A Novel Approach to Trojan Horse Detection in Mobile Phones Messaging and Bluetooth Services

Juan A. Ortega[1], Daniel Fuentes[1], Juan A. Á lvarez[1], Luis Gonzalez-Abril[2] and Francisco Velasco[2]

[1] Computer Languages and Systems Department, University of Seville
Avda. Reina Mercedes s/n, 41012, Seville - Spain
[e-mail: {jortega, dfuentes, jaalvarez}@us.es]
[2] Applied Economics Department I, University of Seville
Avda. Ramón y Cajal 1, 41018, Seville - Spain
[e-mail: {luisgon, velasco}@us.es]
*Corresponding author: Daniel Fuentes

Received October 13, 2010; revised December 24, 2010; accepted January 24, 2011; published August 29, 2011

Abstract

A method to detect Trojan horses in messaging and Bluetooth in mobile phones by means of monitoring the events produced by the infections is presented in this paper. The structure of the detection approach is split into two modules: the first is the Monitoring module which controls connection requests and sent/received files, and the second is the Graphical User module which shows messages and, under suspicious situations, reports the user about a possible malware. Prototypes have been implemented on different mobile operating systems to test its feasibility on real cellphone malware. Experimental results are shown to be promising since this approach effectively detects various known malware.

Keywords: Security, mobile devices, mobile malware, spyware

This research is supported by the Spanish Ministry of Science and Innovation R&D project ARTEMISA (TIN2009-14378-C02-01).

DOI: 10.3837/tiis.2011.08.006

1. Introduction

Mobile devices currently represent the most wide-ranging and fastest technology. Even personal computers are not advancing, or improving as quickly as this growing technology. According to the International Telecommunication Union, even during a global crisis period, the number of mobile broadband subscriptions will exceed one billion in 2010 after reaching 600 million in 2009. This organization added that the ranks of cell phone subscribers will grow to five billion people in 2010 [1]. Today's smartphones have the same processing power and characteristics as a traditional PC. They do not only provide basic services, e.g. voice communication and text messaging, but also offer other data services such as GPS connection, emailing and web surfing. In addition to personal information in the form of contacts and emails, users can save large documents, high-quality videos and images, complex games and music albums. Nevertheless, security experts have found a growing number of infections that target cellular phones.

Currently most mobile users feel no need to install an antivirus program or other software on their terminals to protect their phones from potential infections. However, due to the exponential growth of services and capabilities of these devices and the vast amount of information they contain, it is essential to take measures against a possible attack. It is worth noting that the potential effects of virulent malware infections on consumers and phone providers are severe, including identity and information theft, permanent disablement of mobile devices, excessive fees to customers, and loss of revenue for mobile phone providers. In fact, when a user buys a mobile phone, none of the pre-installed applications is antivirus software. The most common procedure is that, if a mobile device becomes infected, the user downloads the corresponding patch to delete the malware, nevertheless the device stays unprotected.

Let us consider a cellphone malware as malicious code that exploits vulnerabilities in cellphone software and propagates in networks through popular services such as Bluetooth and messaging (SMS/MMS) services. There is also an overlap in how malware is termed, thus it could be called a trojan, backdoor, worm, or virus. It is rare that the predominant antivirus vendors will agree on the general category (much less the name) of a virus. The breakdown of malware is summarized in **Table 1** which shows the percentage of malware in 2009 that was labelled with a generic category by at least one vendor [2]. The values add up to much more than 100 percent, meaning many viruses are not only labelled as something generic, but also labelled with multiple generic names. It can be seen that trojans comprised the most common class of malware in 2009. Hence, the behaviour of the Trojan horses which use messaging and Bluetooth services in mobile phones is studied in this paper. Furthermore, it is indicative of the attacker’s preference to infect machines so that a host of malicious activities (advantageous to the attacker) can be launched from these devices. These malicious activities usually imply high or low-level events in the user's attacked phone such as an email reception or a Bluetooth connection request. These events are produced in the system since a connection to the attacked device and transmission of exchange information is necessary. During their execution, infections such as trojans wait for orders from an external malicious user who tries to send information to the infected phone by different means of communication, such as SMS [3]. Furthermore, malware can use the mobile phone network to propagate as on the Internet.

Mobile phone networks can quickly detect malware and employ defences to contain it before it infects much of the susceptible device population [4]. Nevertheless, mobile malware

has another opportunity for propagation since it can propagate through direct pair-wise communication mechanisms, such as Bluetooth and SMS, between mobile devices in geographic proximity [5,6]. Although slower than propagating over the network, proximity malware has the compelling advantage of being unobserved by the provider network making detecting proximity malware substantially more challenging. However, these connections produce events in both devices while the connection is established or files are sent. Hence, situations like the arrival of a new MMS message or the sending of a file through Bluetooth cause events at high (i.e. SMS received) or low (i.e. socket connection) level in an infected cell phone.

Table 1. 2009 Malware Trends. Many viruses can be labelled with multiple generic names.

Malware Class	Percentage of total malware
Trojan Horse	87
Virus	47
PUP	42
Worm	21
Backdoor	20

By using mobile events and taking advantage of high processing capabilities of modern cell phones, a new method is proposed that can be executed in background and real time, which monitors and analyzes suspected events to detect Trojan horses when they need to use messaging or Bluetooth services. The rest of this paper is structured as follows. Related work on trojans and other malware is shown in Section 2. In Section 3 the general structure of the proposed approach to detect Trojan horses is described. In order to test the implementations, real cases have been applied in different scenarios and the corresponding monitoring results are given in Section 4. Finally, conclusions are drawn.

2. Related Work

In recent years, SMS has become a lucrative playground for attacks and frauds such as spamming, phishing and spoofing. The small size of anSMS (it has 160 characters) is the main reason why there has not been a large-scale infection yet. Nevertheless, the size of the MMS is imposed by the service provider: it usually exceeds 300KB, which seems an appropriate size to accommodate a malware. Although Bluetooth technology provides levels of security based on the identification of the devices involved, the number of vulnerabilities via Bluetooth has increased considerably. The dynamics of mobile phone malware that propagates by proximity contact is considered in [7][8] where strategies for detecting and mitigating proximity malware from a simple local detection to a globally coordinated defence are studied. In [9], combination of four detection schemes is proposed, called SMS-Watchdog, which builds normal social behaviour profiles for each SMS user and then uses them to detect SMS anomalies in an online and streaming fashion. For instance, features such as SMS sent count, RAM-free space and CPU usage, are sent to a remote server for SMS intrusion detection [10].

Other studies focus on malware and user behaviours. An agent-based malware modeling (AMM) framework is given in [11] where a representation of malware behaviours is based on the key observation that the logical ordering of an application's actions. This approach discriminates between the malicious behaviour of malware from the normal behaviour of applications by training a classifier based on Support Vector Machines (SVMs) [12]. A method is presented in [13] which observes unique behaviours of the mobile phone applications and the operating users on input and output constrained devices. Since smart

phones are used to store users' identities and sensitive information/data, it is necessary to authenticate legitimate users and to block imposters. Based on the analysis of keystroke data of diverse smart phone users, a keystroke dynamics-based PIN-verification mode is developed in [14], which ensures information security on smart phones.

Some works propose the study of other mobile features. For instance, a power-aware malware-detection framework that monitors, detects, and examines previously unknown energy-depletion threats is proposed in [15], and a malware detection method that uses power consumption to detect anomalous behaviours on cellphones, called VirusMeter, which is designed in [16]. VirusMeter applies machine learning techniques to further improve the detection accuracy of real malware, including FlexiSPY and Cabir. Another alternative is proposed in [10] where a power-aware malware-detection framework that monitors and analyzes unknown energy depletion threats is given. Nevertheless, these battery and current-based approaches depend on the quality and age of the battery since the internal resistances of these, influence the results.

The presented approach in this paper is based on analyzing system-specific events produced by Trojan horses which use message or Bluetooth services in their infection process. Today, these two infection vectors are not only the most popular ones among existing smartphone viruses, but also the most dangerous ones, because they are unique to smart-phones and have strong spreading capability. Thus, it is critical to have a security solution that can effectively combat these viruses.

3. Trojan Horse Detection through Events Monitoring

If a user sends an email or an SMS to another user then an event which announces this arrival is produced. Some similar situations happen if a user tries to establish a Bluetooth connection or send a file. In all cases, the user is informed through an alert (sometimes with sound) in the graphical interface of the phone. Obviously, one goal of malware, especially in Trojan horses, is to leave no trace, and hence, this software tries to avoid all types of alerts or other ways which can help users to detect the infection. For example, in Neo-Call spyware [3], a malicious user communicates with an infected phone through SMS messages. The user is not informed of these sent and received SMS. Furthermore, none of these messages are saved into the SMS inbox or outbox of the attacked user's phone. However, low-level events are produced and, using these events and appropriate monitoring, suspicious behaviours like this can be detected. Taking advantage of this fact, an events-based technique to detect both known and unknown Trojan horses is presented to the following.

3.1 Generic Client Design

The generic structure of the method includes two main components for the events monitoring: the User Interface and Monitoring modules (see **Fig. 1**):

The **User Interface** enables clients to observe monitoring state. It can be used to visualize established connections (used ports and protocols) and the latest sent and received SMS, MMS and files through Bluetooth. Using this interface, if the Monitoring Module detects an anomaly in the data sent/received or in a connection request, then a user can accept or decline the request. If the user accepts, then the Monitoring Module will transfer the control to the operating system to establish the connection or send the message to a specific folder.

The **Monitoring Module** is the most important since its main task is the management of connections and sent and received data using events. It is made up of small modules that

manage Bluetooth, SMS and MMS communication mechanisms. Suspicious events such as the arrival or sending of an SMS or a multimedia file through Bluetooth are studied by the Monitoring Module. This module is composed of the Connection and File submodules: the **Connection Submodule** controls protocols where a previous connection is necessary before sending or receiving a file or a message. Each connection is defined by a protocol and a specific port. Usually, malicious applications send attached infections through instant messaging or Bluetooth changing the default protocol and port to avoid notifications. In this submodule there is a configuration file that stores the default Bluetooth and messaging protocols and ports as well as the protocol and port detail of each accepted connection. When an external user tries to connect in a suspicious way, an event is produced and the Connection Submodule (through the User Interface) will show the details about the connection through a message.

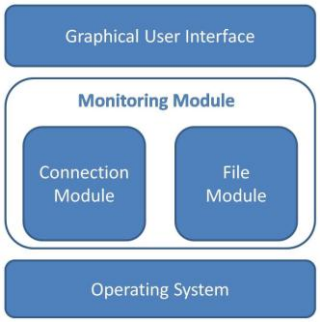

Fig. 1. Monitoring client structure.

On the other hand, the **File Submodule** controls messages and files which can be sent or received. This submodule analyzes the frequency of received files or messages and the origin (path) of the programs which send the suspicious files. The first feature can be configurable to detect the propagation of a Trojan horse which tries to spread to other devices analyzing the time between two similar events. For instance, it is not usual that a user sends a large amount of SMS in a short time because it needs a time to write the message and search the contacts before sending it. The second feature examines the installation path of an application because some malicious applications do not use the default folders to avoid being recognized. In the File Submodule there is another configuration to store the minimum time between two events, the installation path by default and the names and their folder of accepted Bluetooth and messaging applications.

Using these submodules, the Monitoring Module can detect files within which a Trojan horse infection can be packed. Hence, if a file is sent or received then an event is produced and the File Submodule will determine whether it is an infection.

A flowchart of the behaviour when a user tries to establish a connection or send a file is depicted in **Fig. 2**. Let's consider an application which implements our method is running in the background. When the message sent or the connection request arrives to the phone, an event is produced and the application starts to analyze it. Using the Monitoring Module, a connection request or a file is analyzed by the Connection Submodule or File Submodule respectively. If the Monitoring Module detects a possible infection, then a message is in real-time mode sent to the user via the User Interface module. In this message, the connection or file details (sender number, port and the path of the program) are shown. If a connection request is finally rejected, then it will not be established. If a file such as SMS or MMS is

rejected, then the message will not be sent to the inbox or outbox. And if finally a connection or a file is accepted, the Monitoring Module will hand the control over to the operating system to establish the connection or place the file in the correct folder. The decision and the details about the message or the file are saved in a text file for future messages/connections requests with the same details. Furthermore, this file can be edited and modified by the user.

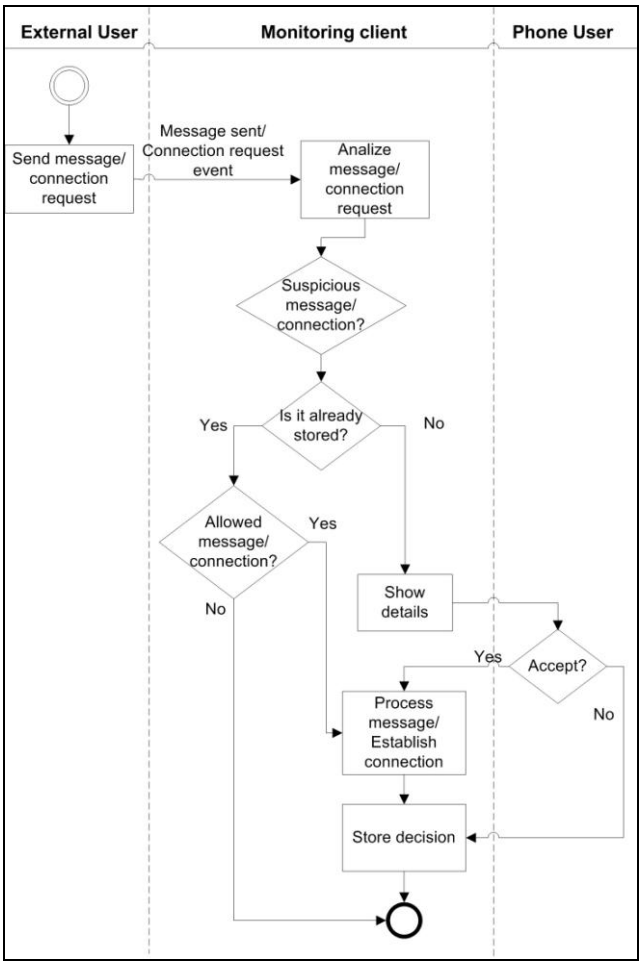

Fig. 2. Monitoring client behaviour when an external user tries to establish a connection or send a message.

3.2 The Symbian Client

The monitoring client was developed in Symbian C++ version S60 3rd with Nokia Carbide IDE. The User Interface module shows the active connections and the last 10 sent/received messages. This module admits or rejects the archives or connection requests. For reasons of program stability and to prevent interferences, the GUI runs in a thread separate from the other components. In the Symbian Monitoring Module, the Connection Submodule analyzes the port and protocol for each connection.

The main reason is that, by using sockets, hackers can transfer data to a phone while no message appears on the user screen to warn against the attack. In Symbian C++, a socket is defined by a port and a protocol. Hence, when an unusual port is used, the Monitoring Module

sends a notification to the user through the user interface. The File Submodule controls the number of messages and times them. Furthermore, it analyzes the path of the programs that send/receive messages or connection requests. For example, a massive sending of SMS could be an infection in its propagation process. Using the User Interface module, the user can configure the interval time and message limit parameters.

3.3 The Windows Mobile Client

For Windows Mobile operating system, the Monitoring client structure has been implemented using the Windows Mobile 6 Professional Software Development Kit and the Microsoft Visual Studio 2008 programming environme

The main difference in the implementation between Windows Mobile and Symbian SO is the events capture. Windows Mobile APIs provide some classes and functions to indicate when a message is sent, received or a connection is established. For example, the *MessageInterceptor* class has the *MessageReceived()* event-handling function to show when an SMS has been received. Hence, these classes are used to capture the events and implement the Connection and File Submodules. In Symbian operating system however, sockets are necessary to capture the same events.

3.4 Android Architecture Draft

The Open Handset Alliance Project Android [17] aims to develop the first complete, free, and open mobile platform. Open means that all sources have been published with the release of the first Android handset in the second half of 2008. Android is based on Linux kernel 2.6 and provides tools and APIs necessary to develop 3rd-party applications using Java. Android applications can be compiled to be executed in a virtual machine called Dalvik virtual **machine**, which is not compatible with Java SE or ME. The monitored feature set of android devices is slightly different from that of Symbian OS and Windows Mobile. Since the openness of Android allows modification of almost every software component and Linux was used as the core OS, this platform provides a good foundation for building a monitoring agent which benefits from several years of Linux security research. A simple architecture is proposed in **Fig. 3**.

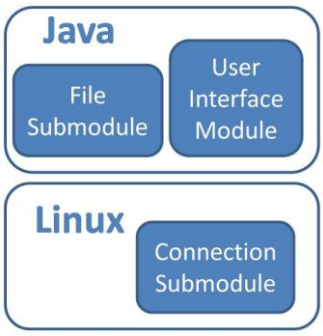

Fig. 3. Correspondence between Android OS layers and our approach.

The Monitoring Module is split and the User Interface module and the Connection Submodule are included in the Java layer because the Android Java environment provides several libraries for their implementation. Since accessing security relevant system

characteristics might be problematic using native calls from Java applications (JNI), the File Submodule is placed in the Linux OS layer.

3.5 iPhone Architecture Draft

The iPhone handheld [18] designed by Apple Inc. is based on the iPhone OS mobile operating system. The iPhone OS consists of a number of different software layers, each of which provides programming frameworks for the development of applications that run on top of the operating system:

- The **iPhone OS Cocoa Touch Layer** contains the most commonly used frameworks for iPhone application development.
- The **iPhone OS Media Layer** provides the iPhone OS with audio, video, animation and graphics capabilities.
- The **iPhone OS Core Services Layer** provides much of the foundation on which the above layers are built. It includes services such as database connection and geographical location data.
- The **iPhone OS Core OS Layer** is the closest layer to the hardware. It provides services including low-level networking, memory management, file system handling and threads.

A simple correspondence scheme between the iPhone architecture and the proposed approach is shown in **Fig. 4**.

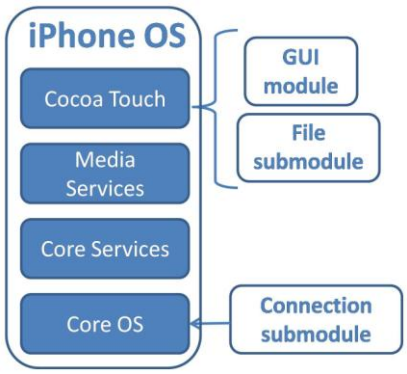

Fig. 4. Correspondence between iPhone OS and our approach

The Monitoring Module is divided into two parts. The User Interface module and the File Submodule correspond to the Cocoa Touch Layer. This layer contains the necessary frameworks to implement the events, messages and data handling and user interface creation and management. However, the Connection Submodule corresponds to the Core OS Layer because it contains the network framework to control the established connections. Specifically, the iPhone OS Security framework provides all the security interfaces for connection with external networks including certificates, public and private keys, trust policies, keychains, encryption, etc.

4. Experimentation

In this section, several examples of real mobile malware are used to evaluate the effectiveness of the detection method. First, we conduct individual test through different real malware and after it we tested it in a group of different users.

4.1 Experiment on the Detection of the Trojan-SMS.Win-CE.Sejweek Trojan

The behaviour of the "Trojan-SMS.WinCE.Sejweek" trojan for Windows Mobile has been simulated in a HTC 3300 PDA. This infection is commonly found associated with pirated software because it downloads an XML file from a website which contains the numbers of premium rate SMS numbers and the frequency at which the expensive ($1 per message) SMS messages are sent. This trojan uses the XML labels to interpret the information and send the SMS.

4.1.1 Simulation

For the experiment, an attacked user receives a MMS with a picture, which includes the infection. In this experiment, a XML file has not been used but instead the phone numbers are sent by SMS from a malicious user phone. It is thought that the trojan is installed and it can send contact information to these phone numbers through SMS messages. Furthermore, the propagation action in the infection program has been implemented. When the trojan finishes sending all the SMS messages, the picture file (with the trojan) is sent to all contacts. The attacked user has been simulated by the Visual Studio Simulator tool included in Microsoft Visual Studio 2008. When the simulation begins, the Trojan horse starts to run in the background while the user only sees the picture in his PDA. The Microsoft Tool Cellular Emulator v1.43 was also used to enable the malicious user to send and receive MMS and SMS messages and make calls (including other services) to the Visual Studio's emulator. Thus, the simulation of information exchange through MMS or SMS between mobile devices has been carried out in an efficient way without using the services of any company. The main objective of this infection is to obtain private data from the attacked phone. In this application that information will consist of the contact names and telephone numbers.

The main Windows Mobile 6 SDK classes used for the demonstration were:

- *OutlookSession*: This allows, among other functions, the access and modification of data in the Contacts. In this case, the nickname and the phone number are used.
- *MessageInterceptor:* This personalized message receiver implements the channel that allows the infection to remain pending for an incoming SMS. The purpose of the *MessageInterceptor* class will be a key factor in the implementation of the Trojan horse and the proposed solution, because it contains the event which receives SMS messages (*MessageReceived()*).
- *SmsMessage:* Implements the creation and sending of SMS.
- *MmsMessage:* Implements the creation and sending of MMS.

The behaviour on the user's attacked device is (see **Fig. 5**):

1. The attacked user receives the picture where the Trojan horse infection is packed. The file can be transferred from the Internet through MMS or via Bluetooth to the terminal.
2. Once the virus reaches the mobile phone, it automatically installs itself.
3. The malicious program awaits orders. The attacker's instructions are introduced by means of an SMS with a default structure.
4. If the received SMS is in the correct format, in this case with the heading *@spy@*, the content processing begins. Otherwise, the message goes to the users inbox.
5. The Trojan horse then checks the label-value pairs. The parser recognizes the pairs *<sms>telephone_number*.

6. The program automatically sends the Contact data in the Phonebook to each phone *<sms>telephone_number* pair which appears in the SMS received. In the demo, it was sent via SMS to each contact name and phone number with the format *contact_name:telephone_number* although other data could also be sent.
7. The Trojan program is sent by MMS to all contacts for the propagation of this malicious software. An MMS for each contact telephone number is necessary because the size of the installer is 52KB.
8. Once every SMS and MMS has been sent, the Trojan horse infection awaits for new orders.

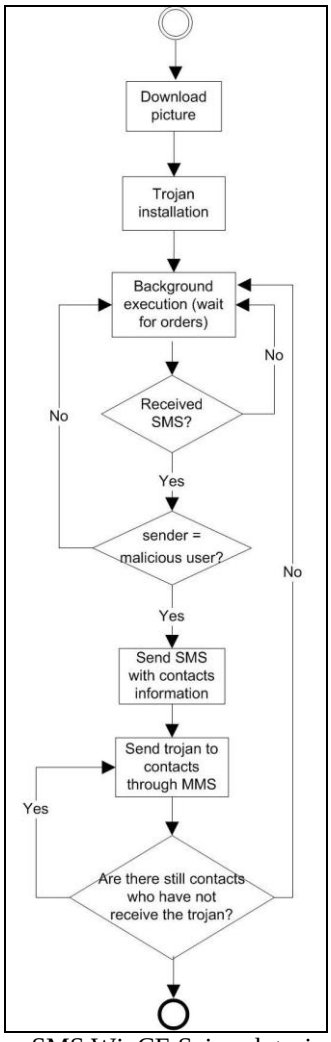

Fig. 5. Trojan-SMS.WinCE.Sejweek trojan behaviour.

The Trojan horse has been installed in the victim's mobile phone. Now, the hacker will carry out the following steps:

1. The malicious user sends an order to the Trojan horse via on SMS to the user under attack in an appropriate format, which in the test application is *@spy@<sms>telephone_number<sms>telephone_number<sms>…*
2. The attacker awaits the response of the Trojan horse.

3. The attacker begins to receive SMS messages with the structure *contact_name:telephone_number-contact_name:telephone_number-…*

The messages are processed through a second parser through which the malicious user decides how to process the information. This process can be repeated if the attacked user remains aware of the infection.

4.1.2 Detection and Results

The *MessageReceived()* event provided by Windows Mobile SDK is implemented by the *MessageInterceptor()* method to detect any access via SMS or MMS that occurs in the system. Furthermore, the *OnClick()* event helps us to identify the user interaction with the device. The analyzing procedure starts when the program sends various messages without previous click events and the difference between two sent messages is less than 3 seconds, whereby the Monitoring Module accesses the process manager to scan the path of all the processes (the number of processes running is limited to 32). Previously, a text file is saved with the information data of authorized programs to check whether the program file is a possible Trojan horse. So, if a program fails to appear in the known-programs list, then a message is displayed using the User Interface module. Obviously, the whole process takes place without the user being aware of the attack because the Trojan horse remains running in the background. In addition, no messages sent from the device to the attacked phone enter the mailboxes and hence no suspicion is detected.

Sending SMS and MMS messages entails an economic cost and, for a possible estimation, it is assumed that the average length of the contact name or nickname is 10 characters. Usually, mobile phone numbers have nine characters and, if a space is added, then this information consumes twelve characters. Let us suppose that an average user has 100 contacts stored in his phone book, and that, in Spain, an SMS costs 0.20€ on average. Thus, adding two separating characters per contact, 2.62€ will be the cost in SMS if a malicious user requests data from the contacts in the agenda of the attacked device. Finally, if the cost of one MMS (1€ approximately in Spain) is taken into account, then the trojan propagation (the infection) consequences could cost about 102,62€. It is important to note that the malicious user can begin this process at any time. Finally, for the detection process, it is used several antivirus mobile programs such as AirScanner 3.0 [19] and BullGuard 2.0 [20]. These experiments show that none of these solutions are capable of detection the Trojan horse implementation. However, the proposed solution detected the Trojan horse in every single test. With regard to time cost, due to the detection method is based on event detection, the total spent time until the infection is detected, is related to the frequency of events generated by the Trojan horse, the used protocols and the attacked operating system. If a Trojan horse generates a small number of events or it generates like a human through typical events interaction (for example, onClick() before MessageSent() in case of a SMS in Windows Mobile) and using defaults ports and protocols our tool could not detect it (but notifications will be visible for the user).

4.1.3 Symbian Variant

The Trojan-SMS.WinCE.Sejweek trojan has been successfully implemented for Symbian SO using a Nokia 6120 Classic and Net60 library [21]. This library enables the adaptation of our Windows Mobile code from C# to C++. Therefore, the code of the first experiment has been reused. However, Symbian SO fails to contain any event listener for SMS or MMS. For this reason, event handling functions have been simulated using sockets, SMS and MMS protocols and default ports.

4.2 Experiment on the Detection of the Neo-Call Spy Software

Neo-Call is a spyware program similar to other spy software such as FlexiSPY or SpyPhone [22] that runs on most Symbian handhelds. Neo-Call software was installed in a Nokia 6120 Classic. It conducts eavesdropping, call interception, GPS tracking, etc, and monitors phone calls and SMS text messages. The malicious user receives the spy information through SMS to a phone and can control spy actions by sending new SMS to the attacked phone using the Neo-Control tool. This tool creates and sends formatted SMS automatically with specific codes and tags.

The SMS and Bluetooth misbehaviours have been tested. In both situations, the connections to send and receive information through SMS are established through sockets. In the proposed method, the Monitoring Module controls the port range for SMS protocol (KSMSDatagramProtocol [23]). Therefore, using the User Interface Module, in all tests carried out, the user was informed when Neo-call tried to send/receive an SMS to/from the malicious user.

4.3 Experiment on the Detection of Cabir

Cabir is the first network worm capable of spreading via Bluetooth; it infects mobile phones which run on Symbian OS. Bluetooth devices are sought by Cabir and once found, a file is transferred to them. In 2008, the Cabir code was published on the web [24]. Like other Symbian malware, Cabir uses a socket with Bluetooth RFCOMM protocol and 0x00000009 port. Usually, port 0 is the normal port used to transfer files with the UI in Symbian OS. However, Cabir uses another port to prevent notification.

An experiment was carried out to discover when this malware tries to connect with other devices, but not when the file was transferred. In the proposed approach, the Connection Submodule includes the RFCOMM protocol and a wide port range. Hence, the attacked device is informed of the connection attempts through a message on the screen.

4.4 Experimentation with real users using Neo-Call

To test the proposed Trojan horse detection tool, a mobile phone with Symbian SO was given to fifteen users during five days. The set of users were composed of five computer science colleagues, five students of computer science and five relatives without experience in computer science. The malicious software Neo-Call and an application which implemented our detection method were installed in each phone. Another malicious user sent various SMS to each phone to use the Neo-Call to spy the phones and, consequently, test the method. We observed that phone skills and expertise using a mobile phone is critical. Although computer science researchers and students can distinguish the suspicious activity, users with no experience used to include all or none of the malicious activity on it. That is evidence that, despite the detection method always shows an alert message when a SMS was sending, it is necessary for the users of a previous knowledge about the most common Bluetooth and messaging applications to decide when a suspicious application can attack the phone. To avoid this misbehavior, we are working on the improvement of the User Interface module to help users without experience to make decisions.

5. Conclusion

We conclude this paper by discussing the limitations encountered in pursuit of this work, along with concluding remarks.

5.1 Limitations

Our method can be implemented on various mobile operating systems. However, mobile frameworks provide different tools for event handling and our detection approach must be adapted. Furthermore, only a few Trojan horse samples are publicly available for research. Testing detection systems with real-world malware is necessary for the evaluation of the effectiveness of any detection system, however due to the nature of in-the-wild malware activities, effective and comprehensive evaluation and testing with real-world Trojan horse samples are difficult to carry out. Moreover, during legitimate events, such as a natural disaster, false positive may emerge in our system. These forms of unusual events will be difficult to distinguish from virus outbreak based only on the connection or message details. To address this issue, a user survey system can be employed to query a subset of the suspected user on whether the suspicious activity is intentional. If users confirm the suspicious activity then the method will refrain from reaction.

5.2 Concluding Remarks

The attacks on mobile handhelds, especially intrusions made by Trojan horses have been studied. Based on the analysis of the operation mechanism of Trojan horses, an effective method to detect these infections in messaging and Bluetooth services has been presented. The proposed method has two modules, the Graphical User module to notify about possible infections and the Monitoring module to control the connection requests and analyze the sent/received files. Using real-world malware such as Cabir and Neo-Call, we experimentally show that our method can effectively and efficiently detect their existence.

References

[1] Physorg, "Five Billion People to Use Mobile Phones in 2010: UN," 2010. http://www.physorg.com/news185467439.html.
[2] IBM X Force Threat Reports, "IBM Internet Security Systems X-Force, Trend and Risk Report," 2009. http://www-935.ibm.com/services/us/iss/xforce/trendreports.
[3] Neo-Call. http://www.neo-call.com.
[4] C. Fleizach, M. Liljenstam, P. Johansson, G.M. Voelker, A. Mehes, "Can You Infect Me Now?: Malware Propagation in Mobile Phone Networks," in *Proc. of the 2007 ACM Workshop on Recurring Malcode (WORM '07)*, pp. 61–68, Nov. 2007. Article (CrossRef Link)
[5] Cabir. http://www.f-secure.com/v-descs/cabir.shtml
[6] Commwarrior. http://www.f-secure.com/v-descs/commwarrior.shtml
[7] G. Zyba, G.M. Voelker, M. Liljenstam, A. Méhes, P. Johansson, "Defending Mobile Phones from Proximity Malware," in *Proc. of INFOCOM 2009*, pp. 1503-1511, Apr. 2009. Article (CrossRef Link)
[8] J. Cheng, H.Y. Wong, H. Yang, S. Lu, "SmartSiren: Virus Detection and Alert for Smartphones," in *Proc. of the 2007 International Conference on Mobile Systems, Applications, and Services (Mobysis '07)*, pp. 258-271, June 2007. Article (CrossRef Link)
[9] G. Yan, S. Eidenbenz, E. Galli, "SMS-Watchdog: Profiling Social Behaviors of SMS Users for Anomaly Detection," Lecture Notes in Computer Science, vol. 5758, pp. 202-223, 2009. Article (CrossRef Link)
[10] A.D. Schmidt, F. Peters, F. Lamour, S. Albayrak, "Monitoring Smartphones for Anomaly Detection," in *Proc. of Mobilware 2008*, pp. 92-96, June 2007. Article (CrossRef Link)
[11] A. Bose, K. Shin, "On Mobile Viruses Exploiting Messaging and Bluetooth Services," in *Proc. of International Conference on Security and Privacy in Communication Networks (SecureComm'06)*, pp. 1-10. May 2007. Article (CrossRef Link)
[12] A. Bose, X. Hu, K.G. Shin, T. Park, "Behavioral Detection of Malware on Mobile Handsets," in

Proc. of the 6th International Conference on Mobile Systems, Applications, and Services, pp. 225-238. June 2008. Article (CrossRef Link)
[13] L. Xie, X. Zhang, J. Seifert, S. Zhu, "pBMDS: a Behavior-based Malware Detection System for Cellphone Devices," in *Proc. of the WISEC 2010*. pp. 37-48, Mar. 2010. Article (CrossRef Link)
[14] S. Zahid, M. Shahzad, S.A. Khayam, M. Farooq, "Keystroke-Based User Identification on Smart Phones," in *Proc. of the RAID 2009*. pp. 224-243, Sep. 2009. Article (CrossRef Link)
[15] H. Kim, J. Smith, K.G. Shin, "Detecting Energy-Greedy Anomalies and Mobile Malware Variants," in *Proc. of the 5th International Conference on Mobile Systems, Applications, and Services*, pp. 17-20, June 2008. Article (CrossRef Link)
[16] L. Liu, G. Yan, X. Zhang, S. Chen, "VirusMeter: Preventing Your Cellphone from Spies," in *Proc. of the RAID 2009*, pp. 244-264, Sep. 2009, Article (CrossRef Link)
[17] Open Handset Alliance Project Android. http://www.android.com
[18] Apple iPhone. http://www.apple.com/iphone
[19] AirScanner Mobile Software. http://www.airscanner.com
[20] BullGuard Antivirus. http://www.spyphone.es
[21] RedFiveLabs. http://www.redfivelabs.com
[22] Spyphone. http://www.spyphone.es
[23] S60 5th Edition C++ Developer's Library v2.1. http://library.forum.nokia.com
[24] Cabir code. http://www.offensivecomputing.net/?q=node/773

Juan Antonio Ortega Ramírez obtained the Ph.D. degree in Computer Science in 2000 at the Seville University in Spain. He is a Lecturer since 1992 in the Department of Languages and Computer Systems at the Seville University. His research interests are: the mobile solutions, temporal series and the global information systems and specifically the domotic and assistencial systems. He is Director of the Scientific Computing Center of Andalusia since 2005.

Daniel Fuentes Brenes received his bachelor degree in Computer Science in 2008 and works in the Scientific Computing Center of Andalusia since 2008. His main research interests are security in mobile devices, information systems and energy-efficiency in wireless sensor networks.

Juan Antonio Álvarez García received his bachelor degree in Computer Science in 2003 and his Ph.D. degree in Computer Science in 2010 from the University of Seville. He is a Lecturer in the Department of Computer Languages and Systems at the Seville University. His main research interests are ubiquitous, mobile and urban computing. He is also interested in healthcare systems.

Luis González Abril. is a Lecturer in the Dep. Of Applied Economy I at the University of Seville (Spain). He obtained his graduate in Mathematics in 1986 and his Ph. D. degree in Economy in 2002 from the University of Seville. His main researchs are: Similarities, Support vector machines, machine learning and Information Systems.

Francisco Velasco Morente is a Lecturer in the Dep. Of Applied Economy I at the University of Seville (Spain). He obtained his graduate in Mathematics in 1979 from the University of Sevilla and his Ph. D. degree in Mathematics in 1991 from the University of Seville. His researchs are: similarities, Support vector machine, bifurcations of continuous and discrete dynamical systems and optimal control problems, both applied to economic problems