symbian OS
InfoJack	2008	Trojan	Attach itself to installation packages	Disable security settings	Windows Mobile
Pmcryptic	2008	Worm	Memory card spreading	Dialing premium-rate numbers	Windows Mobile
Yxe	2009	Worm	SMS containing malicious URL	Send contact lists to external server	Symbian OS
Yxes	2009	Worm/Botnet	SMS containing malicious URL	Send contact lists to external server	Symbian OS
Ikee	2009	Worm	Scanning a IP ranges and SSH	Alter wallpaper	iPhone
FlexiSpy	2009	Spyware	Fake Application	Tracking/log of device's usage	Symbian
Curse of Silence	2009	SMS Exploit	Vulnerabilities in e-mail parsing	Disable SMS functionalities	Symbian OS
ZeuS MitMo	2010	Worm	Fake SMS	Steal bank account information	Cross-Plafrom
iSAM	2011	Multifarious malware	Scanning IP and connecting to SSH	Collect private information, send malicious SMS, DoS	iPhone

Recently, a growing number of viruses, worms, and Trojans that target smartphones have been discovered. As we have already pointed out, the reason of the growing number of mobile malware is due to the widespread use of smartphones. Furthermore, we have to consider that most of the smartphones lack any kind of security mechanisms and are not well prepared against new threats. Within the 2006-2008 period, security issues exploiting several attack vectors have increased [19], and there has been a dramatic escalation of complex attacks targeting lower-level device functionality: early security threats have turned into sophisticated, profit-oriented, attacks driven by experienced criminals.

A discussion of mobile malware, based on OSes and infection routes, is presented in Töyssy and Helenius [20] that describe and cluster mobile malware with respect to:

- the *OS*: Symbian, Palm OS, Linux, Windows Mobile;
- the *infection routes*: MMS, Bluetooth, IP connections via GPRS/EDGE/UMTS, WLAN, copying files, removable media.

The authors propose some prevention solutions and countermeasures, by considering:

- the *users*, which have to be educated to utilize the device in a secure way;
- the *software developer*, which can develop security protection targeted at smartphone;
- the *network operator*, which can enhance the network infrastructure with mechanisms to avoid intrusions;
- the *phone manufacturers*, which should update the devices automatically so that for attackers it would be harder to exploit security holes;
- new *epidemiological models*, to forecast if an already detected virus can initiate an epidemic.

Similar solutions are also proposed in [21], where the authors remark that the protection from malicious code should be implemented at every possible entry point to the network.

For a comprehensive discussion on evolution of mobile malware, see [22].

B. Predictions and Future Threats

Since the first smartphone, discussions about threats targeting these devices have proliferated: the first threat against Symbian and Palm (such as Liberty Crack Trojan [23]) never became widespread and remained just a proof-of-concept. Even if security experts foresee massive attacks to come out at any time, yet they never seem to happen: nonetheless, McAfee Labs [24] predicts that 2011 will be a turning point for threats to smartphones. In fact, in the last months, several new threats to smartphones have emerged: rootkits for the Android platform, remote jail-breaking exploits for the iPhone, and the arrival of ZeuS [25], a largely distributed banking Trojan/botnet. The widespread adoption of smartphones into business environments combined with these attacks is likely to cause the explosion that experts have long anticipated.

According to [26], in the near future cybercriminals will focus their attention on iPhone and Android platforms (this is also confirmed by Cisco [27] and Panda [28]). Symantec [5] explains the state of cybercrime on smartphones by its return on investment: firstly, the installed base of smartphones has grown to an attractive size and they run sophisticated OSes that come with the inevitable vulnerabilities; secondly, Trojans hiding in legitimate applications sold on application stores provides a simple and effective propagation method. Hence, what is currently missing is the ability to turn all this into a profit center.

In March 2011, Google reported [29] that it had removed several malicious Android applications from the Android Market and, in some cases, deleted them from users' smartphones remotely. The Android marketplace is not very closely monitored, since it adopts the "anything goes" philosophy. This, combined with the current buzz around new smartphones running Android, may make the platform more attractive to cybercriminals [30]. Gostev [26] believes there is also a strong probability that malware may soon be found in products available through Android Market. As an example, many legitimate applications can ask for, and typically be granted, access to a user's personal data and authorization to send SMSs and make calls. Hence, this places the reliability of the entire Android security concept in doubt. This is confirmed by Juniper Networks [31] that reports that, since Summer 2010, Android malware has increased by 400%.

As [26] points out, no significant malware events occurred that targeted iPhone and which could be compared to the Ikee worm incident of 2009. However, several concept programs were created for this platform in 2010 that demonstrated techniques that could be used by cybercriminals. As an example, *SpyPhone* allows unauthorized access to information about the user's iPhone device, such as her location, interests, friends, preferred activities, passwords and web search history. This data can then be sent to a remote server without the user's knowledge or consent. This functionality can be hidden within an innocuous-looking application.

[27] reveals that in July 2010 the U.S. Library of Congress added jailbreaking to its list of actions that do not violate copyright protections, hence leaving iPhone users free to unlock their devices and download applications not authorized by Apple. Only a week after, *JailbreakMe* 2.0 appeared, a tool that makes it easier for users to jailbreak their phones. The advent of this tool also revealed a significant security flaw in the iOS 4 that could leave users with jailbroken phones more likely to be attacked by hackers willing to take control of the smartphones.

As pointed out by [32], another feature to consider is the spreading of mobile virus to desktop platforms, e.g. due to devices that are already compromised or tampered with coming off the shelves. As an example, USB devices are responsible for the spread of auto-run malware, while the *Conficker*/*DOWNAD* worm contained a propagation capability that used removable drives to increase spread.

It is important to underline that evolution of malware is a continuous race between attackers and defenders: both use the same programming methods, tools and resources either to create a malware or to develop an intelligent malware detection mechanism. A further aspect to be considered concerns the observation of new forms of malware in a testbed environment to predict their behavior: as an example, [33] presents *Mobile Agent Malware Simulator (MAISim)*, a framework that uses the technology of mobile agents for simulation of various types of malicious software (viruses, worms, malicious mobile code) for smartphones.

Furthermore, since the risks at which smartphones are exposed depend also on how they are used, we have to distinguish strictly personal use of a device from uses that also involve business. As suggested in [34], future threats in a mobile environment may affect different assets, such as:

- personal data;
- corporate intellectual property;
- classified information;
- financial assets;
- device and service availability and functionality;
- personal and political reputation.

Finally, Milligan and Hutcheson [35] discuss some risks, threats and countermeasures for smartphones. Some examples of future risks associated with a smartphone include:

- data leakage resulting from device loss or theft;
- unintentional disclosure of data;
- attacks on decommissioned devices;
- phishing attacks;
- spyware attacks;
- network spoofing attacks;
- surveillance attacks;
- diallerware attacks;
- financial malware attacks;
- network congestion.

C. Mobile Security Versus Personal Computer Security

Despite the similarities between smartphones and PCs, there are several notable differences concerning security. Firstly, we have to consider that malware authors can make money from their illicit activities more easily on smartphones than in desktop environments, e.g. due to premium-rate numbers (Felt et al. [36] classify premium-rate calls/SMS as the second most common behavior found in nearly 50% of recent malware[1]). Secondly, since any event generated by the smartphone has (usually) a cost invoiced by the network operator, from the point of view of the user, the network operator is considered responsible of charging costs even if the event is generated by malware.

We also have to consider the user's point of view in a mobile environment: to this end, Botha et al. [37] explore the availability of security mechanisms from the perspective of a user who wishes to use desktop-based security mechanisms in a mobile environment. The authors remark that a main difference between smartphones and PCs is that the former is usually a personal device and some issues, such as user authentication, device configuration and content protection, need to be dealt differently. As an example, in the case of user authentication, users face several difficulties when moving from desktop to smartphone. Furthermore, there is a trade-off between security and usability, since many solutions that are used on PCs cannot be applied on smartphones.

Compared to common PCs, the basic security principles of smartphones are quite different. The security problem on smartphones originates particularly from the integration process: nowadays, a single device hosts multiple technologies that allow users to access the Internet from any place at any time. Furthermore, smartphone-specific services often require complex software and infrastructures and expose these devices

[1]Two further common behaviors are stealing/selling users' information and sending SMS spam, which are found in, respectively, 60% and 17% of the analyzed malware.

to attacks. As described in [38], five key aspects distinguish mobile security from conventional computer security:

- *mobility*: each device comes with us anywhere we go and; therefore, it can be easily stolen or physically tampered;
- *strong personalization*: usually, the owner of device is also its unique user;
- *strong connectivity*: a smartphone enables a user to send e-mails, to check her online banking account, to access lot of Internet services; in this way, malware can infect the device, either through SMS or MMS or by exploiting the Internet connection;
- *technology convergence*: a single device combines different technologies: this may enable an attacker to exploit different routes to perform her attacks;
- *reduced capabilities*: even if smartphones are like pocket PCs, there are some characteristic features that lack on smartphones, e.g. a fully keyboard.

The limited resources of a smartphone are the most obvious difference with a PC. The main limiting factors are CPU and memory. These two factors limit the sophistication of possible security solutions: for example, complex intrusion detection algorithms that work for real-life applications on PCs cannot be easily transferred to smartphones in the near future (see [39] for a distributed solution that tries to circumvent this problem). In addition, a unique characteristic of smartphones is the battery, which severely limits the resources available for a security solution; therefore, it is highly important that a security solution does not constantly drain large portions of available CPU time to avoid battery exhaustion [40].

As an example of the limited resources of a smartphone, according to [41], the ClamAV antivirus engine available for Nokia device requires about one minute of processing time to initialize the signature database and at least 40 MB of memory. To reduce this overhead, the paper proposes a model where mobile antivirus functionalities are moved to an off-device network service hosting several malware detection engine. This architecture make it possible to significantly reduce on-device CPU, memory and, most importantly, power-resource.

As [42] points out, since smartphones have strict resource constraints both in computational capabilities and power consumption, some computationally expensive algorithms for detecting sophisticated threats, such as those implemented by behavioral detection engines, are simply infeasible to be deployed on current smartphones due to their heavy-weight resource requirements. This means that adapting traditional approaches for malware detection might be infeasible for mobile environments as they consume a significant amount of resources and power.

Moreover, a further security threat for smartphones stems from the fact that wireless medium is, by nature, prone to eavesdropping and, therefore, communication confidentiality cannot be taken for granted. Threats to user privacy in a mobile environment are different from those performed on PCs because, on smartphones, sensors (e.g. microphones) are not optional and can be used illicitly to sniff user's private data. In addition, since mobile applications extensively use, and depend on, sensors and users carry smartphones wherever they go, this increase the opportunities to compromise the privacy of mobile users. Using access control techniques, like those used in PC environments, is not appropriate because it is necessary to take into account also the context in which sensors are used. Furthermore, these attacks work even when the user is not interacting with the mobile phone.

Further discussions on the subject can be found in [43].

IV. Attacks on Mobile Devices

In the following sections, we discuss several kinds of attacks against smartphones. We firstly detail the possible methodologies to perform an attack in a mobile environment and, for each kind of attack, we provide a real example. Secondly, we show how these methodologies can be exploited to reach different goals.

A. Methodologies of the Attacks

The distinct methodologies to perform attacks against smartphones are categorized using the following classes:

- wireless;
- break-in;
- infrastructure-based;
- worm-based;
- botnet;
- user-based.

1) Wireless Attacks: There are many different kinds of wireless attacks against smartphones, especially those targeting personal and sensitive data. The most common attack is eavesdropping on wireless transmissions to extract confidential information, such as usernames and passwords. Wireless attacks can also abuse the unique hardware identification (e.g., wireless LAN MAC address) for tracking or profiling the owner of the device. Finally, malware often exploits Bluetooth as a medium to speed up its propagation.

[44] discusses security problems in wireless environments and presents the current research activities. A comprehensive review of Bluetooth attacks affecting smartphones can be found in [45]. Some studies for preventing this class of attacks are proposed in [44, 46, 47, 48].

Example - Cabir: Cabir is a worm that propagates through Bluetooth. This worm consists of a message containing an application file, `caribe.sis`, that seems like a Security Manager utility. If installed, the worm uses the device's native Bluetooth functionality to search for other Bluetooth-discoverable devices. Then, the worm attempts to send infected SIS files to the discovered devices as well.

2) Break-in Attacks: Break-in attacks enable the attacker to gain control over the targeted device by exploiting either programming errors, e.g. to cause buffer overflows, or format string vulnerabilities. Typically, these attacks are used as a stepping stone for performing further attacks, such as over-billing attacks or data/identity theft.

Some studies for preventing this class of attacks are proposed in [49, 50].

Example - Doomboot.A: This Trojan installs corrupted system binaries into the `C:\` drive of the device. The corrupted binaries contain further Trojans, as CommWarrior, which are also installed on the device.

3) Infrastructure-based Attacks: Since the services provided by the infrastructure are the basis for essential smartphone functionalities, such as placing/receiving calls, SMS and e-mail services, the economic and social impact of these attacks may be very large, such as the one discussed in [51]. [52] evaluates the security impact of the SMS interface on the availability of the cellular phone network. As an example, if an attacker is able to simultaneously send messages through the several available portals into the SMS network, the resulting aggregate load can saturate the control channels and, therefore, block legitimate voice and SMS communications. The authors demonstrate that an attacker that injects text messages from the Internet can deny voice service in a metropolitan area using hit-lists containing as few as 2,500 targets with little more than a cable modem.

a) GPRS: Since the GPRS architecture is built on the GSM infrastructure, it uses a security architecture based upon the security measures already adopted by GSM (for a review of DoS attacks and confidentiality threats in GSM networks, see [53]). Attacks against GPRS can target the device, the radio access network, the backbone network, and the interfaces connecting GPRS networks with each other or with the Internet. The results of these attacks can be the compromise of end-users security, over bill users, the disclosure or alteration of critical information, the services unavailability, or the network breakdown. We have also to consider that GPRS is more exposed to attackers compared to GSM because it uses the IP technology, which is rather vulnerable.

Attacks against GPRS can be active and passive: *active attacks* requires a direct intervention of the attacker to listen, modify, and inject data into the communication channel. Furthermore, if the attacker is not part of the GPRS, the attack can be defined external; on the other hand, the attack is defined as internal. A *passive attack* happens when an attacker taps on a communication channel between two nodes without disturbing the communication to discover some valuable information about the data or control messages.

As described in [54], there are five sensitive area in GPRS security that can be exploited to perform an attack:

1) *the mobile station (MS) and the SIM-card*: the results of these attacks may be the monitoring of the MS usage, the downloading of unwanted files, the placing of unwanted calls. Attacks on the SIM-card are primary based upon the secret key, which is stored in the SIM-card of the MS. When an attacker retrieves this key, she can intercept data exchanged, or clone the original SIM card;
2) *the interface between the MS and the SGSN (Serving GPRS Support Node)*: an attacker can perform attacks like DoS or Man-in-the-Middle[2]. In DoS a malicious third party can jam user data and signaling traffic using special devices called jammer, or induce specific protocol failures, or masquerade as network elements;
3) *the GPRS backbone network*: they refer to both IP and Signaling System 7 (SS7) technologies, which convey user data and signalling information. Once a malicious MS gets access to the GPRS network may perform various attacks, such as DoS, IP spoofing, compromise of privacy or send large amounts of data to users;
4) *the packet network that connects different operators*: the Gp Interface provides connectivity between GPRS networks that belong to different operators and supports users roaming. The security targets are the availability of resources and services, the authentication and authorization of users and actions, and the integrity and confidentiality of the data transferred;
5) *the Internet*: the Gi interface connects the GPRS network to the Internet and service providers that provide services to mobile subscribers. Due to the fact that the Gi interface may carry any type of traffic, the GPRS network elements and the mobile subscribers can be exposed to a variety of threats found on Internet;

Another type of attacks against the SIM-cards of GSM/GPRS is side-channel attacks that allow an attacker to obtain sensitive information from side-channels, as described in [55].

b) UMTS: The UMTS security architecture defines a set of procedures to achieve increased message confidentiality and integrity during their communication. At the kernel of its security architecture lies the user authentication mechanism, known as Authentication and Key Agreement (AKA). Authentication in UMTS is based on a 128-bit symmetric secret key, namely Ki, which is stored in the user's tamper-resistant Universal Integrated Circuit Card (UICC) and in the corresponding Home Location Register (HLR) of the user's Home Network (HN).

[56] discusses several vulnerabilities of the UMTS security architecture that can be exploited by malicious attackers to launch DoS attacks. Typically, an attacker tries to access unprotected control messages in order to manipulate specific procedures. The expected results varies from lower Quality of Service to DoS. Some examples of such an attack are:

- *dropping ACK signal*: an attacker monitors for TMSI Allocation Command messages and then drops any following TMSI Allocation Complete message to repeatedly force the creation of new TMSIs that, eventually, will cause DoS to all the users in that area;
- *modification of unprotected Radio Resource Control (RRC) messages*: an attacker substitutes a valid RRC Connection Setup Complete with a RRC Connection Reject message to cause a lower Quality of Service or a DoS for the end-users;
- *modification of the initial security capabilities of MS*: an attacker modifies a RRC Connection Request message to trigger the termination of the connection. For example, she can cause a serious damage by creating a very large number of simultaneous connection requests;
- *modification of periodic authentication messages*: this happens if the Radio Network Controller (RNC), on receiving a tampered message, releases the connection, disconnecting the MS;
- *SQN synchronization*: an attacker can ask for a resynchronization procedure to be executed simultaneously for large numbers of users, and repeatedly, to greatly overstress the HLR;

[2]In Man-in-the-Middle attacks, an attacker can impersonate both a false base station to a victim MS and, at the same time, the victim to the base station.

- *EAP-ALA originated DoS*: an attacker could spoof an EAP-Response/AKA-Client-Error message and send it to the EAP-Server to force it into halting the protocol or she could spoof an EAP-Response/AKA-Synchronization-Failure notification to force the server to trigger the costly resynchronization procedure.

Some studies for detecting infrastructures-based attacks are discussed in [57, 58, 59].

Examples: [60] describes an attack where a malicious user impersonates a valid GSM base station to a UMTS subscriber and, as a result, she can eavesdrop on all mobile-station-initiated traffic. [61] investigates the feasibility of a DoS attack by taking advantage of a particular flow found in the UMTS security architecture. The proposed attack involves the modification of the RRC connection Request Message that includes the user's equipment security capabilities. This message is not integrity-protected: in case of mismatch, the connection will terminate, but during this process enough resources will have been already consumed at both sides.

[51] characterizes the impact of large scale compromise and coordination of mobile phones in attacks against core networks, by demonstrating that a botnet composed of about 12,000 compromised nodes can degrade the service of an area-code sized regions by 93%. The sample attack attempts to prevent legitimate users of a cellular network from sending or receiving calls or text messages: the attack is carried by an attacker that can control a set of compromised smartphones and overwhelm a specific HLR with a large volume of traffic, so that legitimate users relying on the same HLR are unable to receive service as their requests are dropped.

[62] discusses the types of damage that can be caused to smartphones, such as privacy violation, identity theft and emergency call center distributed DoS attacks.

4) Worm-Based Attacks : The main features that characterize attacks based upon worms are:

- transmission channel;
- spreading parameters;
- user mobility models.

a) Transmission Channel: Smartphones are usually equipped with several connectivity options and, hence, offer many possible routes for infection vectors, such as:

- downloading infected files while surfing the Internet;
- transferring malicious files between smartphones using the Bluetooth interface;
- synchronizing a smartphone with an infected computer;
- accessing an infected memory card;
- opening infected files attached to MMS messages.

In the last years, Bluetooth has become one of the most popular wireless protocols and the class of malware that uses Bluetooth connection to infect devices is growing. Bluetooth worms are different from other classes of worms: the most notable difference is that, to spread the worm, a Bluetooth infection requires that the infection source and the victim are located very close to each other, i.e. in a diameter of 20/30 meters.

b) Spreading Parameters: In addition to infecting the device, worms can also attack the communication network itself. In this scenario, worms not only compromise users' ability to use their smartphones but the networks as well.

Worms that exploit messaging services (SMS/MMS), as their preferred infection routes, are potentially more virulent, in terms of speed and area of propagation, than Bluetooth ones. In fact, these worms can be easily sent out using just one click and can infect any smartphone in any part of the world with a larger chance of success of propagation.

c) User Mobility Models: Compared with the Internet, mobile phone networks have very different characteristics in terms of topologies, services, provisioning and capacity, devices and communication patterns. These features also characterize the way new types of mobile worms propagate: the most important one is that they do not require Internet connectivity for their propagation and, therefore, can spread without being detected by existing security systems. Hence, mobile worms can infect several devices using *proximity attacks* against vulnerable devices that are physically nearby.

To model the propagation of these worms, two steps are required:

1) build a model that precisely describes how devices meet each other;
2) understand how malicious code exploits both the mobility of the users and the capacities of the networks.

The dynamics of proximity propagation depend upon the mobility dynamics of a population in a specific geographic region. Unfortunately, an ideal methodology for modeling user mobility does not exist: traces of mobile user's contacts reflect actual behavior, but they are difficult to generalize and only capture a subset of all contacts due to a lack of geographic coverage.

To model epidemic spreading of malware via proximity-based, point-to-point wireless links, Mickens and Noble [63] introduce a framework called *probabilistic queuing* to deal with node mobility. To capture the skewed connectivity distributions of mobile networks, the model represents different connectivity levels as distinct queues. Each queue represents a separate epidemiological population. A probabilistic queuing model is proposed to explicitly account for both node velocities and the non-homogeneous connectivity patterns induced by mobility of devices.

The problem of proximity attacks of smartphone is also discussed in [64] where the authors introduce an individual-based model and build analytical expressions for contact-rate calculation and worm transmission. In [65], an event-driven simulator that captures the characteristics and constraints of mobile phone networks is proposed. The simulator models realistic topologies and provisioned capacities of the network infrastructure. The goals of this model are:

- model malware propagation in networks under realistic scenarios to characterize its speed and severity;
- understand how network provisioning impacts propagation and how propagation impacts the network;
- highlight the implications for network-based defenses against malware.

Some further solutions to withstand worm attacks, which are based upon mobility models for Bluetooth worm propagation, are also studied in [66, 67, 68]. See also [69] for analytical models for epidemics in mobile networks.

Examples: [70] and [71] investigate whether a large-scale Bluetooth worm outbreak is viable in practice. The authors use trace-driven simulations to examine the propagation dynamics of a Bluetooth worm in a large population, by showing that the worm outbreak's start time is very important (e.g., during week-end). Their results suggest that a worm exploiting a Bluetooth could spread very quickly. As defense solution, the authors suggest to locate monitoring points in high-traffic locations.

[72] investigates the effort required to port a smartphone worm for Windows Mobile. The authors were able to build a prototype worm that could spread autonomously if a vulnerability exists in a Windows Mobile service reachable over the network. The authors state that the lower bound of the time required to both build a worm toolkit and find a vulnerability in the network protocol stack of Windows Mobile requires approximately 14 weeks of full time work.

[73] analyzes in detail the first polymorphic worm that affects smartphones running Windows CE platform on ARM processors, known as `WinCE.Pmcryptic.A`. The worm spreads by generating new polymorphic copies of itself each time and can execute several unwanted actions on a compromised smartphone, including calls to toll numbers.

5) Botnets: Until recently, mobile networks have been relatively isolated from the Internet, so there has been little need for protecting them against attackers trying to create botnets. However, this situation is rapidly changing since mobile networks are now well integrated with the Internet. Hence, threats on the Internet will migrate over the mobile networks (and vice-versa), including botnets, since smartphones can be infected by malware they can be turned into a botclient easily [74].

a) Command-and-Control: The command-and-control (C&C) network, used to remotely propagate messages, tasks, updated payload among the bots and the botmasters (and vice-versa), can be built out using Bluetooth, SMS messages, the Internet (e.g., HTTP), peer-to-peer (P2P) or any combination of them.

Bluetooth C&C: [75] investigates the challenges of constructing and maintaining mobile-based botnets communicating via Bluetooth. By using simulations on publicly available Bluetooth traces, the authors demonstrate that C&C messages can propagate to approximately 66% of infected nodes within 24 hours of being issued by the botmaster. To reduce the amount of traffic observable by the provider to achieve stealthiness, in the developed framework only a small subset of bots (those with the highest degree) communicate directly with the botmaster through cellular channels (e.g., SMS, cellular data). These nodes are selected via their relative frequency of contact with other infected devices: whenever infected smartphones pass within range of each other, they record the identity of the other device. After reaching some threshold set by the botmaster, nodes with a high degree of connectivity send their contact logs to the botmaster which is informed of (i) which devices are under his control (ii) which nodes can help disseminating commands rapidly. The botmaster also disseminates commands and updated payloads through this hierarchical structure by contacting the seed nodes. Due to their high degree of connectivity, these nodes can deliver the payload to the largest number of infected nodes directly and no interaction with the botmaster is required.

SMS C&C: [76] proposes the design of a proof-of-concept mobile botnet resilient even to disruption. The botnet is built out of three components:

1) vectors to spread the bot code to smartphones;
2) a channel to issue commands;
3) a topology to organize the botnet.

Within the testbed mobile botnet, all C&C communications are carried out using SMS messages. To hide the identity of the botmaster, there are no central servers dedicated to command dissemination, since they could be easily identified and removed. Instead, a P2P topology is exploited to allow botmasters and bots to publish and search for commands in a P2P fashion, making their detection and disruption much harder.

Hybrid C&C: [77] shows that it is easy to create a fully functional mobile phone botnet out of Apple's jailbroken iPhone by discussing the design, implementation and evaluation of an iPhone-based mobile botnet. The authors firstly discuss an SMS-based botnet that is then improved with HTTP to reduce the number of SMS messages that need to be sent for controlling the bots. Finally, the authors show how powerful such a botnet could be if attackers combine P2P (namely, Kademlia) with SMS-HTTP hybrid approach.

Examples: Porras et al. [78] recall how, at the end of 2009, some users of jailbroken iPhones began seeing pop-up windows that redirected the victim to a website where a ransom payment was demanded to remove the malware infection. The vulnerability affected several jailbroken iPhones that have been configured with a SSH service with a known default root password. By scanning some IP addresses from the Internet for vulnerable SSH-enabled iPhones, an attacker could upload a very simple ransomware application to several iPhone users. By exploiting this vulnerability, some weeks after, a second iPhone malware (iKee.A) converted the iPhone into a self-propagating worm to infect other iPhones. This time, the worm succeeded in infecting more than 20,000 victims within a week. Some time later, a new malware (iKee.B), similar to iKee.A, was found: in addition to self-propagation, the iKee.B bot client application introduces a C&C check-in service that enables the botmaster to upload and execute shell commands on all infected iPhone bot clients. This service allows the bot to evolve or to redirect infected iPhones to new C&Cs located anywhere on the Internet. The iKee.B also incorporates a feature to exfiltrate the entire SMS database from the victim's iPhone.

[79] provides an overview of Yxes, one of the first malware for Symbian OS 9 and a first step towards a mobile botnet. Once installed, using a valid signed certificate, the main tasks of the malware include getting the IMEI and the IMSI of the mobile phone, parsing contacts, killing unwanted applications and propagating. To this end, the malware begins its propagation phase by sending an SMS to each new victim with a link to a malicious server where the victim can download the malware. One of the main issues concerning Yxes is that its code does not exploit any particular Symbian OS vulnerability, but only uses functions of its API in a smart way. For this reason, the author concludes by remarking that the concept

of capabilities, introduced by Symbian, fails to stop malicious intents, for two reasons: (i) cybercriminals manage to have their malware signed whatever capability they request; (ii) capabilities only grant authorizations for given actions but cannot take into account a context or an intent.

Attackers have also taken a popular legitimate application and added additional code to it, such as with *Pjapps* [80]. Several users noticed that something was wrong when the application was requesting more permissions than should have been necessary: Pjapps attempted to create a bot network out of compromised Android devices. This attack clearly demonstrates that attackers are definitely considering smartphones as a platform for cybercrime.

6) User as an Attack Vector: User-based attacks contain every exploit that is not of technical nature. Many of today's mobile malware samples are not based on a technical vulnerability, but trick the user into overriding technical security mechanisms [40]. This is an important class of vulnerabilities and several studies have been performed to evaluate the security knowledge of the average user; in particular they discuss the purpose of all the security mechanisms implemented by the smartphones if the average user does not understand them. Quite often, if the user clearly understands how to use one specific mechanism, she might find it difficult to understand another, possibly new and updated, mechanism.

As [81] points out, very often social engineering attacks trick a user to override technical security mechanisms, as in the case of abuse of trust relationships, which might happen when a malware access the address book of the victim and send itself to the contacts that trust the infected user. In another scenario, a user cannot distinguish if a feature is a legitimate functionality or an imitated one, e.g. in case of a Bluetooth message with malicious content.

A survey conducted by Sophos [30] asked users whether their smartphone was encrypted: 26% percent of users replied that their data was encrypted, 50% said they were not protected in the event of theft or loss of the device, and 24% of users were not sure whether their smartphone was encrypted. These results show that further education on the security dangers of smartphones is required.

Examples: Trojan-SMS.AndroidOS.FakePlayer.b is a Trojan for Android that requires the user to manually install it [6] and asks the user permissions to send SMS messages. Hence, the user has to activately participate in the installation process. Once the installation has finished, if the user launches the fake application, the Trojan begins sending SMS messages to a premium rate number without the user's knowledge.

ZeuS MitMo [25] exploits social engineering to infect smartphones and performs online banking operations. To this end, an attacker sends an SMS with a link to a malicious mobile application to the user: then, if the user installs it, the attacker can read the SMS message used as a second authentication factor by the online banking service to access the user's account.

B. Goals of the Attacks

In this section, we discuss in detail the goals of the attacker, which can be:

- privacy;
- sniffing;
- denial of service;
- overbilling.

Moreover, for each class of attack we give some pointers to related works (which will be discussed in detail in Section V) that propose either a model or a mechanism to defend against these attacks.

1) Privacy: Privacy attacks of smartphones concern situations in which integrity and confidentiality are corrupted, e.g. when they get lost or stolen. In fact, due to their small dimensions, smartphones can be stolen or lost more often than laptop computers. During the period in which the device is not available for the legal owner, it may be possible that someone installs a spyware on the phone; furthermore, someone can read personal data, as contact list or messages.

[82] announces that a researcher has detailed a proof-of-concept method to steal data from an Android smartphone using a combination of cross-site scripting and Javascript. [83] reports how Apple is being sued for allegedly letting mobile apps on the iPhone and iPad send personal information to advertising networks without the consent of users. Finally, [84] presents an overview of recent iPhone privacy issues and considers the unmodified devices by examining what sensitive data may be compromised by an application downloaded from the App Store. Next, it explains how a malicious application could be crafted to fool Apple's mandatory reviews in order to be accepted on the App Store. Finally, it discusses several attack scenarios, suggesting improvements and basic recommendations.

A further issue, related to Digital Right Manager (DRM) protection, is discussed in [85]. The authors propose a framework for personal DRM for Motorola smartphones, where the user can place DRM protection and keeps control over her personal content. To this end, the personal DRM system enables users to define control and generate licenses on custom content and to securely transfer them to other smartphones. A user is then able to define and restrict the intended audience and ensure expiration of the content as desired. Moreover, compatible DRM devices are able to automatically detect each other and exchange credentials.

Another fundamental topic related to privacy is *location awareness*, i.e. the ability to determine geographical position. Even if this feature has significant benefits, it raises some important privacy implications for users of smartphones. A comprehensive survey on several privacy issues related to location-aware smartphones is presented in [86].

Some studies for preventing this class of attacks are proposed in [87, 88]. An approach for designing anti-theft solutions for smartphones is described in [89].

Examples: The target of the theft performed by ZeuS MitMo [25] is information about online banking account. Exploiting some social engineering techniques and the fact that many companies use SMS as a second authentication factor, ZeuS MitMo infects the smartphones and performs online banking operations in place of the legal owner of the account. To accomplish its tasks, first of all, an SMS with a link to the malicious mobile application is sent to an owner of an online banking account. If the user installs the application,

the attacker gains control of the infected smartphone and can read all the SMS messages that are being delivered. If the user logs into the online banking account, she is asked to enter her mobile number. This number is used by the company to improve verification system: in fact, the customer receives on her number a text message with the transaction details and a code to enter back into the website. Then, the following steps performed by this attack are, in order:

1) the attacker logs in with the stolen credentials using the user's device and performs a specific banking operation that needs SMS authentication;
2) an SMS with the authentication code is sent to the user's mobile device from the verification system. The malicious software running in the device forwards the SMS to a system controlled by the attacker;
3) the attacker fills in the authentication code and completes the operation.

[90] discusses a *drive-by download attack*, i.e. when a user inadvertently downloads a malware (usually a spyware) on visiting a website, against an iPhone 3GS that enables an attacker to steal the SMS database from the phone.

2) Sniffing: Sniffing attacks on smartphones are based upon the use of sensors, e.g. microphone, camera, GPS receiver. These sensors enable a variety of new applications but they can also seriously compromise users' privacy. If a smartphone is compromised, an attacker can access the data stored in the device and also use the sensors to sniff and record all of the user's actions.

A defense system against sniffing attacks is proposed in [91].

Examples: [92] describes the design and implementation of *Stealthy Video Capturer* (SVC), a spyware that can secretly activate the built-in camera on smartphones to compromise the users' privacy by recording private video information, with little power consumption and is stealth to commercial antivirus.

Soundminer [93] is a proof-of-concept Trojan targeting Android devices that is able to extract private data from the audio sensor. Sensitive data, such as credit card number or PIN number, can be sent out by analyzing both tone and speech-based interaction with the phone menu system. As an example, Soundminer can infer the destination phone number by analyzing audio and reporting this data remotely to a malicious party. Few permissions are requested by the Trojan during installation, specifically that it is granted access to the microphone. Other permissions, in particular network connection or intercepting phone calls, are not requested. For this reason, since Soundminer cannot directly access the Internet, transmissions need to be carried out through a second application, either a legitimate network application or through a program with the networking permission. This allows Soundminer to circumvent mechanisms aimed at mediating explicit communications between two untrusted applications, such as those proposed in [94, 95].

3) Denial-of-Service: With a Denial-of-Service (DoS), an attacker denies availability of a service or a device. DoS attacks against smartphones are mostly due to strong connectivity and reduced capabilities: due to the limited hardware, attacking a smartphone can be accomplished with a small effort; even the traffic generated by only one attacker may be enough to make a device unusable. Specific DoS could quickly drain the batteries, shutdown or dramatically limit the operation time and perform CPU intensive tasks that require a lot of energy or force the device to shut-down. Another type of DoS sends a huge amount of SMS/MMS to the same phone number to either deny users to perform their tasks or degrade the service of an area. [96] shows that by using only SMS communications, i.e. messages sent between smartphones, low-end phones can be forced to shut down. To this purpose, the SMS protocol can be used to transmit small programs that run on a smartphone. Network operators use these files to change the settings on a device remotely. The same approach has been exploited to attack smartphones.

Some studies for preventing this class of attacks are proposed in [97, 59].

Examples: In [98], the authors examine current security mechanisms on smartphones, by identifying some critical vulnerabilities of existing security models. Moreover, they show how such vulnerabilities can be exploited to launch Distributed DoS (DDoS) attacks to public service infrastructures by diverting phone calls. This is achieved by injected a crafted shell code through a buffer overflow: in fact, in many Linux-based mobile phone the system single-user can easily access root privileges to invoke `ptrace()` to inject code in any other process. The authors demonstrate that, by only leveraging 1% of Linux-based mobile systems, the service of an emergency-call center, in a region with millions of population, can be disabled.

The *battery exhaustion attacks* targets a unique resource bottleneck in smartphones, namely the battery power. [99] discusses attacks against smartphones that drains the device's battery power up to 22 times faster, thus rendering a device useless in short time. To do this, first an attacker has to build a "hit-list" of all the users with an active Internet connection by taking advantage of the insecure MMS protocol, which automatically downloads MMS messages upon receiving notification through HTTP requests. Secondly, the attacker has to periodically send UDP packets to the target smartphone and exploit PDP context retention and the paging channel. The authors show that if a phone is connected to the Internet continuously, its battery life would be completely drained in less than 7 hours.

The *water torture attack* [100] is another example of battery exhaustion attack that is carried out at the PHY layer. This is achieved by forcing the subscriber station (SS) to drain its battery, or consume computing resources, by sending bogus frames.

An example of attack that targets several Symbian S60 smartphones and that prohibits victims from receiving SMS messages is called *Curse of Silence* [101]. This attack tries to set the Messages Protocol Identifier to "Internet Electronic Mail" so that an SMS can be used to send e-mails. This attack exploits a vulnerability with some smartphones that cannot handle correctly e-mail address with more than 32 characters: by exploiting this vulnerability, the attacker sends a crafted e-mail in such a way that the device is not able to receive any other SMS message. At the end of the attack, the smartphone displays a warning reporting that the memory is not enough to

receive further messages and that some data should be deleted first.

[102] presents a novel method for vulnerability analysis of SMS-implementation by injecting short messages locally into the smartphone and analyzing and testing all the SMS-based services implemented in the smartphone software stack. The vulnerability analysis is conducted by fuzzing various fields in a standard SMS message including elements such as the sender address, user data and various flags, or by fuzzing the UDH header. Another test concatenates various SMS messages in such a way to force messages to arrive out of order or sends large payloads. The authors state that through the use of the testing tools, they were able to identify several vulnerabilities that can be exploited to launch DoS attacks.

4) Overbilling: The overbilling attacks charge additional fees to the victim's account and may transfer these extra fees from the victims to the attackers. Since many wireless services are regulated by pay-per-use contracts, these attacks are very specific to wireless smartphones.

Example: A characteristic of GPRS networks is the "always on" mode: users are billed by the amount of traffic instead of the usage time. A typical example of overbilling attack in this network is the one where an attacker, in cooperation with a malicious server located outside of the GPRS network, hijacks the IP address of the target device and starts a download session from the malicious server. Hence, the legitimate user gets charged for traffic that never requests [110].

V. Security Solutions for Mobile Devices

In this section we survey existing mechanisms that are developed to prevent different type of threats for smartphones. We present, first of all, intrusion detection systems for smartphones, then trusted mobile-based solutions. All the solutions are presented in chronological order.

Table V includes some conventional approaches typically implemented by off-the-shelf smartphone applications to provide basic security; instead, table VII lists, in chronological order, the research security solutions (described in the following sections) that provides a prototype. These solutions are classified according to their detection principles, architecture (distributed or local), reaction (active or passive), collected data (OS event, keystrokes), and OS.

A. Intrusion Detection Systems

In this section, we present the state of the art of models and tools that implement Intrusion Detection Systems (IDSes) on smartphones.

IDSes can be based upon two complementary approaches:

1) *prevention-based* approaches: using cryptographic algorithms, digital signatures, hash functions, important properties such as confidentiality, authentication or integrity can be assured; in this scenario, IDSes have to be running online and in real-time;
2) *detection-based* approaches: IDSes serve as a first line of defense by effectively identifying malicious activities.

Furthermore, there are two main types of detection:

1) *anomaly-based* (alternative names: anomaly detection, behavior-based), which compares the "normal" behavior with the "real" one;
2) *signature-based* (alternative names: signature detection, misuse-based, knowledge based, detection by appearance), based upon patterns of well-known attacks.

There exist also hybrid approaches which combine the aforementioned types of detection. With signature-based approaches, the advantage is the false alarm rate that is usually very low. The disadvantage is that they can detect only known attacks. On the other hand, with an anomaly-based IDS we can detect variations of known attacks and even new attacks, but the amount of false alarms is usually quite high. Some of the metrics used to measure their effectiveness are true positive rate, accuracy and response time. [111] provides a general introduction to IDSes in cellular mobile networks.

In the following, we partition existing IDS solutions using these features:

- *detection principles*:
  - anomaly detection:
    - machine learning;
    - power consumption.
  - signature-based:
    - automatically-defined;
    - manually.
- *architecture*:
  - distributed;
  - local.
- *reaction*:
  - active;
  - passive.
- *collected data*:
  - system calls;
  - CPU, RAM;
  - keystrokes;
  - SMS, MMS.
- *OS*:
  - Symbian;
  - Android;
  - Windows Mobile;
  - Apple iOS.

First of all, we cluster mobile IDSes based upon the detection principles used to find anomalies: anomaly detection (which includes machine learning and power consumption), signature-based (automatically or manually defined) and run-time policy enforcement. Then, we consider both local and distributed architectures. Next, we distinguish tools that perform any kind of reaction from those that only detect anomalies. We further classify IDSes by considering what kind of data are used as input and by the OS. For each feature, all the solutions discussed are presented in chronological order.

1) Detection Principles: We partition existing IDSes using the following detection principles:

- anomaly detection;
- signature-based;
- run-time policy enforcement.

TABLE V
SECURITY APPLICATIONS FOR SMARTPHONES

Product	Features	OS	License	Reference
WaveSecure	Lock and Wipe Backup and Restore Localization and SIM Tracking	Android BlackBerry Symbian Windows Mobile J2ME	Commercial	[103]
Norton Mobile Security Lite	Theft of Private Stuff Unauthorized Access Mobile Viruses Malware and Threats Harmful Downloads	Android	Commercial	[104]
iCareMobile	Parental Control Automatic Pornographic Content Detection	Symbian Android	Free	[105]
BullGuard Mobile Security 10	Antivirus and Anti-spyware Anti-theft Parental Control Firewall Spam-filter Basic Backup	Android BlackBerry Symbian Windows Mobile	Commercial	[106]
Kaspersky Mobile Security 9	Privacy Protection Anti-theft Parental Control Encryption Anti-Spam Anti-Malware Firewall	Android BlackBerry Symbian Windows Mobile	Commercial	[107]
ESET Mobile Security	Antivirus Firewall SMS/MMS Anti-spam Anti-theft	Symbian Windows Mobile	Commercial	[108]
Lookout Mobile Security	Lock Wipe Backup Wipe Privacy of Data	Android iPhone	Free	[109]

a) Anomaly Detection: An *anomaly detection* system compares the "expected" behavior of the smartphone with the "real" behavior (the actions executed at run-time by the device). The solutions included in this section either monitor distinct activities on the mobile, e.g. SMS or MMS services or Bluetooth connections, or analyze the power consumption model of the phone to detect anomalies. Moreover, we detail frameworks that adopt run-time monitoring of the activities.

As discussed in [112], we can split the architecture of a generic smartphone in the following layers:

- user;
- application;
- virtual machine or guest OS;
- hypervisor;
- physical.

For each functional layer, the authors propose several distinct features that should be collected for measuring the phone's behavior and used by an anomaly detection IDS. Table VI lists some of the capabilities that can be measured at each distinct layer.

Anomaly-based approaches for smartphones are either based upon machine learning techniques or upon monitoring power consumption.

Machine Learning: The paper [57], from the same authors of [59], proposes *Proactive Group Behavior Containment* (PGBC), a framework aimed at containing malicious software spreading in messaging networks such as IM and SMS/MMS. The primary components of PGBC are service-behavior graphs generated from client messaging patterns and behavior clusters that partition the service-behavior graph into clusters of similar behavior. The authors present an algorithm that, in the presence of a malicious attack, automatically identifies the most vulnerable clients based upon interactions among them. Moreover, they describe a proactive containment framework that applies two commonly-used mechanisms (namely, rate-limiting and quarantine) to the dynamically-generated list of vulnerable clients in a messaging network whenever a worm or virus attack is suspected.

Ho and Heng [46] attempt to identify generic behavioral patterns in mobile malware to build a generic defense model. To slow down the spread of mobile malware, the authors introduce an extended model (which incorporates the works of [47] and [48]), which is a Java-based engine that is independent from the platform. In addition to previous solutions, this model includes a feature for blocking the silent automated transmission attempts of virus installation files from a compromised mobile smartphone via MMS, Bluetooth, Infrared, e-mail and Instant Messaging.

Schmidt et al. [48] exploit the monitoring process of smartphones to extract features that can be used in a machine learning algorithm to detect anomalies. The framework includes a monitoring client, a *Remote Anomaly Detection System* (RADS) and a visualization component. The monitoring client, which represents the main goal of this paper, is a client that runs on a smartphone and includes three main components:

- user interface;
- communication module, which manages connection with RADS;
- Feature Extractor, which implements measurements and observation of resources and other components.

TABLE VI
ANOMALY DETECTION CAPABILITIES AT DISTINCT LAYER

Layer	Features
Hypervisor	address spaces data types, data constructors and data fields system parameters virtual registers system calls communication protocols
Application	application manager framework security framework messaging framework multimedia framework
User	SIM/phone password key-stroke/T9 usage/spelling analysis top-n called/texted numbers/contacts based on hour, day, week, month frequently executed smartphone applications smartphone application usage analysis Bluetooth/Wi-Fi usage analysis

The RADS is a web service that receives, from the monitoring client, the monitored features and exploits this information, stored in a database, to implement a machine learning algorithm.

Some other solutions use a probabilistic approach to trace behavior's profiles on smartphones. For example, [49] devises a behavior-based malware detection system (*pBMDS*) that adopts a probabilistic approach through the correlation of user's inputs with system calls to detect anomalous activities. pBMDS observes unique behaviors of the applications and the user's input and leverages hidden Markov model to learn application and users' behavior through process state transitions and user behavioral patterns. This statistical approach is used to learn the behavioral difference between applications initiated by user and applications initiated by malware. The most distinguishing feature of the proposed solution is that its malware detection capability focuses on recognizing non-human behavior instead of relying on known attack signatures to identify malware. Therefore, in the training process, pBMDS does not require the number of negative samples to be equivalent to that of the positive ones. The system exploits behavior graphs, which reflect intermediate process states towards each key system call based on user operational patterns, e.g. keystrokes.

In [113], time-stamped security data are continuously monitored within the target smartphone and then processed by the Knowledge-Based Temporal Abstraction (KBTA) methodology. Using KBTA, continuously measured data (e.g., the number of sent SMSs) and events (e.g., software installation) are integrated with a smartphone security domain knowledge-base, which is an ontology for abstracting meaningful patterns from raw and time-oriented security data. These patterns are used to create higher level, time-oriented concepts and patterns, called *temporal abstractions*. Automatically-generated temporal abstractions are then monitored to detect suspicious temporal patterns and to issue an alert. These patterns are compatible with a set of predefined classes of malware as defined by a security expert employing a set of time and value constraints.

Damopoulos et al. [114] use a large dataset of iPhone users' data log with four machine learning algorithms, namely Bayesian Networks, Radial Basis Function, K-Nearest Neighbors and Random Forest, to detect illegal use of a smartphone. They classify the behavior of users through telephone calls, SMSs and, differently from earlier research, on Web browsing history. To preserve users' anonymity, each record is hashed through SHA-1. Then, the data are examined either independently or in combination in a Multimodal fashion: in the first case, Random Forest was the most promising classifier with a true positive rate above 99.8% and accuracy of 98.9%; in the second case, the best results were given by K-Nearest Neighbors with 99.8% true positive rate and 99.5% accuracy.

Andromaly [115] is a general and modular framework for detecting malware on Android smartphones using a supervised anomaly detection technique. The framework is based upon a host-based IDS that sample numerous system metrics, such as CPU consumption, number of sent packets, number of running processes. The framework is composed of four main components, namely the feature extractors, processors, main service and the graphical user interface. The classifiers used to evaluate the framework are k-means, logistic regression, histograms, decision tree, Bayesian networks and Naïve Bayes; furthermore, a filter approach has been used for feature selection. A total of 88 features were collected for each monitored application. Experiment results show that Naïve Bayes and logistic regression were superior over the other classifiers in most of the testbed configurations and fisher score was the best setting for feature selection.

[116] presents a framework to dynamically analyze applications behavior to detect malware on Android by collecting system traces from several real users through crowdsourcing and a central server. An application client, *Crowdroid*, monitors Linux system calls and, after preprocessing, sends them to a central server, which then parses data and creates a system call vector. Finally, each dataset is clustered through a partitional clustering algorithm, namely 2-means.

Power Consumption: An example of monitoring power consumption is presented in [117], where the authors propose a battery-based IDS. This solution performs the sensing of abnormal battery behavior and energy patterns to detect a variety of attacks and includes two modules:

- Host Intrusion Detection Engine (HIDE), which measures energy consumed over a period of time (established according to an algorithm);

- Host Analyzed Signature Trace Engine (HASTE), which matches frequency patterns of attacks based upon power signature and compares these signatures with a short list of know attack signatures.

Similarly to the previous solution, Kim et al. [97] propose a power-aware malware detection framework that monitors, detects, and analyzes energy-greedy malware. The framework is composed of a power monitor, to collect power samples and to build a power consumption history from the collected samples, and of a data analyzer, to generate a power signature from the constructed history.

Another example of monitoring power consumption is proposed in [118], which bases its observations on the fact that any malware activity on a smartphone consumes battery power. Hence, the proposed solution (*VirusMeter*) performs malware detection using state machine and power consumption model by comparing the actual measured power consumption with the power that could have been consumed, according to the predefined power consumption model. The user-centric power model characterizes power consumed as a function of common user operations and relevant environmental factors (e.g., calls, SMS, MMS). To collect data, for each user operation a state machine is constructed. This state machine describes the evolution of internal events related to user's operations.

Even if power consumption is often considered as a limitation, Yan et al. [119] consider its positive impact on preventing and suppressing mobile malware. Even if no specific solutions are implemented, three potential techniques, based upon limitations of smartphone, are proposed:

- *monitoring power consumption*, which concerns the ability to detect an attack depending on battery usage on smartphone;
- *enforcing hardware sandbox*, in which all the hardware modules that are not used frequently and that can be used for malware propagation (e.g. GPS or Bluetooth) are switched off;
- *increasing platform diversity*, in which different APIs are used to develop and execute an application.

b) Signature-Based: This paragraph discusses mechanisms that detect anomaly on smartphones using signatures. The signature-based approach checks if each signature derived from an application matches any signature in a malware database. The database of malware signature can be automatically or manually defined.

Automatically-Defined: Venugopal et al. [120] apply Bayesian decision theory to the dynamic-link library (DLL) usage of a program to detect viruses. In fact, since most mobile viruses have common functionalities (e.g., deleting system files, sending MMS), these programs need to use DLLs. By exploiting the common patterns of DLL usage among viruses, the proposed approach can detect old and new viruses: a classifier takes as input a binary vector specifying the occurrence (or not) of each DLL function in the feature set.

In [121, 122] the authors propose to adopt a twofold approach to block the spreading of worms on smartphones:

- at the *terminal level*, where a graphic Turing test and identity-based signatures block unauthorized messages from leaving compromised phones;
- at the *network level*, where a push-based automated patching scheme clean compromised phone. This means that network providers automatically push software patches to compromised terminals.

In [123] the signatures to be compared are automatically generated by analyzing the device's activities. The authors discuss the dynamics of mobile malware that propagate by proximity contact and explore three strategies to detect and mitigate proximity malware, namely:

- *local detection*, in which devices detect when they become infected and disable further propagation;
- *proximity signature dissemination*, in which devices create content-based signatures of malware and disseminate them via proximity communication as well;
- *broadcast signature dissemination*, in which a centralized server aggregates observations from individual devices, detects propagating malware, and broadcasts signatures to smartphones.

Bauckhage et al. [124] present a probabilistic diffusion scheme for detection anomalies indicating malware, which is based upon the device's usage patterns. The basic idea is to model dependencies of samples and features by means of a bipartite graph, which then serves as the domain of a Markov process. The algorithm is applied to two separate data sets obtained from smartphones during normal daily usage.

Manually-Defined: These mechanisms to detect anomaly extract the signatures of mobile malware by manually analyzing the malware and its behavior.

Ellis et al. [125] present a new approach for the automatic detection of worms on smartphones using behavioral signatures, which are manually defined to represent common features across a family of worms. The presented approach focuses on detecting patterns at a higher level of abstraction, where a pattern may be:

- sending similar data between devices;
- tree-like propagation and reconnaissance;
- changing a server device into a client.

Ideally, a pattern is a specific behavior of a spreading worm and should be distinct from normal network traffic. The frequency of and interrelationships between behaviors improve the accuracy of the detection. To evade a behavioral signature requires a fundamental change in the behavior and this task is rather challenging.

In [47], the authors present a behavioral detection framework for viruses, worms and Trojans that extracts key behavior signatures of mobile malware by applying *Temporal Logic of Casual Knowledge* (TLCK) on a set of atomic steps. The authors have generated a database of malware signatures based upon a review of mobile malware: every signature corresponds to the description of the behavior of a family of malware. The run-time monitoring is implemented on the Symbian emulator through a proxy DLL to monitor API calls. To distinguish malicious behavior from partial signatures, the framework exploits support vector machines to train a classifier from normal and malicious data.

c) Run-Time Policy Enforcement: The basic idea of these models is that mobile code consumers essentially accept the code "as-is" and exploit a supporting mechanism to enforce the policy associated with the code to detect and stop anomalies.

Nowadays, several smartphones are able to run Java applications, which can also create Internet connections, send SMS messages, and perform other expensive or dangerous operations on the smartphone. Hence, an adequate security support is required to meet the needs of this scenario. To this end, [126] proposes an approach to enhance the security support of Java Micro Edition (J2ME), based upon the monitoring of the usage of the smartphone's resources performed by MIDlets. A process algebra-based language defines the security policy whereas a reference monitor is exploited to check the resource usage.

Security-by-contract [127, 128] is a run-time policy enforcement solution based upon a digital signature that:

- certifies the origin of the code;
- binds the code with a contract.

A *contract* contains a description of the relevant features of an application and the relevant interactions with its host platform. A mobile platform can specify platform contractual requirements (a *policy*), which should be matched by the application's contract. The authors also propose some algorithms to verify contract-policy matching.

[129] enhances the security-by-contract architecture by adding new modules and configurations for managing contracts. At deploy-time, the proposed system selects the run-time configuration depending upon the credentials of the contract provider; at run-time, the system can both enforce a security policy and monitor the declared contract. According to the actual behavior of the running programs, the architecture can update the trust level associated with the contract provider. The main advantage of the proposed architecture is the automatic management of the level of trust of software and contract releasers.

Finally, [94, 95] propose *Kirin* security service for Android, which performs lightweight certification of applications to mitigate malware at install time. Kirin certification uses security rules that match undesirable properties in security configuration bundled with applications.

2) Architecture: Local or Distributed: In this section, we partition mechanisms for intrusion detection that use a local architecture from solutions that exploit a distributed architecture.

In a *local architecture*, both the collecting phase and the analysis phase are locally performed on the device and no interactions with an external server is required. Examples of local solutions are presented, for instance, in [130, 118, 94, 95]. On the other hand, a *distributed architecture* usually requires a distinct and separated component (i.e., a server) to analyze the activities collected and sent by each device. In this architecture, the external security component can perform all the analysis on the activities monitored on the smartphones without having problems of:

- power consumption;
- small screen size;
- limited resource.

[39] discusses a framework to detect attacks against smartphones using a separate, loosely-synchronized, security server, which hosts one or more exact replicas of the smartphone and applies distinct detection mechanisms. In this way, several expensive detection techniques, which cannot be easily implemented on the smartphone, can be applied on the server to prevent attacks on the phone software. The architecture includes a *tracer* on the phone itself, to intercept both system calls and signals of a set of protected processes, whereas a *replayer* on the security server later replays the execution trace and looks for anomalies.

3) Reaction: Passive or Active: In this section, we consider whether existing mechanisms for intrusion detection react or not whenever a new threat is found, e.g. by trying to prevent the attacks to damage the smartphone.

A *reaction* can be a strategy, to contain the virus or malware propagation, or a mechanism, like alerting the user of the infection. An example of a reaction strategy is presented in [57], where the alerts about potential attacks are collected before starting the reaction strategy. The proposed framework is implemented at the messaging service center where logs of client communication are kept. These logs can be analyzed to generate a *service-behavior graph* for the messaging network: this graph is then further processed to generate *behavior clusters*, i.e., groups of clients whose behavior patterns are similar with respect to a set of metrics, namely:

- interaction frequency;
- attachment and message size distributions;
- number of messages;
- number of outgoing connections to other clients;
- list of traced contacts.

When the number of alerts in a particular behavior cluster reaches a threshold, the messages belonging to that cluster are first rate-limited to slow down a potential malware. When the alerts reach a second threshold, the containment algorithm applies proactive quarantine, i.e., it blocks messages from suspicious clients of these behavior clusters. This step essentially enables the behavior clusters to enter into a group defense mode against the spreading malware.

In [130], the proposed system detects viruses running on smartphones using a proxy that performs the analysis of the phone's behavior and, when a potential virus is detected, sends targeted alerts to both infected devices and a subset of the uninfected devices to prevent the spreading. These devices are chosen based upon the users' contact list and mobility profiles, to locate those devices they may be in direct contact with an infected device.

A further example of a reaction-strategy is given in [58] where, to limit the spread of MMS- and SMS-based worms, the authors propose a methodology based upon a graph partitioning approach. The problem is tackled by using a social relationship graph where the devices are divided into multiple partitions based upon the social relationship among them. Each partition includes smartphones that closely interact with each other. Communication patterns are extracted using a network trace and, from these patterns, a social relationship graph of smartphones is built so that an optimal set of phones to be

patched can be located firstly. In this way, the system can locate the phones that have the capability to infect the highest number of other phones. The authors propose two patching schemes, namely:

- *balanced partitioning*, where the significance level of each partition are chosen to be as similar as possible so that the worm damage to each partition can be balanced;
- *clustered partitioning*, where edges within each partition have higher weights compared to the edges between the two partitions, so that smartphones that are socially close to each other are in the same partition and nodes that are not close are into different partitions.

Ruitenbeek et al. [131] study the propagation of smartphone viruses based upon MMS and propose several response mechanisms to quantify the effectiveness of virus mitigation techniques. The authors present four MMS virus scenarios: in every scenario the virus on the phone sends MMS messages with an infected attachment file to other phones, which are selected from the contact list of the infected phone or by dialing a random phone number. After receiving this new MMS message, if the user accepts the infected attachment file, the virus is installed, the target phone becomes infected and under control of the attacker. The evaluated response mechanisms for each of the four scenarios are:

- scan of all MMS attachments in MMS gateways to detect viruses;
- user education;
- immunization using software patches;
- monitoring for anomalous behavior;
- blacklist of phones that are suspected of infection.

The experimental results revealed that any response mechanism must be agile enough to quickly react to rapidly propagating viruses and discriminating enough to detect stealthier, slowly propagating, viruses.

4) Collected Data: As suggested in [132], monitoring data in a mobile environment can be a challenge due to administrative, technical and conceptual limitations. The visibility of the data can be limited by explicit agreements on exchange of monitored data because, for example, the call records can be under administrative control. Furthermore, we have to consider that due to the integration of several communication interfaces, a smartphone can be connected, at the same time, to different access points (such as Bluetooth, Wi-Fi) and thus the amount of collected data on the communication interfaces can be huge. Finally, we have to take into account that there are some kinds of attacks, e.g. Trojans, not involved in communication activities, so that they are invisible to the network.

Due to the fact that all the solutions based upon intrusion detection need to access several features of a smartphone, we should carefully consider the problem of privacy of the data accessed. By accessing a smartphone, several pieces of information can be retrieved, such as the user's location, communications, and personal contacts. All these kinds of information are, obviously, private and should not be shared to third parties: hence, any tool for intrusion detection should be run only with the user's explicit authorization in case it may access private data. Private data should never be carried out of the smartphone by any mechanisms, including system that have to protect the mobile. Therefore, an intrusion detection system should be designed so that:

- the private data used during the intrusion detection process are not transferred out of the device;
- the communication of the intrusion detection functionalities on the smartphone should be restricted to the generated intrusion alarms.

In [91] the authors focus on threats and attacks that violate user's privacy by sniffing on the sensors on smartphones. The authors develop a threat model based upon the use of sensors and design a general framework for a defense system. This framework consists of three modules: (i) policy engine, (ii) interceptor, (iii) user interaction. The policy engine, based upon the input from user interaction and application monitoring/profiling, determines access: these decisions are based mainly upon application monitoring and profiling without requiring much user intervention. Several policies are considered, such as white-listing, blacklisting and information-flow tracking. The interceptor is interposed between the application and the sensors, and/or between the application and the network, and it enforces the decision of the policy engine. The user interaction is not a mandatory component, since it simply notifies the user by asking her decision. For each module, different mechanisms are explored and discussed but no real implementations are presented.

In the following paragraphs, we discuss solutions that analyze different kinds of data available in a smartphone, namely:

- OS events (e.g., system calls);
- measurements (e.g., CPU, RAM, I/O activities);
- keystrokes;
- communication events (e.g., SMS/MMS).

a) Operating System Events: These events indicate those activities related to the normal functioning of the OS, which can be used to retrieve relevant information about the behavior of the smartphone, and include:

- system calls;
- function calls;
- network operations.

Data from these events can be obtained by exploiting some built-in OS mechanisms. However, in many cases the OS events cannot be monitored because there is no direct access interface. Therefore, in these cases some extensions to the OS are required.

The solution presented in [133] monitors the system calls executed by running processes and labels executing code based upon its access to the network interfaces (e.g., wireless, GSM, Bluetooth). The labels are then transferred between processes and system resources as a consequence of either access or execution. During sensitive operations, the labels collected in this way are compared to a set of rules that allow users to specify fine-grained access control to services and data. The labeling process, which can involve processes and resources, as well as the enforcement of the policies, are performed by a kernel-level reference monitor. The framework is independent of the OS because it uses a policy language that allows users to express what actions are allowed by specific classes of programs with respect to specific classes of resources.

Another mechanism, *MobileSandbox* [50], monitors the system calls issued by the software that a user is going to install on its smartphone. MobileSandbox is a background system that samples and translates an operation in a sequence of API calls that the program has issued during its execution. The proposed system can either use the device emulator or an actual device. To perform mobile dynamic malware analysis, the proposed solution includes three steps, namely:

- collecting the software samples as complete as possible;
- analyzing the samples as complete as possible;
- taking certain actions as a response to the analysis.

In [134], the authors discuss in details the implementation of system call interception for Windows CE. Two solutions that analyze these events are proposed: the first one exploits a sandbox approach to analyze malware, whereas the second solution implements the concept of a reference monitor at the OS level.

There are also some solutions to detect anomalies that monitor *function calls* rather than system calls. One of these solutions is presented in [135] where the authors apply static analysis of function calls to detect malicious applications. *Centroid Machine* is the name of the light-weight algorithm developed according to common clustering methods. The algorithm can detect Symbian OS malware on the basis of function calls according to the requirements of smartphones, e.g. efficiency, speed and limited resource usage.

A similar solution specific to Android platform is presented by Schmidt et al. [136]: the discussed framework performs static analysis on the executables to extract their functions calls using `readelf` command. This command returns detailed information on relocation and symbol tables of each Executable and Linking Format (ELF) object file. The output of this analysis is the static list of referenced function calls for each system command. Then, these calls are compared with malware executables for classification.

b) Measurements: A set of measurements includes several performance indicators of a smartphone, such as CPU activity, memory consumption, file I/O activity and network I/O activity. The key idea is that, supposing that changes in the usage of a smartphone are gradual, the normal usage remains constant over time. Therefore, we can extract behavioral profiles and use them for comparison with normal behaviors in order to detect anomalies. Some of these features (e.g., RAM free, user inactivity time, process count, CPU usage, SMS sent count), which are used for anomaly detection, are discussed in [48, 115].

c) Keystrokes: Some solutions exploit *keystroke logging* (keylogging) techniques to detect anomalies. These techniques track the keys struck on a keyboard to monitor the actions of the user. Typically, the logging is provided in a covert manner so that the user is unaware of the monitoring. The conventional technique of behavior-based anomaly detection focuses on the rhythm of keystroke patterns or transition probability of commands.

In [137], keylogging is applied to smartphones to detect illegal user operations using a scheme that uses a background process to records keystrokes. Keystrokes are divided into long-term and short-time: using the frequency of long-term keystrokes, an anomaly detection algorithm constructs a user profile; then, the short-term keystrokes are compared with the user profile to detect illegal users.

[138] demonstrates that keystroke dynamics of a user can be translated into a set of features for accurately identifying users. To this end, keystroke data of twenty-five smartphone users are collected and analyzed: based upon them, six distinguishing keystroke features are extracted and used for user identification. The results show that the proposed user identification system has an average error rate of 2% after the detection mode and the error rate of rejecting legitimate users drops to zero in the PIN verification mode.

d) Communication Events: Communication events indicate a particular class of events that happen in a device at the application level, such as high-level actions (e.g. sending/receiving of SMS, files). Typically these actions are composed of several elementary actions that cannot be automatically generated by the smartphone's OS. Communication events include operations such as sending and receiving of messages, or file downloads/uploads.

[139] discusses threats on smartphones mirrored from PCs and proposes a detector application that monitors SMS sent without user authorization. Bose and Shin [59] investigate the propagation of mobile worms and viruses that spread primarily via SMS/MMS messages and short-range radio interfaces (e.g. Bluetooth). Each smartphone is modeled as an autonomous mobile agent capable of sending SMS messages and of discovering other devices equipped with Bluetooth. To identify a set of common behavior vectors, and to develop mobile virus detection and containment algorithms, the existing mobile viruses are investigated. The authors study the vulnerabilities of Bluetooth and SMS/MMS messaging systems in depth and identify the vulnerabilities that may be exploited by future mobile viruses. Finally, they develop the state diagram of a generic mobile virus that can spread via SMS/MMS and Bluetooth. The discovery, infection and replication states of the generic virus are implemented in an agent-based malware modeling framework to study its propagation and containment strategies.

[140] presents a novel approach to the security testing of MMS user agents by taking into account the effects of the infrastructure on the delivery of MMS messages and then by using a virtual infrastructure to speed up the testing phase. As in [102], the paper exploits fuzzing to deliver fuzzed MMS messages to the user agents, by finding several string-length buffer overflows. *SmartSiren* [130] is a collaborative virus detection and alert system for smartphones that collects communication activity information and performs analysis to detect abnormal behaviors. To halt potential virus outbreaks the authors try to minimize the number of smartphones that can be infected by a new released virus. A light-weight agent runs on each smartphone, while a centralized proxy assists the virus detection and alert process. Each agent keeps track of the communication activities on the device and periodically reports a summary of these activities to the proxy. The proxy performs joint analysis on the reports and detects any single-device or system-wide viral behaviors.

Finally, [141] proposes a lightweight scheme that can detect anomalous SMS behaviors with high accuracy. The authors start analyzing an SMS trace collected within a five-month

period and, according to the analyzed results, four detection schemes are proposed. Each scheme builds normal social behavior profiles for each SMS user and then uses them to detect SMS anomalies in an on-line and streaming fashion. Since a scheme stores only a few states in memory for each SMS user, it imposes very low overhead during on-line anomaly detection. Finally, the authors evaluate these four schemes and also two hybrid approaches with realistic SMS traces by showing that the proposed approach can detect more than 92% of SMS-based attacks with a false alarm rate of 8.5% and 66% of attacks without generating any false alarm.

5) Operating Systems: In this section, existing IDSes are clustered based upon the OS for which they are developed or studied, namely:

- Symbian;
- Android;
- Windows Mobile;
- iPhone OS.

a) Symbian: Symbian is the Nokia's open source OS and software platform designed for smartphones. The latest Symbian platform includes a user interface component based on S60 5th Edition. Originally developed by Symbian Ltd, Symbian OS is now at the third version, Symbian^3, released in the fourth semester of 2010. Devices based on Symbian accounted for 43.5% of worldwide smartphone sales in the first two semesters of 2010 [142].

Schmidt et al. [48] introduces an approach to monitor Symbian-based smartphones to extract features that can be used by a machine learning algorithm to detect anomalies. In [143], the authors test vulnerabilities on smartphones based upon the Symbian 9.1 OS. Several attacks have been experimented to test the stability of the network stack of the Symbian OS. Some vulnerability has been found that can render the devices unusable.

Further discussions can be found in [144, 145].

b) Android: Android is a mobile OS based upon a modified version of the Linux kernel. The applications are written by a large community of developers to extend the functionality of the devices.

[146] and [147] present a set of results on the evaluation of the security of Android smartphones. Firstly, the authors analyze the Android framework and the Linux Kernel to check security functions, by also surveying some known security tools and mechanisms to increase the smartphones security. Then, the authors analyze the possibilities of applying malware detection mechanisms at the kernel-level, i.e. by monitoring key-kernel events (log file, file system activities). Finally, they apply static function call analysis to detect malware on ELF executables, by exploiting a decision tree for deciding if a new application is suspicious compared to previously analyzed applications (both good and bad ones). [148] presents a collaborative architecture to detect anomalies on Android platforms. [149] presents a review of current security solutions for Android platforms.

In [150], the authors provide a security assessment of the Android framework: firstly, they discuss the current security mechanisms incorporated in Android (namely, Linux mechanisms, environmental features and Android-specific mechanisms); secondly, they propose some security solutions for mitigating threats on Android, using five "threat cluster":

1) threats that compromise availability, confidentiality, or integrity by maliciously using the permissions granted to an installed application;
2) threats that compromise availability, confidentiality, or integrity threats that happen when an application exploits a vulnerability in the Linux kernel or system libraries;
3) threats that compromise the availability, confidentiality, or integrity of private or confidential content: e.g., applications that can read the SD card's contents or attackers eavesdropping on wireless communication remotely;
4) attackers draining a smartphone's resources: e.g., since applications for Android have neither disk storage nor memory quotas hogging memory or CPU is possible;
5) threats that compromise internal or protected networks: as an example, attackers can use Android devices to compromise other devices, computers, or networks by running network or port scanners, SMS/MMS/e-mail worms, and various other attacks.

[151] proposes a security solution for Android-based smartphones that exploits Security-Enhanced Linux.

TaintDroid [87] is an extension to Android that tracks the flow of sensitive data through third-party applications. TaintDroid assumes that downloaded third-party applications are not trusted, and monitors how these applications access and manipulate users' personal data. TaintDroid automatically labels data from sensitive sources and transitively applies labels as sensitive data propagates through program variables, files, and interprocess messages. When tainted data are transmitted over the network, TaintDroid logs the data's labels, the application responsible for transmitting the data, and the data's destination. The tested performance overhead is 14%.

Enck et al. [152] study 1,100 popular free Android applications using a decompiler to recover Java application source code from its Dalvik installation image and by statically analyzing more than 20 million lines of code. The study shows that several applications misuse privacy sensitive information, in particular phone identifiers (IMES, IMSI and ICC-ID) and geographic location, such as leaking phone identifiers through plaintext requests (such as HTTP GET/POST), tracking users through their phone identifier.

c) Windows Mobile: Windows Mobile is an OS developed by Microsoft for use in smartphones. Based upon the Windows CE 5.2 kernel, Windows Mobile was designed to be similar to desktop versions of Windows and is now superseded by Windows Phone 7. Third-party software development is also available and users can purchase software applications via the Windows Marketplace for Mobile.

Windows Mobile Malware Detection system (WMMD) [153] is a behavior-based malware detection system for Windows Mobile platform. WMMD uses API interception techniques to dynamic analyze application's behavior and compare it with malicious behavior characteristics library using model checking. The results show that it can effectively detect the obfuscated or packed malware variants that cannot be detected by other main stream anti-virus products.

An example of anomaly detection system developed for the Advanced RISC Machine (ARM) architecture, is discussed in [154]. The proposed system analyze the relation between system calls and their return address to describe the software behavior on ARM architectures. To this end, during a training phase, the system builds a model using an array of system calls and return addresses and produces a score. Then, at run-time the system freezes the software execution each time a system call is issued, obtains stack information and compares the software behavior with the model using an anomaly score.

d) iPhone OS: iOS (previously known as iPhone OS) is the Apple's mobile OS that was originally developed for the iPhone, but now it has been extended to support other devices, such as iPad.

To protect its users from malicious applications, Apple has introduced a vetting process, which should ensure that all applications conform to Apple's rules before they can be offered via the App Store. Unfortunately, this vetting process is not well documented, and there have been cases where malicious applications had to be removed from the App Store after user complaints. To this purpose, Egele et al. [88] study the privacy threats that applications for Apples iOS pose to users. The authors present a novel approach to automatically create comprehensive control-flow graphs from binaries compiled from Objective-C code and perform reachability analysis to identify possible leaks of sensitive information from a smartphone to third parties.

Finally, [155] discusses ten ways to keep user data on iPhone and Android-based devices safe from insecure apps.

B. Trusted Mobile

Trusted Computing Group (TCG) has published a set of specifications to measure, store, and report hardware and software integrity through a hardware *root-of-trust*, which is the Trusted Platform Module (TPM) and Core-Root-of-Trust-Measurement (CRTM). On a TPM-enabled platform, the CRTM measures the bootloader of the system before it is executed, and then stores the measured value into one of the Platform Configuration Registers (PCRs) inside the TPM. The bootloader then loads OS image, measures it, stores via PCR extension and then executes it [159]. In turn, the OS measures the loaded applications and stores their integrity values in PCRs before executing them. Upon an attestation challenge from a third party, the TPM signs a set of PCR values with an Attestation Identity Key (AIK) and sends back the result. The challenger then can make decisions on the trust status of the platform by verifying the integrity of these values and comparing with the corresponding known-good values.

There are also specifications for mobile phone platforms released by the TCG Mobile Phone Working Group, i.e. the Mobile Trusted Module (MTM) [160]. TCG advocates using MTM to increase the security of smartphones by providing basic cryptographic capabilities, such as random number generation, hashing, protected storage of sensitive data (e.g. secret keys), asymmetric encryption, as well as generation of signatures. These cryptographic primitives can be exploited to implement hardware-based security services, such as device authentication, integrity measurement, secure boot, and remote attestation. The MTM provides a root-of-trust for smartphones in the same way as the TPM does for personal computers. In principle, the MTM is an adaption of the TPM for smartphones and, hence, its specification is similar to that of the TPM, which facilitates interoperability within the existing trusted computing framework for personal computers.

There are two different types of integrity measurement for any application binary: *load-time* and *dynamic measurements*. The TCG only specifies load-time integrity measurement, when a piece of code or data is measured or when it is mapped/loaded into main memory. Dynamic measurements refer to the act of measuring the integrity of critical applications at run-time, i.e. when they are executing. In the Integrity Measurement Architecture (IMA) framework [161], measurements are invoked in several system call functions when code or kernel modules are loaded but before they are executed. After a code is mapped into memory and during run-time, it is very difficult to measure the integrity of the process considering very dynamic and nondeterministic behaviors of typical applications, such as loading active code, receiving external inputs, and allocating dynamic memory.

In addition to trusted boot and load-time integrity measurement, integrity protection for mobile phones raises the following extra requirements [158]:

- *secure boot*: a set of mandatory engines [160] reside on a single mobile platform and provide critical and indispensable services that have to be running in known-good states, i.e. their integrity must be verified to assure their trustworthiness. Therefore TCG Mobile Phone Reference Architecture [159] states that secure boot is mandatory for MTM;
- *low booting and run-time overhead*: most smartphones are still limited in computing power. This requires any security solution to be very efficient and integrity measurement during boot and in post-boot state should not degrade the performance and user experience too much;
- *run-time integrity assurance*: although run-time integrity measurement is not practical in both PC and mobile platforms, there should be some mechanism to preserve the integrity level of critical applications and resources during run-time, e.g., phone related services (telephony server) and platform management agents. Both TCG and IMA do not propose any mechanism for this purpose.

[156, 158] discuss a framework for mobile integrity measurement and attestation mechanisms, by proposing a secure boot mechanism. The proposed mechanism ensures that a mobile platform can boot into a secure state by exploiting a flow integrity model to achieve high integrity for the system. The solution leverages SELinux MAC mechanisms and adds some SELinux security policy extensions. The framework requires a root-of-trust, such as the MTM. [157] tries to protect the integrity of critical applications from potentially untrusted functionality and develop a small SELinux policy to measure the integrity of a mobile phone using the PRIMA approach [162]. The resulting SELinux policy enables the phone system to be attested to remote parties and protects critical applications from untrusted code, thus allowing users to install and run trusted applications in a safe way: the policy is 90% smaller than a custom SELinux reference policy.

TABLE VII
CLASSIFICATION OF THE IMPLEMENTED SOLUTIONS

Reference	Year	Detection Principles	Architecture	Reaction	Collected Data	Operating Systems
[125]	2004	Signatures (Manually)	Local	Passive	All	OS-Independent
[139]	2005	Anomaly Detection	Local	Passive	Communication Events	Symbian
[117]	2006	Power Consumption	Local	Passive	All	OS-Independent
[57]	2006	Machine Learning	Distributed	Reactive	Communication Events	OS-Independent
[59]	2006	Machine Learning	Local	Passive	Communication Events	OS-Independent
[120]	2006	Signatures (Automatically)	Local	Passive	OS Events	Symbian
[133]	2006	Run-Time Policy Enforcement	Local	Passive	OS Events	OS-Independent
[127]	2007	Run-Time Policy Enforcement	All	Active	All	OS-Independent
[156]	2007	Integrity Verification	Local	Passive	OS Events	SELinux
[130]	2007	Machine Learning	Local	Active	All	OS-Independent
[47]	2008	Signatures (Manually)	Distributed	Passive	Applications	Symbian
[157]	2008	Integrity Verification	Local	Passive	OS Events	SELinux
[97]	2008	Power Consumption	Distributed	Passive	Measurements	Windows Mobile
[121]	2008	Signatures (Automatically)	Distributed	Active	Communication Events	Symbian
[137]	2008	Anomaly Detection	Local	Passive	Keystrokes	OS-Independent
[148]	2008	Anomaly Detection	Distributed	Passive	All	Android
[154]	2008	Signatures (Automatically)	Local	Active	OS Event	Windows Mobile
[58]	2009	Machine Learning	Local	Passive	Communication Events	OS-Independent
[46]	2009	Machine Learning	Local	Active	Communication Events	OS-Independent
[48]	2009	Machine Learning	Distributed	Passive	Measurements	OS-Independent
[118]	2009	Power Consumption	Local	Passive	Communication Events	OS-Independent
[123]	2009	Signatures (Automatically)	Local	Active	All	OS-Independent
[39]	2009	Machine Learning	Distributed	Passive	OS Events	OS-Independent
[138]	2009	Machine Learning	Local	Passive	Keystrokes	OS-Independent
[141]	2009	Machine Learning	Local	Passive	Communication Events	OS-Independent
[143]	2009	Machine Learning	Local	Passive	All	Symbian
[147]	2009	Signatures (Manually)	Local	Passive	OS Events	Android
[158]	2009	Integrity Verification	Local	Passive	OS Events	LIMO
[122]	2009	Signatures (Manually)	Distributed	Active	Communication Events	Linux
[94]	2009	Run-Time Policy Enforcement	Local	Active	All	Android
[95]	2009	Run-Time Policy Enforcement	Local	Active	All	Android
[134]	2009	Interception	Local	Passive	OS Events	Windows Mobile
[135]	2009	Signatures (Manually)	Local	Passive	OS Events	Symbian
[136]	2009	Signatures (Manually)	Local	Passive	OS Events	Android
[151]	2010	Run-Time Policy Enforcement	Local	Active	OS Event	Android + SELinux
[153]	2010	Anomaly Detection	Local	Passive	OS Event	Windows Mobile
[124]	2010	Signatures (Automatically)	Local	Passive	Keystrokes	OS-Independent
[49]	2010	Machine Learning	Local	Passive	OS Events	Linux
[113]	2010	Machine Learning	Local	Passive	All	Android
[115]	2011	Machine Learning	Local	Passive	All	Android

In [163] the authors propose a practical approach for the design and implementation of trusted mobile platform. The approach, based upon the concept of a trusted platform as a set of trusted engine, defines a method for the take-ownership of a device by the user and the migration (i.e., portability) of user credentials between devices.

[164] identifies three specific problems in the MTM specification and provides some possible solutions. The first one concerns the need of balancing some contrasting goals at the system-level designs, such as performance and power consumption. A suggested solution integrates some TPM features directly into a processor core as opposed to a monolithic implementation of all the functions in a separate module. The second problem considers which cryptographic algorithms a MTM must support: some algorithms, namely RSA and SHA-1, can either have bad performances or security weaknesses. The suggested solution considers elliptic curve cryptography as a viable alternative. Finally, the third problem is related to the implementation of cryptographic primitives: the authors propose a hardware/software solution as opposed to a hardware-based solution, which suffers from poor flexibility.

Further solutions can be found in [165].

VI. CONCLUSIONS

With the rapid proliferation of smartphones equipped with a lot of features, as multiple connections and sensors, the number of mobile malware is increasing. Differently from PC environment, solutions aimed at preventing the infection and the diffusion of malicious code in smartphone have to consider multiple factors: the limited resources available, including the power and the processing unit, the large number of features that can be exploited by the attackers, such as different kinds of connections, services, sensors and the privacy of the user.

In this work, first of all we have discussed the current scenario of mobile malware, by summarizing its evolution, along with some notable examples; we have also outlined likely future threats and reported some predictions for the near future. Secondly, we have categorized known attacks against smartphones, especially at the application level, focusing on how the attack is carried out and what is the goal of the attacker. Finally, we have reviewed current security solutions for smartphones focusing on existing mechanisms based upon intrusion detection and trusted mobile platforms.

ACKNOWLEDGEMENT

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions, which have greatly improved the quality of the paper.

Work partially supported by EU-funded project FP7-256980 NESSoS.

REFERENCES

[1] M. Kotadia, "Major smartphone worm by 2007," *Gartner Study*, June 2005.

[2] Gartner Research, "Gartner Says Worldwide Mobile Phone Sales Grew 35 Percent in Third Quarter 2010; Smartphone Sales Increased 96 Percent," 2010. [Online]. Available: http://www.gartner.com/it/page.jsp?id=1466313

[3] IMS Research, "Global Smartphones Sales Will Top 420 Million Devices in 2011, Taking 28 Percent of all Handsets, According to IMS Research," July 2011. [Online]. Available: http://imsresearch.com/press-release/Global_ Smartphones_Sales_Will_Top_420_Million_Devices_in _2011_Taking_28_Percent_of_all_Handsets_According_to_IMS_Research

[4] Q. Yan, Y. Li, T. Li, and R. Deng, "Insights into Malware: Detection and Prevention on Mobile Phones," in *Security Technology*, D. Ślzak, T.-h. Kim, W.-C. Fang, and K. P. Arnett, Eds. Springer Berlin Heidelberg, 2009, vol. 58, ch. 30, pp. 242–249.

[5] S. Coorporation, "Symantec Internet Security Threat Report Volume XVI," *Whitepaper*, vol. 16, Apr 2011.

[6] Kasperksy Lab, "Popular Porn Sites Distribute a New Trojan Targeting Android Smartphones," 2010. [Online]. Available: http://www.kaspersky.com/news?id=207576175

[7] C. Papathanasiou and N. J. Percoco, "This is not the droid you're looking for..." in *DEFCON 18*, July 2010.

[8] J. Bickford, R. O'Hare, A. Baliga, V. Ganapathy, and L. Iftode, "Rootkits on smart phones: attacks, implications and opportunities," in *Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications*, ser. HotMobile '10. New York, NY, USA: ACM, 2010, pp. 49–54.

[9] D. Damopoulos, G. Kambourakis, and S. Gritzalis, "iSAM: An iPhone Stealth Airborne Malware," in *Future Challenges in Security and Privacy for Academia and Industry*, ser. IFIP Advances in Information and Communication Technology, J. Camenisch, S. Fischer-Hübner, Y. Murayama, A. Portmann, and C. Rieder, Eds. Springer Boston, 2011, vol. 354, ch. 2, pp. 17–28.

[10] A. Gostev, "Mobile malware evolution: An overview," *Kaspersky Labs Report on Mobile Viruses*, 2006.

[11] M. Hypponen, "Malware Goes Mobile," *Scientific American*, vol. 295, no. 5, pp. 46–53, 2006.

[12] G. Lawton, "Is It Finally Time to Worry about Mobile Malware?" *Computer*, vol. 41, pp. 12–14, May 2008.

[13] M. Hypponen, "Mobile Security Review September 2010," F-Secure Labs, HelsinkiFinland, Tech. Rep., September 2010.

[14] A.-D. Schmidt and S. Albayrak, "Malicious Software for Smartphones," Technische Universität Berlin - DAI-Labor, Tech. Rep. TUB-DAI 02/08-01, February 2008, http://www.dai-labor.de.

[15] A. P. Felt, M. Finifter, E. Chin, S. Hanna, and D. Wagner, "Survey of Mobile Malware in the Wild," 2011. [Online]. Available: http://www.eecs.berkeley.edu/~afelt/malware.html

[16] N. Leavitt, "Mobile Phones: The Next Frontier for Hackers?" *Computer*, vol. 38, pp. 20–23, April 2005.

[17] F-Secure, "Liberty (Palm)," Aug 2000. [Online]. Available: http://www.f-secure.com/v-descs/lib_palm.shtml

[18] ——, "Bluetooth-Worm:SymbOS/Cabir," Jun 2004. [Online]. Available: http://www.f-secure.com/v-descs/ cabir.shtml

[19] McAfee Labs, "Mobile Security Report 2009," 2009. [Online]. Available: http://www.mcafee. com/us/resources/reports/rp-mobile-security-2009.pdf

[20] S. Töyssy and M. Helenius, "About malicious software in smartphones," *Journal in Computer Virology*, vol. 2, no. 2, pp. 109–119, 2006.

[21] S. Viveros, "The economic impact of malicious code in wireless mobile networks," in *3G Mobile Communication Technologies, 2003. 3G 2003. 4th International Conference on (Conf. Publ. No. 494)*, Jun 2003, pp. 1–6.

[22] K. Dunham, *Mobile malware attacks and defense*. Syngress, 2008.

[23] N. Leavitt, "Malicious Code Moves to Mobile Devices," *Computer*, vol. 33, pp. 16–19, December 2000.

[24] McAfee Labs, "2011 Threats Predictions," 2010. [Online]. Available: http://www.mcafee.com/ us/resources/reports/rp-threat-predictions-2011.pdf

[25] D. Barroso, "ZeuS Mitmo: Man-in-the-mobile," Semptember 2010. [Online]. Available: http:// securityblog.s21sec.com/2010/09/zeus-mitmo-man-in- mobile-i.html

[26] A. Gostev, "Kaspersky Security Bulletin. Malware Evolution 2010," Feb 2011. [Online]. Available: http://www.securelist.com/en/analysis/204792161/Kas- persky_Security_Bulletin_Malware_Evolution_2010

[27] Cisco, "Cisco 2010 Annual Security Report," Jan 2011. [Online]. Available: http://www.cisco.com/ en/US/prod/collateral/vpndevc/security_annual_report_ 2010.pdf

[28] PandaLabs, "Annual Report," 2011. [Online]. Available: http://press.pandasecurity.com/wp-content/uploads/2010/ 05/PandaLabs-Annual-Report-2010.pdf

[29] Google Mobile Blog, "An Update on Android Market Security," March 2011. [Online]. Available: http://googlemobile.blogspot.com/2011/03/update-on-an- droid-market-security.html

[30] Sophos, "Security Threat Report," 2010. [Online]. Available: http://www.sophos.com/sophos/docs/eng/ papers/sophos-security-threat-report-jan-2010-wpna.pdf

[31] Juniper Networks Global Threat Center, "Malicious Mobile Threats Report 2010/2011," 2011. [Online]. Available: http://www.juniper.net/us/en/local/pdf/whitepapers/2000415-en.pdf

[32] Trend Micro, "The Future of Threats and Threat Technologies. How the Landscape Is Changing," 2011. [Online]. Available: http:// us.trendmicro.com/imperia/md/content/us/trendwatch/ researchandanalysis/trend_micro_2010_future_threat_report_final.pdf

[33] R. Leszczyna, I. N. Fovino, and M. Masera, "MAISim: mobile agent malware simulator," in *Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*, ser. Simutools '08. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, pp. 35:1–35:6.

[34] G. Hogben and M. Dekker, "Smartphones: Information security risks, opportunities and recommendations for users," European Network and Information Security Agency, 710 01 Heraklion - Crete - Greece, Tech. Rep., December 2010.

[35] P. M. Milligan and D. Hutcheson, "Business risks and security assessment for mobile devices," in *MCBE'07: Proceedings of the 8th Conference on 8th WSEAS Int. Conference on Mathematics and Computers in Business and Economics*. Stevens Point, Wisconsin, USA: World Scientific and Engineering Academy and Society (WSEAS), 2007, pp. 189–193.

[36] A. P. Felt, M. Finifter, E. Chin, S. Hanna, and D. Wagner, "A survey of mobile malware in the wild," in *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*, ser. SPSM '11. New York, NY, USA: ACM, 2011, pp. 3–14. [Online]. Available: http://doi.acm.org/10.1145/2046614.2046618

[37] R. A. Botha, S. M. Furnell, and N. L. Clarke, "From desktop to mobile: Examining the security experience," *Computers & Security*, vol. 28, no. 3-4, pp. 130–137, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/B6V8G- 4TYJTT6-1/2/335760cc014d6e724fd8d188942025e6

[38] C. R. Mulliner, "Security of Smart Phones," Master's thesis, University of California, Santa Barbara, 2006.

[39] G. Portokalidis, P. Homburg, N. FitzRoy-Dale, K. Anagnostakis, and H. Bos, "Protecting smart phones by means of execution replication," Vrije Universiteit Amsterdam, Tech. Rep., 2009.

[40] M. Becher, F. C. Freiling, J. Hoffmann, T. Holz, S. Uellenbeck, and C. Wolf, "Mobile Security Catching Up? Revealing the Nuts and Bolts of the Security of Mobile Devices," in *Proceedings IEEE Security and Privacy*, May 2011.

[41] J. Oberheide, K. Veeraraghavan, E. Cooke, J. Flinn, and F. Jahanian, "Virtualized in-cloud security services for mobile devices," in *Proceedings of the First Workshop on Virtualization in Mobile Computing*, ser. MobiVirt '08. New York, NY, USA: ACM, 2008, pp. 31–35.

[42] J. Oberheide and F. Jahanian, "When mobile is harder than fixed (and vice versa): demystifying security challenges in mobile environments," in *Proceedings of the Eleventh Workshop on Mobile Computing Systems & Applications*, ser. HotMobile '10. New York, NY, USA: ACM, 2010, pp. 43–48.

[43] M. Becher and F. C. Freiling, "Towards dynamic malware analysis

to increase mobile device security," in *Proc. of SICHERHEIT*, 2008.

[44] A. Makhlouf and N. Boudriga, "Intrusion and anomaly detection in wireless networks," in *Handbook of Research on Wireless Security*, Y. Zhan, J. Zheng, and M. Ma, Eds. Information Science Publishing, 2008.

[45] K. Haataja, "Security threats and countermeasures in Bluetooth-enabled systems," Ph.D. dissertation, Department of Computer Science, University of Kuopio, 2009.

[46] Y. L. Ho and S.-H. Heng, "Mobile and ubiquitous malware," in *MoMM '09: Proceedings of the 7th International Conference on Advances in Mobile Computing and Multimedia*. New York, NY, USA: ACM, 2009, pp. 559–563.

[47] A. Bose, X. Hu, K. G. Shin, and T. Park, "Behavioral detection of malware on mobile handsets," in *MobiSys '08: Proceeding of the 6th international conference on Mobile systems, applications, and services*. New York, NY, USA: ACM, 2008, pp. 225–238.

[48] A.-D. Schmidt, F. Peters, F. Lamour, C. Scheel, S. A. Çamtepe, and S. Albayrak, "Monitoring smartphones for anomaly detection," *Mob. Netw. Appl.*, vol. 14, no. 1, pp. 92–106, 2009.

[49] L. Xie, X. Zhang, J.-P. Seifert, and S. Zhu, "pBMDS: a behavior-based malware detection system for cellphone devices," in *Proceedings of the Third ACM Conference on Wireless Network Security, WISEC 2010, Hoboken, New Jersey, USA, March 22-24, 2010*. ACM, 2010, pp. 37–48.

[50] M. Becher and F. C. Freiling, "Towards Dynamic Malware Analysis to Increase Mobile Device Security," in *Sicherheit 2008: Sicherheit, Schutz und Zuverlässigkeit. Konferenzband der 4. Jahrestagung des Fachbereichs Sicherheit der Gesellschaft für Informatik e.V. (GI), 2.-4. April 2008 im Saarbrücker Schloss*, ser. LNI, vol. 128. GI, 2008, pp. 423–433.

[51] P. Traynor, M. Lin, M. Ongtang, V. Rao, T. Jaeger, P. McDaniel, and T. La Porta, "On cellular botnets: measuring the impact of malicious devices on a cellular network core," in *CCS '09: Proceedings of the 16th ACM conference on Computer and communications security*. New York, NY, USA: ACM, 2009, pp. 223–234.

[52] W. Enck, P. Traynor, P. McDaniel, and T. La Porta, "Exploiting open functionality in SMS-capable cellular networks," in *Proceedings of the 12th ACM conference on Computer and communications security*, ser. CCS '05. New York, NY, USA: ACM, 2005, pp. 393–404.

[53] V. Bocan and V. Cretu, "Security and Denial of Service Threats in GSM Networks," *Periodica Politechnica, Transactions on Automatic Control and Computer Science*, vol. 49, no. 63, 2004.

[54] C. Xenakis, "Malicious actions against the GPRS technology," *Journal in Computer Virology*, vol. 2, no. 2, pp. 121–133, 2006.

[55] J. R. Rao, P. Rohatgi, H. Scherzer, and S. Tinguely, "Partitioning Attacks: Or How to Rapidly Clone Some GSM Cards," in *Proceedings of the 2002 IEEE Symposium on Security and Privacy*. Washington, DC, USA: IEEE Computer Society, 2002, pp. 31–.

[56] G. Kambourakis, C. Kolias, S. Gritzalis, and J. H. Park, "DoS attacks exploiting signaling in UMTS and IMS," *Computer Communications*, vol. 34, no. 3, pp. 226–235, 2011.

[57] A. Bose and K. G. Shin, "Proactive security for mobile messaging networks," in *WiSe '06: Proceedings of the 5th ACM workshop on Wireless security*. New York, NY, USA: ACM, 2006, pp. 95–104.

[58] Z. Zhu, G. Cao, S. Zhu, S. Ranjan, and A. Nucci, "A Social Network Based Patching Scheme for Worm Containment in Cellular Networks," in *INFOCOM 2009. 28th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 19-25 April 2009, Rio de Janeiro, Brazil*, 2009, pp. 1476–1484.

[59] A. Bose and K. G. Shin, "On Mobile Viruses Exploiting Messaging and Bluetooth Services," in *Securecomm and Workshops, 2006*, Sept 2006, pp. 1–10.

[60] U. Meyer and S. Wetzel, "A man-in-the-middle attack on UMTS," in *Proceedings of the 3rd ACM workshop on Wireless security*, ser. WiSe '04. New York, NY, USA: ACM, 2004, pp. 90–97.

[61] M. Khan, A. Ahmed, and A. R. Cheema, "Vulnerabilities of UMTS Access Domain Security Architecture," in *Proceedings of the 2008 Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 350–355.

[62] C. Guo, H. J. Wang, and W. Zhu, "Smart-phone attacks and defenses," in *HotNets III*. Citeseer, 2004.

[63] J. W. Mickens and B. D. Noble, "Modeling epidemic spreading in mobile environments," in *WiSe '05: Proceedings of the 4th ACM workshop on Wireless security*. New York, NY, USA: ACM, 2005, pp. 77–86.

[64] C. Rhodes and M. Nekovee, "The opportunistic transmission of wireless worms between mobile devices," *Physica A: Statistical Mechanics and its Applications*, vol. 387, no. 27, pp. 6837–6844, 2008. [Online]. Available: http://www.science- direct.com/science/article/pii/S0378437108007772

[65] C. Fleizach, M. Liljenstam, P. Johansson, G. M. Voelker, and A. Mehes, "Can you infect me now?: malware propagation in mobile phone networks," in *WORM '07: Proceedings of the 2007 ACM workshop on Recurring malcode*. New York, NY, USA: ACM, 2007, pp. 61–68.

[66] G. Yan, H. D. Flores, L. Cuellar, N. Hengartner, S. Eidenbenz, and V. Vu, "Bluetooth worm propagation: mobility pattern matters!" in *ASIACCS '07: Proceedings of the 2nd ACM symposium on Information, computer and communications security*. New York, NY, USA: ACM, 2007, pp. 32–44.

[67] G. Yan and S. Eidenbenz, "Bluetooth Worms: Models, Dynamics, and Defense Implications," in *ACSAC '06: Proceedings of the 22nd Annual Computer Security Applications Conference*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 245–256.

[68] Y. Bulygin, "Epidemics of Mobile Worms," *Performance, Computing, and Communications Conference, 2002. 21st IEEE International*, vol. 0, pp. 475–478, 2007.

[69] J. W. Mickens and B. D. Noble, "Analytical Models for Epidemics in Mobile Networks," in *WIMOB '07: Proceedings of the Third IEEE International Conference on Wireless and Mobile Computing, Networking and Communications*. Washington, DC, USA: IEEE Computer Society, 2007, p. 77.

[70] G. Yan and S. Eidenbenz, "Modeling Propagation Dynamics of Bluetooth Worms (Extended Version)," *IEEE Transactions on Mobile Computing*, vol. 8, pp. 353–368, 2009.

[71] J. Su, K. K. W. Chan, A. G. Miklas, K. Po, A. Akhavan, S. Saroiu, E. de Lara, and A. Goel, "A preliminary investigation of worm infections in a bluetooth environment," in *WORM '06: Proceedings of the 4th ACM workshop on Recurring malcode*. New York, NY, USA: ACM, 2006, pp. 9–16.

[72] M. Becher, F. C. Freiling, and B. Leider, "On the Effort to Create Smartphone Worms in Windows Mobile," in *Information Assurance and Security Workshop, 2007. IAW '07. IEEE SMC*, june 2007, pp. 199–206.

[73] A. Lelli, "A Smart Worm for a Smartphone WinCE.PmCryptic.A," 2009. [Online]. Available: http://www.symantec.com/connect/blogs/smart-worm- smartphone-wincepmcryptica

[74] A. R. Fløe and A. Jøsang, "Consequences of botnets spreading to mobile devices," in *14th Nordic Conference on Secure IT Systems*, 2009, pp. 37–43.

[75] K. Singh, S. Sangal, N. Jain, P. Traynor, and W. Lee, "Evaluating Bluetooth as a medium for botnet command and control," in *Proceedings of the 7th international conference on Detection of intrusions and malware, and vulnerability assessment*, ser. DIMVA'10. Berlin, Heidelberg: Springer-Verlag, 2010, pp. 61–80.

[76] K. G. S. Yuanyuan Zeng, Xin Hu, "How to Construct a Mobile Botnet?" in *The 40th Annual IEEE/IFIP International Conference on Dependable Systems and Networks (DSN 2010)*, 2010.

[77] C. Mulliner and J.-P. Seifert, "Rise of the iBots: Owning a telco network," in *Malicious and Unwanted Software (MALWARE), 2010 5th International Conference on*, Oct 2010, pp. 71–80.

[78] P. A. Porras, H. Saïdi, and V. Yegneswaran, "An Analysis of the iKee.B iPhone Botnet," in *Security and Privacy in Mobile Information and Communication Systems - Second International ICST Conference, MobiSec 2010, Catania, Sicily, Italy, May 27-28, 2010, Revised Selected Papers*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, A. U. Schmidt, G. Russello, A. Lioy, N. R. Prasad, and S. Lian, Eds., vol. 47. Springer, 2010, pp. 141–152.

[79] A. Apvrille, "Symbian Worm Yxes: Towards Mobile Botnets?" in *The 19th EICAR Annual Conference*, May 2010, pp. 31–54.

[80] M. Ballano, "Android Threats Getting Steamy," Feb 2011. [Online]. Available: http://www.symantec. com/connect/blogs/android-threats-getting-steamy

[81] M. Becher, "Security of Smartphones at the Dawn of their Ubiquitousness," Ph.D. dissertation, Universität Mannheim, 2009.

[82] Techie Buzz, "Android Data Theft Vulnerability Detailed," 2011. [Online]. Available: http://techie-buzz.com/mobile-news/android-data-theft- vulnerability-detailed.html

[83] L. Whitney, "Apple sued over privacy in iPhone, iPad apps," 2011. [Online]. Available: http://news.cnet.com/8301-13579_3-20026677-37.html

[84] N. Seriot, "iPhone Privacy," *Black Hat DC*, 2010. [Online]. Available: http://seriot.ch/resources/talks_papers/iPhonePrivacy.pdf

[85] S. Bhatt, R. Sion, and B. Carbunar, "A personal mobile DRM manager for smartphones," *Computers & Security*, vol. 28, no. 6, pp. 327–340, 2009.

[86] R. P. Minch, "Privacy Issues in Location-Aware Mobile Devices," in *Proceedings of the Proceedings of the 37th Annual Hawaii International Conference on System Sciences (HICSS'04) - Track 5 - Volume 5*. Washington, DC, USA: IEEE Computer Society, 2004, pp. 50 127.2–.

[87] W. Enck, P. Gilbert, B.-G. Chun, L. P. Cox, J. Jung, P. McDaniel, and A. N. Sheth, "TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones," in *Proceedings of the 9th USENIX conference on Operating systems design and implementation*, ser. OSDI'10. Berkeley, CA, USA: USENIX Association, 2010, pp. 1–6.

[88] M. Egele, C. Kruegel, E. Kirda, and G. Vigna, "PiOS: Detecting Privacy Leaks in iOS Applications," in *Proceedings of the Network and Distributed System Security Symposium (NDSS)*, San Diego, CA, February 2011.

[89] S. Whitehead, J. Mailley, I. Storer, J. McCardle, G. Torrens, and G. Farrell, "IN SAFE HANDS: A Review of Mobile Phone Anti-theft Designs," *European Journal on Criminal Policy and Research*, vol. 14, pp. 39–60, 2008.

[90] A. Portnoy, "Pwn2Own 2010," 2010. [Online]. Available: http://dvlabs.tippingpoint.com/blog/2010/ 02/15/pwn2own-2010

[91] L. Cai, S. Machiraju, and H. Chen, "Defending against sensor-sniffing attacks on mobile phones," in *Proceedings of the 1st ACM workshop on Networking, systems, and applications for mobile handhelds*, ser. MobiHeld '09. New York, NY, USA: ACM, 2009, pp. 31–36.

[92] N. Xu, F. Zhang, Y. Luo, W. Jia, D. Xuan, and J. Teng, "Stealthy video capturer: a new video-based spyware in 3G smartphones," in *Proceedings of the second ACM conference on Wireless network security*, ser. WiSec '09. New York, NY, USA: ACM, 2009, pp. 69–78.

[93] R. Schlegel, K. Zhang, X. Zhou, M. Intwala, A. Kapadia, and X. Wang, "Soundminer: A Stealthy and Context-Aware Sound Trojan for Smartphones," in *Proceedings of the 18th Annual Network & Distributed System Security Symposium (NDSS)*, Feb. 2011.

[94] W. Enck, M. Ongtang, and P. McDaniel, "On lightweight mobile phone application certification," in *CCS '09: Proceedings of the 16th ACM conference on Computer and communications security*. New York, NY, USA: ACM, 2009, pp. 235–245.

[95] M. Ongtang, S. McLaughlin, W. Enck, and P. McDaniel, "Semantically Rich Application-Centric Security in Android," in *Computer Security Applications Conference, 2009. ACSAC '09. Annual*, dec. 2009, pp. 340–349.

[96] E. Naone, "SMS of Death Could Crash Many Mobile Phones," 2011. [Online]. Available: http://www. technologyreview.com/printer_friendly_article.aspx?id =27021

[97] H. Kim, J. Smith, and K. G. Shin, "Detecting energy-greedy anomalies and mobile malware variants," in *MobiSys '08: Proceeding of the 6th international conference on Mobile systems, applications, and services*. New York, NY, USA: ACM, 2008, pp. 239–252.

[98] L. Liu, X. Zhang, G. Yan, and S. Chen, "Exploitation and threat analysis of open mobile devices," in *Proceedings of the 5th ACM/IEEE Symposium on Architectures for Networking and Communications Systems*, ser. ANCS '09. New York, NY, USA: ACM, 2009, pp. 20–29.

[99] R. Racic, D. Ma, and H. Chen, "Exploiting MMS Vulnerabilities to Stealthily Exhaust Mobile Phone's Battery," in *Securecomm and Workshops, 2006*, aug. 2006, pp. 1–10.

[100] D. Johnston and J. Walker, "Overview of IEEE 802.16 security," *Security Privacy, IEEE*, vol. 2, no. 3, pp. 40–48, may-june 2004.

[101] T. Engel, "Remote SMS/MMS Denial of Service - Curse Of Silence for Nokia S60 phones," Dec 2008. [Online]. Available: http://berlin.ccc.de/~tobias/cursesms.txt

[102] C. Mulliner and C. Miller, "Injecting SMS messages into smart phones for security analysis," in *WOOT'09: Proceedings of the 3rd USENIX conference on Offensive technologies*. Berkeley, CA, USA: USENIX Association, 2009, pp. 5–5.

[103] McAfee, "WaveSecure," 2011. [Online]. Available: https://www.wavesecure.com/

[104] Norton, "Norton Mobile Security Lite," 2011. [Online]. Available: http://us.norton.com/mobile-security/

[105] IIT-CNR, "iCareMobile," 2011. [Online]. Available: http://icaremobile.iit.cnr.it/

[106] BullGuard Ltd, "BullGuard Mobile Security 10," 2011. [Online]. Available: http://www.bullguard.com

[107] Kaspersky Lab ZAO, "Kaspersky Mobile Security 9," 2011. [Online]. Available: http://www.kaspersky.com/kaspersky_mobile_security

[108] ESET, "ESET Mobile Security," 2011. [Online]. Available: http://www.eset.com/us/home/products/mobile-security/

[109] Lookout, "Lookout Mobile Security," 2011. [Online]. Available: https://www.mylookout.com

[110] C. Xenakis and L. Merakos, "Vulnerabilities and Possible Attacks Against the GPRS Backbone Network," in *Critical Information Infrastructures Security*, ser. Lecture Notes in Computer Science, J. Lopez, Ed. Springer Berlin / Heidelberg, 2006, vol. 4347, pp. 262–272.

[111] B. Sun, Y. Xiao, and K. Wu, "Intrusion Detection in Cellular Mobile Networks," in *Wireless Network Security*, ser. Signals and Communication Technology, Y. Xiao, X. S. Shen, and D.-Z. Du, Eds. Springer US, 2007, pp. 183–210.

[112] G. W. Chow and A. Jones, "A Framework for Anomaly Detection in OKL4-Linux Based Smartphones," in *Australian Information Security Management Conference*, 2008.

[113] A. Shabtai, U. Kanonov, and Y. Elovici, "Intrusion detection for mobile devices using the knowledge-based, temporal abstraction method," *J. Syst. Softw.*, vol. 83, no. 8, pp. 1524–1537, 2010.

[114] D. Damopoulos, S. A. Menesidou, G. Kambourakis, M. Papadaki, N. Clarke, and S. Gritzalis, "Evaluation of anomaly-based IDS for mobile devices using machine learning classifiers," *Security and Communication Networks*, vol. 5, no. 1, pp. 3–14, 2012. [Online]. Available: http://dx.doi.org/10.1002/sec.341

[115] A. Shabtai, U. Kanonov, Y. Elovici, C. Glezer, and Y. Weiss, ""Andromaly": a behavioral malware detection framework for android devices," *Journal of Intelligent Information Systems*, pp. 1–30, 2011, 10.1007/s10844-010-0148-x. [Online]. Available: http://dx.doi.org/10.1007/s10844-010-0148-x

[116] I. Burguera, U. Zurutuza, and S. Nadjm-Tehrani, "Crowdroid: behavior-based malware detection system for Android," in *Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices*, ser. SPSM '11. New York, NY, USA: ACM, 2011, pp. 15–26. [Online]. Available: http://doi.acm.org/10.1145/2046614.2046619

[117] G. A. Jacoby, R. Marchany, and N. J. D. IV, "How Mobile Host Batteries Can Improve Network Security," *IEEE Security and Privacy*, vol. 4, pp. 40–49, 2006.

[118] L. Liu, G. Yan, X. Zhang, and S. Chen, "VirusMeter: Preventing Your Cellphone from Spies," in *RAID '09: Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 244–264.

[119] Q. Yan, R. H. Deng, Y. Li, and T. Li, "On the Potential of Limitation-oriented Malware Detection and Prevention Techniques on Mobile Phones," *International Journal of Security and Its Applications (IJSIA)*, vol. 4(1), pp. 21–30, Jan 2010.

[120] D. Venugopal, G. Hu, and N. Roman, "Intelligent virus detection on mobile devices," in *PST '06: Proceedings of the 2006 International Conference on Privacy, Security and Trust*. New York, NY, USA: ACM, 2006, pp. 1–4.

[121] L. Xie, H. Song, T. Jaeger, and S. Zhu, "A systematic approach for cell-phone worm containment," in *WWW '08: Proceeding of the 17th international conference on World Wide Web*. New York, USA: ACM, 2008, pp. 1083–1084.

[122] L. Xie, X. Zhang, A. Chaugule, T. Jaeger, and S. Zhu, "Designing System-Level Defenses against Cellphone Malware," in *SRDS '09: Proceedings of the 2009 28th IEEE International Symposium on Reliable Distributed Systems*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 83–90.

[123] G. Zyba, G. M. Voelker, M. Liljenstam, A. Méhes, and P. Johansson, "Defending Mobile Phones from Proximity Malware," in *INFOCOM 2009. 28th IEEE International Conference on Computer Communications, Joint Conference of the IEEE Computer and Communications Societies, 19-25 April 2009, Rio de Janeiro, Brazil*. IEEE, 2009, pp. 1503–1511.

[124] C. Bauckhage, T. Alpcan, and A.-D. Schmidt, "A Probabilistic Diffusion Scheme for Anomaly Detection on Smartphones," in *Proceedings of the Fourth International Workshop in Information Security Theory and Practice 2010 (WISTP'10)*, ser. Lecture Notes in Computer Science. Springer, 2010, pp. 31–46.

[125] D. R. Ellis, J. G. Aiken, K. S. Attwood, and S. D. Tenaglia, "A behavioral approach to worm detection," in *WORM '04: Proceedings of the 2004 ACM workshop on Rapid malcode*. New York, NY, USA: ACM, 2004, pp. 43–53.

[126] A. Castrucci, F. Martinelli, P. Mori, and F. Roperti, "Enhancing Java ME Security Support with Resource Usage Monitoring," in *Information and Communications Security, 10th International Conference,*

ICICS 2008, Birmingham, UK, October 20-22, 2008, Proceedings, ser. Lecture Notes in Computer Science, L. Chen, M. D. Ryan, and G. Wang, Eds., vol. 5308. Springer, 2008, pp. 256–266.

[127] N. Dragoni, F. Massacci, K. Naliuka, and I. Siahaan, "Security-by-Contract: Toward a Semantics for Digital Signatures on Mobile Code," in *Public Key Infrastructure, 4th European PKI Workshop: Theory and Practice, EuroPKI 2007, Palma de Mallorca, Spain, June 28-30, 2007, Proceedings*, ser. Lecture Notes in Computer Science, vol. 4582. Springer, 2007, pp. 297–312.

[128] N. Dragoni, F. Martinelli, F. Massacci, P. Mori, C. Schaefer, T. Walter, and E. Vetillard, "Security-by-Contract (SxC) for Software and Services of Mobile Systems," in *At your service: Service Engineering in the Information Society Technologies Program*. MIT press, 2008.

[129] G. Costa, N. Dragoni, A. Lazouski, F. Martinelli, F. Massacci, and I. Matteucci, "Extending Security-by-Contract with Quantitative Trust on Mobile Devices," *Complex, Intelligent and Software Intensive Systems, International Conference*, vol. 0, pp. 872–877, 2010.

[130] J. Cheng, S. H. Wong, H. Yang, and S. Lu, "SmartSiren: virus detection and alert for smartphones," in *MobiSys '07: Proceedings of the 5th international conference on Mobile systems, applications and services*. New York, NY, USA: ACM, 2007, pp. 258–271.

[131] E. V. Ruitenbeek, T. Courtney, W. H. Sanders, and F. Stevens, "Quantifying the Effectiveness of Mobile Phone Virus Response Mechanisms," in *DSN '07: Proceedings of the 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks*. Washington, DC, USA: IEEE Computer Society, 2007, pp. 790–800.

[132] M. Miettinen and P. Halonen, "Host-Based Intrusion Detection for Advanced Mobile Devices," in *AINA '06: Proceedings of the 20th International Conference on Advanced Information Networking and Applications*. Washington, DC, USA: IEEE Computer Society, 2006, pp. 72–76.

[133] C. Mulliner, G. Vigna, D. Dagon, and W. Lee, "Using Labeling to Prevent Cross-Service Attacks Against Smart Phones," in *Proceedings of the Conference on Detection of Intrusions and Malware, and Vulnerability Assessment (DIMVA)*, ser. LNCS, vol. 4064. Berlin, Germany: Springer, July 2006, pp. 91–108.

[134] M. Becher and R. Hund, "Kernel-Level Interception and Applications on Mobile Devices," Department for Mathematics and Computer Science, University of Mannheim, Tech. Rep. TR-2008-003, 2009.

[135] A.-D. Schmidt, J. H. Clausen, S. A. Camtepe, and S. Albayrak, "Detecting Symbian OS Malware through Static Function Call Analysis," in *Proceedings of the 4th IEEE International Conference on Malicious and Unwanted Software (Malware 2009)*. IEEE, 2009, pp. 15–22.

[136] A.-D. Schmidt, R. Bye, H.-G. Schmidt, J. H. Clausen, O. Kiraz, K. A. Yüksel, S. A. Çamtepe, and S. Albayrak, "Static Analysis of Executables for Collaborative Malware Detection on Android," in *Proceedings of IEEE International Conference on Communications, ICC 2009, Dresden, Germany, 14-18 June 2009*. IEEE, 2009, pp. 1–5.

[137] T. Isohara, K. Takemori, and I. Sasase, "Anomaly Detection on Mobile Phone Based Operational Behavior," *IPSJ Digital Courier*, vol. 4, no. 0, pp. 9–17, 2008.

[138] S. Zahid, M. Shahzad, S. Khayam, and M. Farooq, "Keystroke-Based User Identification on Smart Phones," in *Recent Advances in Intrusion Detection*, ser. Lecture Notes in Computer Science, E. Kirda, S. Jha, and D. Balzarotti, Eds. Springer Berlin / Heidelberg, 2009, vol. 5758, pp. 224–243.

[139] T. S. Yap and H. T. Ewe, "A Mobile Phone Malicious Software Detection Model with Behavior Checker," in *Web and Communication Technologies and Internet-Related Social Issues - HSI 2005, 3rd International Conference on Human.Society@Internet, Tokyo, Japan, July 27-29, 2005, Proceedings*, ser. Lecture Notes in Computer Science, vol. 3597. Springer, 2005, pp. 57–65.

[140] C. Mulliner and G. Vigna, "Vulnerability Analysis of MMS User Agents," in *Computer Security Applications Conference, 2006. ACSAC '06. 22nd Annual*, dec. 2006, pp. 77–88.

[141] G. Yan, S. Eidenbenz, and E. Galli, "SMS-Watchdog: Profiling Social Behaviors of SMS Users for Anomaly Detection," in *RAID '09: Proceedings of the 12th International Symposium on Recent Advances in Intrusion Detection*. Berlin, Heidelberg: Springer-Verlag, 2009, pp. 202–223.

[142] D. S. Troy Vennon, "Threat Analysis of the Android Market," SMobile Systems, Tech. Rep., June 2010.

[143] S. M. Habib, C. Jacob, and T. Olovsson, "An Analysis of the Robustness and Stability of the Network Stack in Symbian-based Smartphones," *Journal of Networks*, vol. 4, no. 10, pp. 968–975, 2009.

[144] T. Badura and M. Becher, "Testing the Symbian OS Platform Security Architecture," in *Advanced Information Networking and Applications, 2009. AINA '09. International Conference on*, may 2009, pp. 838–844.

[145] J. de Haas, "Symbian phone Security," 2005. [Online]. Available: http://www.blackhat.com/ presentations/bh-europe-05/ BH_EU_05-deHaas.pdf

[146] A.-D. Schmidt, H.-G. Schmidt, J. Clausen, K. A. Yksel, O. Kiraz, A. Camtepe, and S. Albayrak, "Enhancing Security of Linux-based Android Devices," in *Proceedings of 15th International Linux Kongress*. Lehmann, Oct 2008.

[147] A.-D. Schmidt, H.-G. Schmidt, L. Batyuk, J. H. Clausen, S. A. Camtepe, S. Albayrak, and C. Yildizli, "Smartphone Malware Evolution Revisited: Android Next Target?" in *Proceedings of the 4th IEEE International Conference on Malicious and Unwanted Software (Malware 2009)*. IEEE, 2009, pp. 1–7.

[148] A.-D. Schmidt, R. Bye, H.-G. Schmidt, K. A. Yksel, O. Kiraz, J. Clausen, K. Raddatz, A. Camtepe, and S. Albayrak, "Monitoring Android for Collaborative Anomaly Detection: A First Architectural Draft," Technische Universität Berlin - DAI-Labor, Tech. Rep. TUB-DAI 08/08-02, Aug 2008. [Online]. Available: http://www.dai-labor.de/fileadmin/files/publications/080 8-02_DAI_TechReport_ Monitoring_Android.pdf

[149] W. Enck, M. Ongtang, and P. McDaniel, "Understanding Android Security," *IEEE Security and Privacy*, vol. 7, pp. 50–57, January 2009.

[150] A. Shabtai, Y. Fledel, U. Kanonov, Y. Elovici, S. Dolev, and C. Glezer, "Google Android: A Comprehensive Security Assessment," *IEEE Security and Privacy*, vol. 8, pp. 35–44, 2010.

[151] A. Shabtai, Y. Fledel, and Y. Elovici, "Securing Android-Powered Mobile Devices Using SELinux," *IEEE Security and Privacy*, vol. 8, pp. 36–44, May 2010.

[152] W. Enck, D. Octeau, P. McDaniel, and S. Chaudhuri, "A study of Android application security," in *Proceedings of the 20th USENIX conference on Security*, ser. SEC'11. Berkeley, CA, USA: USENIX Association, 2011, pp. 21–21. [Online]. Available: http://dl.acm.org/citation.cfm?id=2028067.2028088

[153] S. Dai, Y. Liu, T. Wang, T. Wei, and W. Zou, "Behavior-Based Malware Detection on Mobile Phone," in *Wireless Communications Networking and Mobile Computing (WiCOM), 2010 6th International Conference on*, Sept 2010, pp. 1–4.

[154] Y. Ikebe, T. Nakayama, M. Katagiri, S. Kawasaki, H. Abe, T. Shinagawa, and K. Kato, "Efficient Anomaly Detection System for Mobile Handsets," in *SECURWARE '08: Proceedings of the 2008 Second International Conference on Emerging Security Information, Systems and Technologies*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 154–160.

[155] D. Reisinger, "Android, iPhone Security: 10 Ways to Avoid Personal Data Theft," 2011. [Online]. Available: http://tinyurl.com/239npbj

[156] X. Zhang, O. Aciiçmez, and J.-P. Seifert, "A trusted mobile phone reference architecture via secure kernel," in *STC '07: Proceedings of the 2007 ACM workshop on Scalable trusted computing*. New York, NY, USA: ACM, 2007, pp. 7–14.

[157] D. Muthukumaran, A. Sawani, J. Schiffman, B. M. Jung, and T. Jaeger, "Measuring integrity on mobile phone systems," in *SACMAT '08: Proceedings of the 13th ACM symposium on Access control models and technologies*. New York, NY, USA: ACM, 2008, pp. 155–164.

[158] X. Zhang, O. Aciiçmez, and J.-P. Seifert, "Building Efficient Integrity Measurement and Attestation for Mobile Phone Platforms," in *Security and Privacy in Mobile Information and Communication Systems, First International ICST Conference, MobiSec 2009, Turin, Italy, June 3-5, 2009, Revised Selected Papers*, ser. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 17. Springer, 2009, pp. 71–82.

[159] Trusted Computing Group, "TCG TPM Main Part 1 Design Principles Specification Version 1.2, revision 62," Oct 2003.

[160] ——, "TCG Mobile Reference Architecture Specification Version 1.0, Revision 1," Jun 2007.

[161] R. Sailer, X. Zhang, T. Jaeger, and L. van Doorn, "Design and implementation of a TCG-based integrity measurement architecture," in *Proceedings of the 13th conference on USENIX Security Symposium - Volume 13*, ser. SSYM'04. Berkeley, CA, USA: USENIX Association, 2004, pp. 16–16.

[162] T. Jaeger, R. Sailer, and U. Shankar, "PRIMA: policy-reduced integrity measurement architecture," in *SACMAT '06: Proceedings of the eleventh ACM symposium on Access control models and technologies*. New York, NY, USA: ACM, 2006, pp. 19–28.

[163] A. U. Schmidt, N. Kuntze, and M. Kasper, "On the Deployment of Mobile Trusted Modules," in *Wireless Communications and Network-*

ing Conference, 2008. WCNC 2008. IEEE, mar. 2008, pp. 3169–3174.
[164] J. Grossschadl, T. Vejda, and D. Page, "Reassessing the TCG specifications for trusted computing in mobile and embedded systems," in *Proceedings of the 2008 IEEE International Workshop on Hardware-Oriented Security and Trust*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 84–90.
[165] O. Aciiçmez, A. Latifi, J.-P. Seifert, and X. Zhang, "A Trusted Mobile Phone Prototype," in *Consumer Communications and Networking Conference, 2008. CCNC 2008. 5th IEEE*, jan. 2008, pp. 1208–1209.

Mariantonietta La Polla received her Bachelor's and Master's degrees in Computer Engineering from University of Pisa respectively in 2006 and 2010. Currently she is a Research Fellow at IIT-CNR, Pisa. She is also a Ph.D. student in Information Engineering at the Engineering Ph.D. School "Leonardo da Vinci", University of Pisa. Her research interests include the study of techniques for Web acquisition and Web data analysis.

Fabio Martinelli (M.Sc. 1994, Ph.D. 1999) is a senior researcher of IIT-CNR, Pisa, where he is the scientific coordinator of the security group. His main research interests involve security and privacy in distributed and mobile systems and foundations of security and trust. He serves as PC-chair/organizer in several international conferences/workshops. He is the co-initiator of the International Workshop series on Formal Aspects in Security and Trust (FAST). He is serving as scientific co-director of the international research school on Foundations of Security Analysis and Design (FOSAD) since 2004 edition. He chairs the WG on security and trust management (STM) of the European Research Consortium in Informatics and Mathematics (ERCIM). He usually manages R&D projects on information and communication security and he is involved in several FP6/7 EU projects.

Daniele Sgandurra received his M.Sc. degree in Computer Science in 2006, and a Ph.D in Computer Science in 2010, both from the University of Pisa. He is currently a PostDoc Researcher at IIT-CNR, Pisa, where his main research fields include virtualization security, intrusion detection systems, critical infrastructures and risk management, cloud security and mobile security.