Sistemas & Telemática
ISSN: 1692-5238
EditorSyT@icesi.edu.co
Universidad ICESI
Colombia

Londoño, Sebastián; Urcuqui, Christian Camilo; Navarro Cadavid, Andrés; Fuentes Amaya, Manuel; Gómez, Johan
SafeCandy: System for security, analysis and validation in Android
Sistemas & Telemática, vol. 13, núm. 35, 2015, pp. 89-102
Universidad ICESI
Cali, Colombia

Original Research / Articulo Original - Tipo 1

SafeCandy: System for security, analysis and validation in Android

Sebastián Londoño, MSc. / slondono@icesi.edu.co

Christian Camilo Urcuqui / ulcamilo@gmail.com

Andrés Navarro Cadavid, Ph.D. / anavarro@icesi.edu.co

Universidad Icesi, Cali-Colombia

Manuel Fuentes Amaya / manuel.fuentes@password.com.co

Johan Gómez / johan.gomez@password.com.co

Password Consulting Services, Bogotá-Colombia

ABSTRACT Android is an operating system which currently has over one billion active users for all their mobile devices, a market impact that is influencing an increase in the amount of information that can be obtained from different users, facts that have motivated the development of malicious software by cybercriminals. To solve the problems caused by malware, Android implements a different architecture and security controls, such as a unique user ID (UID) for each application, while an API permits its distribution platform, Google Play applications. It has been shown that there are ways to violate that protection, so the developer community has been developing alternatives aimed at improving the level of safety. This paper presents: the latest information on the various trends and security solutions for Android, and SafeCandy, an app proposed as a new system for analysis, validation and configuration of Android applications that implements static and dynamic analysis with improved ASEF. Finally, a study is included to evaluate the effectiveness in threat detection of different malware antivirus software for Android.

KEYWORDS Mobile security; Android security; ASEF; anti-malware.

SafeCandy: un sistema para seguridad, análisis y validación en Android

RESUMEN Android es un sistema operativo para dispositivos móviles con más de un billón de usuarios activos. Su creciente peso en el mercado y la cantidad de información que, gracias a ello, puede ser obtenida de diferentes usuarios, ha motivado el desarrollo de software malicioso por parte de cibercriminales. Para resolver los problemas causados por el malware, Android implementa una arquitectura diferente y controles de seguridad, como un ID único de usuario (UID – Unique User ID) para cada aplicación, mientras que un API permite la distribución en la plataforma de aplicaciones Google Play. Sin embargo, está demostrado que hay formas de violar esta protección, por lo que la comunidad de desarrolladores viene implementando alternativas dirigidas a mejorar los niveles de seguridad. Este artículo presenta: la más reciente información sobre tendencias y soluciones de seguridad para Android; SafeCandy, un nuevo sistema para el análisis, la validación y configuración de aplicaciones Android, el cual implementa análisis estáticos y dinámicos y un ASEF [*Android Security Evaluation Framework*] mejorado; y la evaluación de efectividad en la detección de amenazas por parte de diferentes antivirus para *malware* en Android, incluido SafeCandy.

PALABRAS CLAVE Seguridad móvil; seguridad en Android; ASEF; anti-malware.

SafeCandy: Sistema de segurança, análise e validação em Android

RESUMO O Android é um sistema operacional que conta atualmente com mais de um bilhão de usuários ativos para todos os seus dispositivos móveis, um impacto no mercado que está influenciando um aumento na quantidade de informação que pode ser obtida a partir de diferentes usuários, fatos que têm motivado o desenvolvimento de software malicioso por parte de cibercriminosos. Para resolver os problemas causados por malware, Android implementa uma diferete arquitetura e controles de segurança, como um ID de usuário exclusivo (UID) para cada aplicação, enquanto uma API permite na sua plataforma de distribuição, aplicativos do Google Play. Tem sido demonstrado que existem maneiras de violar essa proteção, de modo que a comunidade de desenvolvedores tem vindo a desenvolver alternativas destinadas a melhorar o nível de segurança. Este artigo apresenta informações sobre as últimas tendências e soluções de segurança para Android, e propõe também o SafeCandy como um novo sistema para análise, validação e configuração de aplicativos Android que implementa a análise estática e dinâmica, com uma ASEF melhorada. Finalmente, se inclui um estudo para avaliar a eficácia na detecção de ameaças de diferentes software antivírus malware para Android.

PALAVRAS-CHAVE Segurança móvel; segurança Android; ASEF; anti-malware.

Received / Recepción: Octubre 1, 2015 - Accepted / Aceptación: Noviembre 22, 2015 *doi: 10.18046/syt.v13i35.2154*

I. Introduction

Android is an open source operating system for mobile devices with currently more than a billion active users (Pichai, 2014). This OS is based on a Linux kernel (**Figure 1**), and for that reason implements various security controls and is divided into two layers, the first referring to security at the operating system through the Sandbox application and a second API that permits the application level permission (Smalley & Craig, 2013) (**Figure 2**). Additionally, Google has its own digital distribution platform, Google Play applications, where other processes that aim to limit the spread of malicious code are implemented. From the first appearance of Android malware, it has been shown that this OS and its security processes have problems with the protection of user information. Additionally, it has been found that these malicious codes are designed to disrupt the proper functioning of the device or to obtain user data such as contacts, text messages, social networks, bank accounts, etc. During 2014, several cases of malware and interesting Android vulnerability studies were shown, including the following:

Dendroid Toolkit allows the customization of APK to generate malicious files (Khandelwal, 2014a), Ransomware locks the victim's device until a payment is made for a key to unlock Android (Khandelwal, 2014b). MisoSMS captured text messages and encrypted them with XTEA in order to reduce their level of detection in security systems (Dhar-

I. Introducción

Android es un sistema operativo de código abierto para dispositivos móviles con más de un billón de usuarios activos (Pichai, 2014). Está basado en un *kernel* de Linux (**Figura 1**), implementa varios controles de seguridad y está dividido en dos capas, la primera referente a la seguridad en el sistema operativo a través de la aplicación *Sandbox*, la segunda, el API que permite los permisos a nivel de aplicación (Smalley & Craig, 2013), como ilustra la **Figura 2**. Por su parte, Google Play, la plataforma de distribución digital de aplicaciones de Google implementa otros procesos con el objetivo de limitar la propagación de código malicioso [*malware*]. Sin embargo, desde la primera aparición de *malware* en los dispositivos Android, se ha demostrado que este sistema operativo [OS, *Operating System*] y sus procesos de seguridad tienen problemas con la protección de la información de usuario y se ha encontrado que estos códigos maliciosos están diseñados para perturbar el correcto funcionamiento del dispositivo o para obtener datos del usuario, tales como contactos, mensajes de texto, redes sociales, cuentas bancarias, etc.

Durante 2014 se presentaron varios casos de *malware* y algunos estudios interesantes sobre la vulnerabilidad en Android, algunos de los cuales se describen a continuación:

Dendroid Toolkit permite personalizar la APK para generar archivos maliciosos (Khandelwal, 2014a); Ransomware bloquea el dispositivo de la víctima, hasta que se realiza un pago y así obtiene una clave para desbloquear Android (Khandelwal, 2014a); MisoSMS captura mensajes de texto y los encripta con XTEA, con el objetivo de reducir su nivel de detección en sistemas de seguridad (Dharmdasani & Pidathala, 2014). Asimismo, se han encontrado importantes

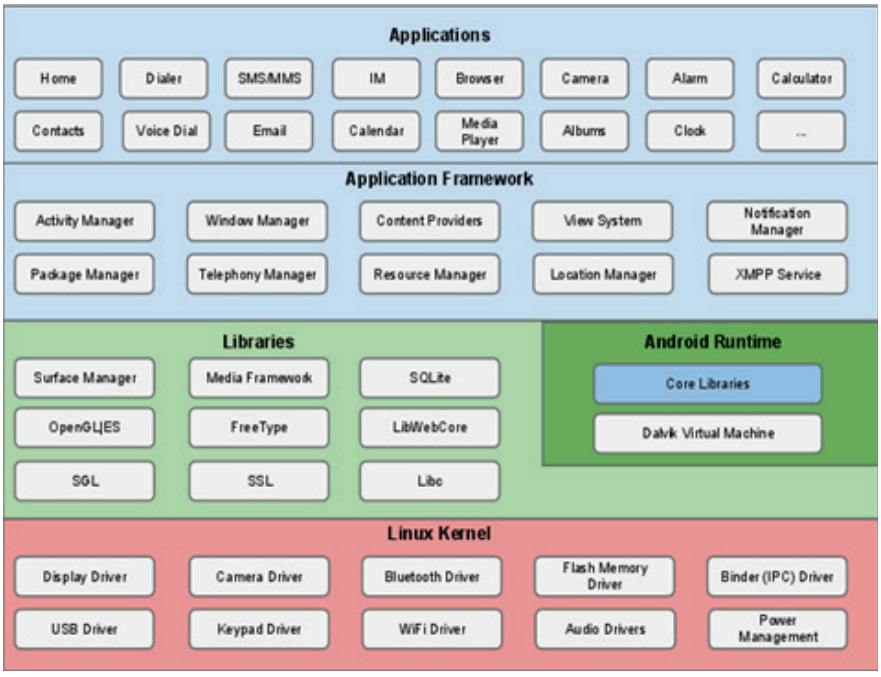

Figure 1. Android Architecture (Oja, Kaski, & Kohonen, 2003) / Arquitectura Android (Oja, Kaski, & Kohonen , 2003)

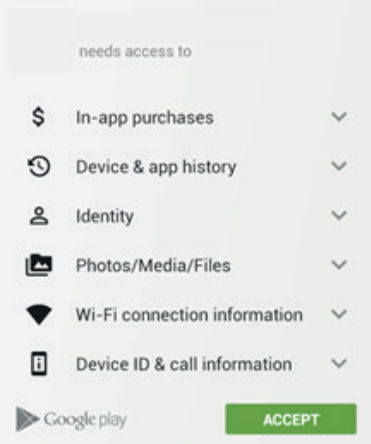

Figure 2. List of permissions for an application (image from Google Play) / Lista de permisos para una aplicación (imagen obtenida del portal de Google Play)

vulnerabilidades en el OS, como Android Fake ID, en el cual se describe como copiar el identificador único de una aplicación (Forristal, 2014), el incorrecto uso de la comunicación inter-componente y las vulnerabilidades *Sidewinder Targeted Attack* explotadas en las librerías de anuncios (Wei et al., 2014). Como se mencionó, hay *malware* desarrollado y pruebas de vulnerabilidad que muestran que este OS presenta problemas de seguridad que aún necesitan resolverse, lo que ha motivado el trabajo de la comunidad de desarrolladores en la búsqueda de alternativas para mejorar los niveles de seguridad.

Para resolver estos problemas causados por los ataques de *malware* se han propuesto algunas técnicas: evaluación de análisis estáticos, análisis dinámicos, aprendizaje de máquina; *frameworks* para dispositivos móviles con Android (*Android Security Evaluation Framework* [ASEF], *Android Static Analysis Framework* [SAAF]), *Wall-Droid*; sistemas de seguridad, como MobSafe que mejora el análisis y la detección de amenazas desde plataformas de computación en la nube basadas en tecnologías de minería de datos; y soluciones con sandboxes, tales como AASandbox, que implementa análisis estático y dinámico sobre ambientes aislados. Sin embargo, aún los sistemas son incapaces de proveer un nivel razonable de seguridad frente a los problemas presentados por el *malware*, mientras las tendencias sugieren un camino hacia los sistemas inteligentes que integren varias de las ventajas que poseen las anteriores soluciones. Este artículo presenta SafeCandy, un sistema que adapta las ventajas de las soluciones encontradas en el estudio, con mejoras en el *framework* ASEF.

SafeCandy es un sistema con arquitectura cliente-servidor para el análisis, la validación y la configuración de seguridad para aplicaciones Android. A través de una modificación al *framework* ASEF, el sistema implementa análisis estático y análisis dinámico de ambientes arquitecturales [AVD] con el fin de cubrir las brechas presentes en cada uno de ellos. Adicionalmente la solución implementa una arquitectura en la nube que conecta varias bases de datos

mdasani & Pidathala, 2014). Also, different vulnerabilities have been found in the OS, like Android Fake ID, which describes how to copy the unique identity of an application (Forristal, 2014), the incorrect use of the inter-component communication and *Sidewinder Targeted Attack* vulnerabilities exploited in ad libraries (Wei et al., 2014). As mentioned, there are malware developments and vulnerability tests that show that this OS currently presents security problems that still need to be solved, a fact that has motivated the community of developers working on alternatives to improve those levels of security.

To solve the problems caused by malware attacks, some techniques have been proposed –static analysis assessment, dynamic analysis, machine learning, frameworks for mobile devices with Android (Android Security Evaluation Framework [ASEF], Android Static Analysis Framework [SAAF]), Wall-Droid–, security systems, as MobSafe that performs the analysis and threat detection from a cloud computing platform based on data mining technology, and solutions with sandboxes, such as AASandbox, which implements static analysis and dynamic analysis on isolated environments. However, there is still no system capable of providing a reasonable level of security against the problems presented by malware; the trends suggest a pathway toward intelligent systems that integrate many of the advantages of the above solutions. To address the problems identified in the paper, we present a new solution named SafeCandy, a system that adapts the advantages of the solutions found in the study with improvements in the ASEF framework.

SafeCandy is a client-server architecture system for analysis, validation and security settings for Android applications. Through a modification of the ASEF, the system implements both static and dynamic analysis of AVDs' environments architecture to cover the gaps presented in each, which will be presented later in the article. Additionally, the solution implements a cloud architecture that connects various malware databases, for which decision-making is improved. SafeCandy implements the various advantages found in the study performed and demonstrates from an experiment that our system is able to provide a significant level of security against known solutions in the security market for Android apps.

II. State of the art

A. Static analysis

Static analysis (Source Code Analysis) is a technique to analyze malicious features in source code, data, or bina-

ries without directly executing the application (Android Open Source Project, s/f). This technique commonly includes the use of tools to get the source code of the application to evaluate the functions performed by the malware, hard-coded URIs, credentials, keys, analyses of application logic and flow, and to identify what level of threat they possess (Batyuk et al., 2011).

Through a novel methodology called SOM (Self-Organizing Map), a visualization in 2D is able to identify the correlation between the licenses provided and then to determine irregular behavior (Drake et al., 2014).

DroidMOSS is a system that uses fuzzy logic techniques to efficiently locate and detect changes in repackaging applications (Zhou, Zhou, Jiang, & Ning, 2012).

Stowaway is a tool that detects over-privileged applications of Android. With this, different applications in different marketplaces were evaluated and it was concluded that more than one third of the published applications are over-privileged (Au, Zhou, Huang, & Lie, 2012). The results reveal that most developers are trying to follow the principles of low-privilege but fail for lack of reliable information about permissions.

Other implementations are tools for specific analysis, to build static dependency graphs with calls to internal procedures connectivity. The developments capture the relationship of consumption information from applications, through the identification of the paths of information obtained by the inputs and methods of access points to critical system services.

Airmid uses cooperation between in-network sensors and smart devices to identify the provenance of malicious traffic (Nadji, Giffin, & Traynor, 2011). The tool has a remote repair option that validates a detection system located on an external server malware.

Static analysis allows in-depth evaluation of whether an application is malicious from reverse engineering and evaluation of each of their files. The complexity of these methods has increased, due to the experience gained by malware developers over time and, as mentioned, their developments tend to have code or behaviors that hinder the assessment process.

B. Dynamic analysis

These methods study the behavior of the malware by simulating gestures while the application is running, and the user interface, running processes, network connections, and open sockets are analyzed. The objective of

de *malware*, con lo que se mejora la toma de decisiones. SafeCandy integra varias soluciones encontradas en el estudio realizado y demuestra, desde un experimento, que es capaz de proveer un nivel significativamente mayor de seguridad frente a otras soluciones en el mercado de aplicaciones Android.

II. Estado del arte

A. Análisis estático

El análisis estático (análisis de código fuente) es una técnica para analizar características maliciosas en el código fuente, los datos o los archivos binarios, sin ejecutar directamente la aplicación (Android Open Source Project, s/f). Esta técnica incluye comúnmente el uso de herramientas para obtener el código fuente de la aplicación; evaluar las funciones realizadas por el *malware*, hard-coded URIs, credenciales, llaves, análisis del flujo y lógica de aplicación; y para identificar qué nivel de amenaza poseen estas aplicaciones (Batyuk et al., 2011).

Usando una nueva metodología llamada SOM [*Self-Organizing Map*) se puede obtener una visualización en 2D para identificar la correlación entre las licencias provistas (Drake et al., 2014) y así determinar comportamientos irregulares.

DroidMOSS es un sistema que usa técnicas de lógica difusa para localizar y detectar cambios en aplicaciones reempaquetadas eficientemente (Zhou, Zhou, Jiang, & Ning, 2012). Stowaway es una herramienta que detecta aplicaciones sobre privilegiadas (*over-privileged*) de Android. Con este fueron evaluadas diferentes aplicaciones en diversos mercados, gracias a lo cual se concluyó que más de un tercio de las aplicaciones publicadas están sobre privilegiadas (Au, Zhou, Huang, & Lie, 2012). El resultado revela que la mayoría de desarrolladores está intentando seguir los principios de privilegios bajos (*low-privilege*), pero falla por la ausencia de información confiable acerca de permisos.

Otras implementaciones son herramientas para análisis específico para construir gráficas de dependencia estática con llamadas a la conectividad de los procedimientos internos. Los desarrollos capturan la relación de información que es consumida desde las aplicaciones a través de identificación de trayectorias de información obtenida por las entradas y los métodos de puntos de acceso a servicios críticos del sistema.

Airmid usa la cooperación entre sensores internos de la red y dispositivos inteligentes para identificar la procedencia de tráfico malicioso (Nadji, Giffin, & Traynor, 2011) y cuenta con una opción de reparación remota que valida un sistema de detección localizado en un servidor externo de *malware*.

Los análisis estáticos permiten evaluar a profundidad si una aplicación es maliciosa, haciendo ingeniería inversa y también evaluar cada uno de sus archivos. La complejidad de estos métodos ha crecido debido a la experiencia obtenida por los desarrolladores de *malware* en el tiempo y,

como se mencionó, sus desarrollos tienden a tener código o comportamientos que impiden los procesos de evaluación.

B. Análisis dinámicos

Estos métodos estudian el comportamiento del *malware* mediante la simulación de gestos, analizando los procesos que están corriendo, las conexiones de red y los sockets abiertos mientras la aplicación y la interfaz de usuario se están ejecutando. El objetivo de este análisis es generar gestos que evidencien un comportamiento malicioso (Enck, Octeau, McDaniel, & Chaudhuri, 2011).

Paranoid Android es un sistema que permite a los investigadores realizar un análisis amplio de aplicaciones maliciosas en la nube, usando máquinas virtuales de dispositivos móviles (Portokalidis, Homburg, Anagnostakis, & Bos, 2010); TaintDroid notifica al usuario sobre aplicaciones que desean acceder a permisos que están en la lista negra; usando un decompilador como Dalvik, el código fuente java es analizado a través del archivo de instalación (Enck et al., 2014); Crowdroid es una propuesta que busca llamadas con parámetros como open(), read(), access(), chmod() and chown(),usadas con frecuencia por aplicaciones *malware* (Burguera, Zurutuza, & Nadjm-Tehrani, 2011); y SAAF provee un análisis del flujo de datos en la aplicación y un control del flujo gráfico (Yadav & Shivamurthy, 2013).

Los métodos basados en análisis dinámico proveen información adicional acerca del comportamiento de una aplicación en comparación con los métodos de análisis estático, pero hay técnicas para evaluar el proceso realizado por el análisis dinámico (Petsas, Voyatzis, Athanasopoulos, Polychronakis, & Ioannidis, 2014).

C. Sandbox

Un *sandbox* puede ser definido como "un ambiente el cual las acciones de los procesos son restringidas a un acuerdo sobre una política de seguridad" (Bishop, 2002). En la práctica, esto significa que el *sandbox* es una instancia del sistema operativo a evaluar que es aislada de tal forma que se previene la ejecución de acciones dañinas por parte de software malicioso. Además, hay otras opciones de monitoreo, una de las cuales involucra el registro de todas las actividades del sistema mientras las aplicaciones están corriendo. Este enfoque es útil para el monitoreo y clasificación de software desconocido, dependiendo del estado anormal de sus causas (Mutz, Robertson, Vigna, & Kemmerer, 2007; Gopan & Reps, 2007).

D. Métodos de aprendizaje de máquina

Las técnicas de aprendizaje de máquina (*machine learning*) pueden predecir o describir un conjunto de datos a través de un algoritmo de aprendizaje. Para la seguridad de las aplicaciones Android, algunos métodos tienen como objetivo, aprender de una aplicación sus diferentes comportamientos a través del estudio de las llamadas a la API, usando comandos y permisos.

Algunos estudios han empleado el uso de técnicas de aprendizaje supervisado, así como el uso paralelo de diferentes algoritmos: Decision Tree, Simple Logistic, Naïve Bayes, PART, and Ridor (Yerima, Sezer, & Muttik, 2014). Además, hay trabajos que usan algoritmos basados en redes neuronales (Ghorbanzadeh, Chen, Ma, Clancy, & McGwier, 2013).

this analysis is to generate gestures to detect malicious behavior (Enck, Octeau, McDaniel, & Chaudhuri, 2011).

Paranoid Android is a system where researchers can perform a comprehensive analysis of malicious applications in the cloud using virtual machines of mobile phones (Portokalidis, Homburg, Anagnostakis, & Bos, 2010); TaintDroid notifies the user about applications that want access permissions that are blacklisted. Using a decompiler like Dalvik, the Java source code is analyzed through the installation file (Enck et al., 2014); Crowdroid is a proposal that seeks called-with parameters like open(), read(), access(), chmod() and chown(), which are the calls most often used by malware applications (Burguera, Zurutuza, & Nadjm-Tehrani, 2011); and SAAF provides an analysis of the data flow in the application and a control flow graph (Yadav & Shivamurthy, 2013).

Methods based on dynamic analysis provide additional information about the behavior of an application compared with static analysis methods, but there are techniques to evade the process performed by the dynamic analysis (Petsas, Voyatzis, Athanasopoulos, Polychronakis, & Ioannidis, 2014).

C. Sandbox

A sandbox can be defined as "an environment in which shares of the processes are restricted to an agreement on a security policy" (Bishop, 2002). In practice, this means that the sandbox is an instance operating system to assess that is isolated in a way that prevents malicious software running harmful actions. On the other hand, there are other monitoring options, one of which involves recording all the system activities while the application is running. This approach is useful to monitor unknown software and then classify it depending on abnormal states and their causes (Mutz, Robertson, Vigna, & Kemmerer, 2007; Gopan & Reps, 2007).

D. Methods of machine learning

The machine learning techniques can predict or describe a set of data through a learning algorithm. For the safety of Android applications, some methods have as an objective to learn from an application its different behaviors through the study of API calls, using commands and permissions.

Some studies have employed the use of supervised learning techniques such as parallel use of different algorithms: Decision Tree, Simple Logistic, Naïve Bayes, PART, and Ridor (Yerima, Sezer, & Muttik, 2014). Also, there is works which used algorithms based on neural networks (Ghorbanzadeh, Chen, Ma, Clancy, & McGwier, 2013).

III. SafeCandy

SafeCandy is a system with client-server architecture in the cloud for analysis, validation and security settings for Android applications. The server provides malware protection services to all types of Android applications through a modification of ASEF and the implementation of both the static and dynamic analysis AVD environments (Fuentes & Gómez, 2014; Navarro, Londoño, Urcuqui, Fuentes, & Gomez, 2014). The services provided by the server are performed remotely by an Android application (**Figure 3**), which allows interaction with the user and also provides an analysis of the device locally. SafeCandy has a central database that stores the various analyses and external databases that provide information. The Android malware system has the following characteristics:

- protection against malware applications that can access private information, spies location, premium subscription services through SMS without authorization, etc.;
- tests are performed by cloud services and locally using a database on the device;
- vulnerability detection of the scanned application and preparing reports from the data collected;
- testing applications on Android Virtual Devices [AVD];
- running scans from scheduled events that aim to improve the user experience without altering the performance of the device (e.g., when there is an active Wi-Fi connection);
- using external databases on malware applications

ferentes algoritmos: Decision Tree, Simple Logistic, Naïve Bayes, PART and Ridor (Yerima, Sezer, & Muttik, 2014). También hay trabajo que muestran el uso de algoritmos basados en redes neuronales (Ghorbanzadeh, Chen, Ma, Clancy, & McGwier, 2013).

III. SafeCandy

SafeCandy es un sistema con arquitectura cliente-servidor en la nube para el análisis, la validación y la configuración de seguridad para aplicaciones Android. El servidor provee servicios de protección contra *malware* para todos los tipos de aplicaciones Android a través de una modificación de ASEF y la implementación de análisis –tanto dinámico, como estático– en ambientes AVD (Fuentes & Gómez, 2014; Navarro, Londoño, Urcuqui, Fuentes, & Gomez, 2014). Los servicios provistos por el servidor son consultados de manera remota por una aplicación Android (**Figura 3**), lo que permite la interacción con el usuario y provee un análisis del dispositivo localmente. SafeCandy tiene una base de datos central que almacena los diversos análisis y una base de datos externa que provee información.

A continuación se presentan las características generales del sistema, con detalle del cliente, el servidor, las herramientas y las ventajas comparativas.

El sistema, en general:

- ofrece protección contra aplicaciones *malware* que pueden acceder a información privada, espías de localización, suscripción a servicios SMS *premium* sin autorización, etc.;
- realiza pruebas a través de servicios, en la nube y localmente, usando una base de datos en el dispositivo;
- permite la detección de vulnerabilidad de las aplicaciones escaneadas y prepara reportes desde los datos recolectados;

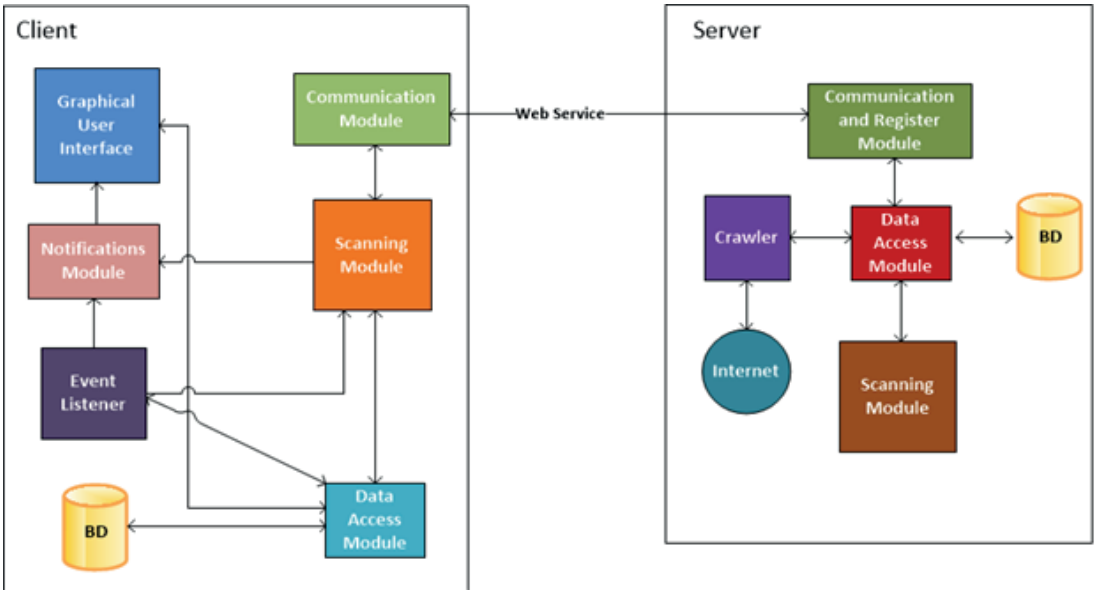

Figure 3. SafeCandy Architecture / Arquitectura SafeCandy

- realiza prueba de aplicaciones en dispositivos virtuales Android [AVD];
- corre escaneos desde eventos programados que pretenden mejorar la experiencia de usuario sin alterar el rendimiento del dispositivo (e.g., cuando hay una conexión Wi-Fi activa);
- usa bases de datos externas de aplicaciones *malware* y repotencia la base de datos del servidor para adquirir nuevas definiciones de *malware*; y
- analiza y valida una aplicación antes de ser instalada, independiente de su procedencia (sea que provenga de desarrolladores de Google Play o de otras tiendas de aplicaciones).

Cliente

Incluye:

- módulo de comunicación, a cargo de la comunicación entre el cliente y el servidor;
- escuchador de eventos, encargado de la ejecución de los escaneos necesarios una vez se ha identificado un programador de eventos;
- módulo de escaneo, responsable de la ejecución de los escaneos, iniciando escaneos remotos y actualizando la base de datos del dispositivo;
- módulo de notificación, responsable de gestionar las notificaciones de usuario;
- interfaz gráfica de usuario [*graphics user interface*, GUI], responsable de controlar la interacción entre el usuario y la aplicación; y
- módulo de acceso a datos, que permite la persistencia de información y comunicación con la base de datos.

Servidor

Incluye:

- módulo de comunicación y registro, que permite la recepción y respuesta a las peticiones del cliente a través de su propio protocolo de comunicación;
- módulo de acceso a datos, responsable de la comunicación con la base de datos y de la gestión de información;
- módulo de escaneo, responsable de la gestión (dinámica y estática) de las aplicaciones registradas y escaneadas para analizar y validar;
- *crawler*, aplicación que permite descargar las aplicaciones desde Google Play y lleva el registro en la cola restante de los escaneos; y
- base de datos, que contiene la persistencia de la información proveniente de los módulos de comunicación y escaneo, y del *crawler* –los datos almacenados representan las aplicaciones escaneadas o por escanear, reportes de seguridad, detalles de cada análisis a cada aplicación, lista de permisos, lista de emuladores para cada prueba, etc.–.

Escáner y herramientas

El servidor SafeCandy es la plataforma central en la nube,

and repowering the database server to acquire new malware definitions; and

- analysis and validation of an application before it is installed, regardless of whether this comes from developers of Google Play or other Android app stores.

Client

Include:

- communication module, that allows communication between the client and the server;
- event listener, in charge of executing the necessary scans once it has identified a scheduled event;
- scanning module, responsible for performing local scans, initiating remote scans and updating the database of the device;
- notification module, responsible for managing user notifications;
- Graphic User Interface [GUI], responsible for controlling the interaction between the user and the application; and
- data access module, that allows the persistence of information and communication with the database.

Server

Include:

- communication and register module, that allows the reception and response to customer requests through its own communication protocol;
- data access module, responsible for communication with the database and likewise the information management;
- scanning module, responsible for managing the (dynamic and static) scans registered applications to analyze and validate;
- crawler, application that allows downloading of applications from Google Play and carries registration in the queue remaining of scans; and
- database, containing the persistence of the information provided by the communication modules, scanning and crawler; the stored data represents the scanned applications or those that have not been scanned yet, security reports, details of each analysis of each application, permission list, list of emulators for dynamic test, etc.

Scan and tools

The SafeCandy server is the central cloud platform dedicated to analyzing new cases of malware from the

clients (**Figure 4**). The system uses both static and dynamic analysis. Additionally, both analysis techniques use information from three phishing databases, PhishTank, WOOT, and Google Safe Browsing. Furthermore, the crawler module is dedicated to downloading applications from Google Play. Finally, all the information generated from the analyses and the crawler will be saved into the database server.

Static analysis tools

- Androguard is a tool whose function is the analysis of malware from reverse engineering. Among its features are: indicates the risk of an application, validates if an application is in a database of malware, and accesses the static analysis of the code, such as permits, instructions, etc.;
- APKtool is a tool to decompile an APK file's content and likewise re-compile;
- filescan.py is a code that allows analysis of all the files contained in the APK, with the ability to identify archives, DEX files, ELF executable files, text files containing URLs, phone numbers, and commands shell;
- Scanapp.py is a development that makes most of the static scans using Androguard, APKtool and filescan.py;
- Dex2Jar allows DEX files to be decompiled to Java code; and
- Android Asset Packing Tool, to get information from the manifest file for each application.

Service tools

- WOT service, checks the reputation and reliability of a URL;

está dedicada a analizar nuevos casos de *malware* desde los clientes (**Figura 4**). El sistema usa ambos, análisis estático y análisis dinámico, adicionalmente ambas técnicas de análisis usan información proveniente de tres bases de datos de phishing: PhishTank, WOOT y Google Safe Browsing. Además el módulo crawler está dedicado a descargar las aplicaciones desde Google Play. Finalmente, Toda la información generada por los análisis y el módulo crawler se guarda en la base de datos del servidor.

Herramientas de análisis estático

- Androguard, herramienta cuya función es el análisis de *malware* con ingeniería inversa, que tiene entre sus características la capacidad de: indicar el riesgo de una aplicación, validar si una aplicación está en una base de datos de *malware* y evaluar el análisis estático del código, así como permisos, instrucciones, etc.;
- APKtool, herramienta para decompilar –y volver a compilar– el contenido de un archivo APK;
- filescan.py, código que permite el análisis de todos los archivos contenidos en el APK, con la habilidad para identificar archivos DEX, archivos ejecutables ELF, archivos de texto con URL, números de teléfono y comandos *shell*;
- Scanapp.py, desarrollo que hace la mayoría de análisis estáticos usando Androguard, APKtool y filescan.py;
- Dex2Jar, que permite decompilar a código Java archivos DEX; y
- Android Asset Packing Tool, herramienta para obtener información desde el *manifest file* para cada aplicación.

Herramientas de servicios

- WOT, revisa la reputación y confiabilidad de una URL;
- PhishTank, permite descargar la base de datos de información de URLs con *phishing*; y

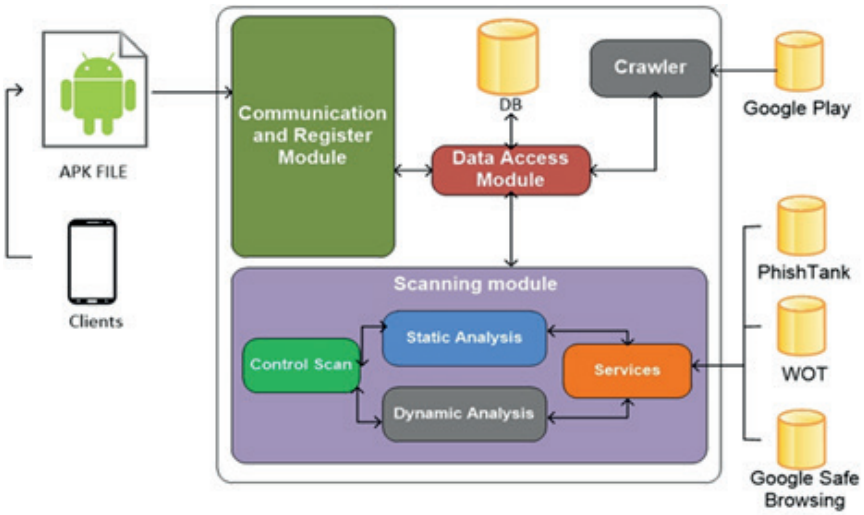

Figure 4. SafeCandy server / Servidor SafeCandy

- Google Safe Browsing, API de Google que busca la reputación y confiabilidad de una URL en la base de datos.

Herramientas de análisis dinámico

- Tcpdump, que permite la captura de paquetes enviados a un receptor desde un computador conectado;
- Android Debug Safe, herramienta de ventana de comandos que facilita la comunicación, sea con una instancia de un emulador o con un dispositivo con el OS Android;
- Drozer, que asume el rol de una aplicación Android e interactúa con el Dalvik VM para encontrar vulnerabilidades en aplicaciones y servicios.

Ventajas

SafeCandy es un sistema que, a diferencia de otros ambientes, implementa una arquitectura que permite la incorporación de varias herramientas y metodologías para la detección de software malicioso. Con el fin de descubrir comportamientos inusuales para cada aplicación la solución implementa análisis estático y análisis dinámico sobre los *sandbox* del AVD de manera que es capaz de hacer frente a todas las deficiencias de cada uno de los métodos. Otra de las ventajas es la modificación del análisis a través de ASEF, que busca reducir el tiempo de estudio para cada aplicación sin afectar la eficiencia de los procesos y la integración de diferentes bases de datos de *malware* para Android. A diferencia de otras soluciones en el mercado, SafeCandy fue desarrollado con la incorporación de un nuevo concepto que tiene como objetivo proporcionar una buena experiencia. La aplicación incluye dos formas de uso, automático –para quienes prefieren un modo fácil de usar–, y "paranoico" para quienes desean un sistema con seguridad completa y tienen conocimientos superiores al promedio en Android. Finalmente, a través de su arquitectura es posible implementar soluciones en la nube, como aprendizaje de máquina y minería de datos.

IV. Experimento

El experimento propuesto consistió en evaluar *malware* genérico en una muestra de antivirus seleccionados. Dado que SafeCandy es una solución basada en la nube, no se tendrá en cuenta el momento de la detección. La muestra analizada fue seleccionada con base en tres tendencias de investigación de código malicioso en Android (Zhou, & Jiang, 2012; Grace, Zhou, Zhang, Zou, & Jiang, 2012; Yan & Yin, 2014), las cuales han concluido que muchas variantes de *malware* actuales son modificadas con comportamientos específicos o funciones dependiendo del alcance del ataque. La **Tabla 1** presenta la muestra seleccionada de *malware*.

Los antivirus para escanear, seleccionados teniendo en cuenta su ranking en el mercado de aplicaciones Google Play, son: Antivirus Security de AVG Mobile, Mobile Security & Antivirus -de AVAST Software, CM Security & Find My Phone

- PhishTank service, allows downloading of database information of phishing URL; and
- Google Safe Browsing service is an API from Google that searches for the reputation and reliability of a URL in the database.

Dynamic analysis tools

- Tcpdump allows the capture of packets sent to and received from a connected computer;
- Android Debug Safe is a command line tool that facilitates communication with either an instance of an emulator or a device with an Android operating system; and
- Drozer, which assumes the role of an Android application and interacts with the Dalvik VM in order to find vulnerabilities in applications and devices.

Advantages

SafeCandy is a system that implements an architecture that is able to incorporate various tools and methodologies for the detection of malicious software, unlike other existing environments. In order to discover unusual behaviors of each application, the solution implements both static and dynamic analysis on the AVD's sandboxes so as to be able to deal with the weaknesses that occur in each of the methods. Another advantage is the modification of the analysis through ASEF, with which it seeks to reduce the study time for each application without affecting the efficiency of the process and the integration of different databases of malware for Android. Unlike other solutions on the market, the developed SafeCandy incorporates a new concept that aims to provide a good experience. There are two types of users, firstly users who need an automatic and easy-going mode and secondly a "paranoid" mode for people who want a complete security system and have advanced knowledge in Android. Finally, through its architecture it is possible to implement cloud solutions such as machine learning and data mining.

IV. Experiment

The proposed experiment was to evaluate generic selected malware in a sample of selected antiviruses. Since SafeCandy is a solution based on the cloud, it will not take into account the time of detection.

The analyzed samples were selected based on three research trends in malicious code in Android (Zhou, & Jiang, 2012; Grace, Zhou, Zhang, Zou, & Jiang, 2012;

Yan & Yin, 2014). In such investigations it has been concluded that many current malware variants are modified with specific behaviors or functions depending on the scope of the attack. **Table 1** presents the selected samples of malware.

de Cheetah Mobile, Norton Security & Antivirus de Norton-Mobile, Avira Antivirus Security de AVIRA, Free Antivirus & Security de McAfee Mobile Security, Mobile Security & Antivirus de ESET, Kaspersky Internet Security de Kaspersky Lab y Bitdefender Antivirus Free -de Bitdefender.

Table 1. Samples of malware / Muestra de malware

Name / *Nombre*	**Description** / *Descripción*	**Type** / *Tipo*
Andr/PJApps -C	Refers to an application that has been modified using a public internet tool. / *Aplicación que ha sido modificada mediante una herramienta de internet pública.*	Malware
Andr/BBridge -A	Uses an exploit to escalate privileges in order to install additional malware. It can scan and delete SMS incoming notification messages that the user may be charged for. / *Código malicioso que usa un exploit para escalar privilegios con el objetivo de instalar malware adicional. Este puede escanear y borrar notificaciones de mensajes SMS entrantes, por los que el usuario tiene que pagar.*	Malware
Andr/Generic- S	May include features from privilege escalation exploits to aggressive adware programs / *Variedad de aplicaciones maliciosas que pueden incluir características desde exploits para escalar privilegios hasta programas adware agresivos.*	Malware
Andr/batterd -A	Masquerades as an application to save battery life on Android devices, its goal is actually to send identified information to a server using HTTP and display ads on the phone. / *Se hace pasar por una aplicación para guardar la vida de la batería sobre dispositivos Android, pero su objetivo es enviar información de identificación a un servidor usando HTTP y desplegar anuncios (ads) en el dispositivo.*	Malware
Andr/DrSheep -A	Allows the developer/hacker to hack or hijack sessions from Facebook, Twitter or LinkedIn in a wireless network environment. / *Permite a los desarrolladores/hackers hackear o robar sesiones de Facebook, Twitter o LinkedIn en un ambiente de red inalámbrico.*	Malware
Android-Keylogger.net	Software to monitor an Android device type, and allows the capture of calls, text messages, contacts, calendars, photos, visited websites, installed applications and phone usage. It is installed directly on the phone / *Software para monitorear un dispositivo de tipo Android, y permite la captura de llamadas, mensajes de texto, contactos, calendarios, fotos, sitios web visitados, aplicaciones instaladas y el uso del teléfono. Se instala directamente en el dispositivo.*	Keylogger
KidLogger	Uses an extra keyboard installed on the device to store all the information that the user types. Then it allows the attacker to review the information locally or send it remotely. / *Software que usa un teclado extra instalado en el dispositivo para almacenar toda la información que el usuario digita, de tal manera que le permite al atacante revisar la información localmente o enviarla a un sitio remoto.*	Keylogger
Trojan.Android.Geinimi.A	One of the first variants of the Trojan Geinimi Android, and allows the device to send private information like IMEI and IMSI to a remote server location. The server is able to listen to multiple commands such as send / receive SMS, send contact details, calls initialize, install / uninstall apps / *Una de las primeras variantes del troyano Geinimi Android que le permite al dispositivo enviar información privada –como el IMEI y el IMSI– a un servidor localizado remotamente. El servidor es capaz de escuchar a múltiples comandos como enviar y recibir mensajes SMS, enviar detalles de contactos, iniciar llamadas, instalar y desinstalar aplicaciones.*	Trojan / *Troyano*
Backdoor.AndroidOS.Torec	A trojan for the TOR network. In its first version the malicious code can send information from the phone to an external server using the TOR network, and like the previous Trojan, allows the information to be sent in a "silent" mode. Type: Trojan / *Troyano para la red TOR; en su primera versión puede enviar información desde el teléfono a un servidor externo usando la red TOR y, como el anterior troyano, permite enviar la información en un modo "silencioso".*	Trojan / *Troyano*
Android-Trojan.Koler.A	A new generation of Trojan. Once installed on the device of the victim, the software proceeds to "hijack" the information in the device. The malicious software blocks the device management including information and informs the user that the device has been blocked for your safety, and then demands a ransom to release the information again. / *Nueva generación de troyano, una vez es instalado en el dispositivo de la víctima procede a "piratear" la información en el dispositivo; bloquea la gestión del dispositivo –incluyendo información–, le informa al usuario que el dispositivo ha sido bloqueado por su seguridad y luego exige un rescate para liberar la información de nuevo.*	Trojan / *Troyano*
Android.SpamSold.A	A rootkit for Android, one of the most complexes in terms of code and behavior. After the application has been installed it proceeds to hide the icon and run it in the background, and then advances to hide itself in another application. Like other Trojans, the device periodically sends information to the attacker and sends messages to premium numbers. / *Rootkit para Android, uno de los más complejos en términos de código y comportamiento, después de que la aplicación ha sido instalada procede a ocultar el icono, correr en segundo plano y ocultarse dentro de otra aplicación; al igual que otros troyanos, periódicamente envía información al atacante y mensajes a números premium.*	Rootkit

V. Resultados

Con el fin de poder comparar diferentes antivirus se usaron máquinas virtuales con Android 4.2.2. El computador donde corrieron las máquinas virtuales fue configurado usando las siguientes especificaciones: Intel Core i7-2630QM a 2Ghz, 12GB RAM, Nvidia Geforce GTX 560m con 3GB, Windows 8.1 x64. El software usado para virtualizar fue VMWare Workstation 10.0.1. La **Tabla 2** presenta los resultados obtenidos.

VI. Conclusiones

Como se ha visto, el análisis estático y el análisis dinámico pueden usar diferentes medios para detectar comportamientos y código fuente que puede ser clasificado como *malware*. Además, los métodos de aprendizaje automático y sandbox pueden simular y prevenir el *malware* sin afectar la integridad del sistema Android.

SafeCandy presenta resultados de detección del 100% de la muestra seleccionada, por lo que se puede afirmar que su desempeño es similar al de Kaspersky y BitDefender en la detección; por otra parte, respecto de los productos mejor posicionados en el mercado, esto es AVG (con 90% de éxito) y AVAST (con 72% de éxito), SafeCandy se muestra superior en la detección con la muestra de *malware* seleccionado. Sin embargo, se debe comprobar si el sistema que maneja en SafeCandy el proceso de desinstalación es la mejor solución, ya que algunos troyanos tratan de ocultar instalaciones o desinstalaciones en Android.

Después de desinstalar el Rootkit Android.SpamSold.A, la máquina virtual presenta un comportamiento errático (notificaciones de error, y el retraso al desplazarse a través de las ventanas).

Figure 5. Threat detection graphic / Gráfica de detección de amenazas

The antiviruses selected for scanning were chosen taking into account their ranking, ratings and positioning in the Google Play marketplace: Antivirus Security de AVG Mobile, Mobile Security & Antivirus -de AVAST Software, CM Security & Find My Phone de Cheetah Mobile, Norton Security & Antivirus de NortonMobile, Avira Antivirus Security de AVIRA, Free Antivirus & Security de McAfee Mobile Security, Mobile Security & Antivirus de ESET, Kaspersky Internet Security de Kaspersky Lab y Bitdefender Antivirus Free -de Bitdefender.

V. Results

In order to be able to compare different antivirus software, virtual machines with Android 4.2.2 were used. The computer where the virtual machines were installed used the following specifications: Intel Core i7-2630QM at 2Ghz, 12GB RAM, Nvidia geforce gtx 560m with 3GB, Windows 8.1 x64. The software used to virtualize was VMWare Workstation 10.0.1. **Table 2** presents the results obtained.

VI. Conclusions

As seen before, the static and dynamic analysis can use different mediums to detect behaviors and source code that can be classified as malware. Additionally, machine learning methods and sandboxes can simulate and prevent malware without affecting the integrity of the Android system.

SafeCandy presents results of 100 % detection for the selected sample. The application is therefore equal to the Kaspersky and Bitdefender antiviruses in detection. Compared to versions that are better positioned in the marketplace, such as AVG (90%) and AVAST (72%),

Table 2. Threat detection results / Resultados de detección de amenazas

Comparative Table	Y = It is been detected	N = It has not been detected	Malware	Malware	Malware	Malware	Malware	Keylogger	Keylogger	Trojan	Trojan / TOR	Trojan / Hijacker	Rootkit		
Software Name	**Enterprise**	**Software type**	AndroPJApps-C	Andr/BBridge-A	Andr/Generic-S	AndroBatterD-A	Andr/DrSheep-A	Android-Keylogger	KidLogger	Troyan.Android.GeinimiA	Backdoor.AndroidOS.Torec	Trojan.Koler.A	Android.SpamSold.A	**Detected**	**Percentage**
Antivirus security - FREE	AVG Mobile	Antivirus	Y	Y	Y	Y	Y	Y	N	Y	Y	Y	Y	10 / 11	90%
Mobile Security & Antivirus	AVAST Software	Antivirus	Y	Y	N	Y	Y	Y	Y	Y	N	N	Y	8 / 11	72%
CM Security & Find My Phone	Cheetah Mobile	Antivirus	Y	N	Y	Y	Y	N	N	Y	N	Y	Y	7 / 11	63%
Norton Security & Antivirus	Lookout Mobile Security	Antivirus	N	Y	Y	Y	Y	Y	N	Y	N	Y	Y	8 / 11	72%
Avira Antivirus Security	AVIRA	Antivirus	Y	Y	Y	N	Y	Y	Y	Y	Y	N	N	8 / 11	72%
Free Antivirus & Security	McAfee Mobile Security	Antivirus	Y	N	Y	Y	N	Y	Y	Y	Y	N	Y	8 / 11	72%
Mobile Security & Antivirus	ESET	Antivirus	Y	Y	Y	Y	Y	Y	Y	Y	N	Y	N	9 / 11	81%
Kaspersky Internet Security	Kaspersky Lab	Antivirus	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	11 / 11	100%
Bitdefender Antivirus Free	Bitdefender	Antivirus	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	Y	11 / 11	100%
Safecandy	**Password Consulting Services**	**Real time analysis - Virtualized**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**Y**	**11 / 11**	**100%**

SafeCandy is superior in detection with the selected malware sample. It should be checked whether the system in SafeCandy that handles the uninstallation process is the best solution, because some Trojans try to hide installations / uninstallations in Android.

After uninstalling the Rootkit Android.SpamSold.A, the virtual machine presented erratic behavior (error notifications, and lag when scrolling through the windows).

For the next upgrade of the SafeCandy platform, it is important to consider implementing machine learning techniques and data mining due to the constantly evolving nature of the malware in Android. ST

Para la próxima actualización de la plataforma SafeCandy es importante considerar la implementación de técnicas de aprendizaje automático y minería de datos, debido a la naturaleza en constante evolución del *malware* en Android. ST

References / Referencias

Android Open Source Project (n.d). *Security*. Retrieved from: https://source.android.com/devices/tech/security/

Au, K. W. Y., Zhou, Y. F., Huang, Z., & Lie, D. (2012). Pscout. Analyzing the android permission specification. In *Proceedings of the 2012 ACM Conference on Computer and Communications Security*, (pp. 217-228). New York, NY: ACM.

Batyuk, L., Herpich, M., Camtepe, S. A., Raddatz, K., Schmidt, A., & Albayrak, S. (2011). Using static analysis for automatic assessment and mitigation of unwanted and malicious activities within Android applications. In *2011 6th International Conference on Malicious and Unwanted Software (MALWARE)*, (pp. 66-72). Piscataway, NJ: IEEE.

Bishop, M.A. (2002). *The art and science of computer security*, Boston, MA: Addison-Wesley Longman.

Burguera, I., Zurutuza, U., & Nadjm-Tehrani, S. (2011). Crowdroid: behavior-based malware detection system for android. In Proceedings of the 1st ACM workshop on Security and privacy in smartphones and mobile devices, (pp. 15-26). New York, NY: ACM.

Dharmdasani H., & Pidathala V. (2014, march 31). Android.MisoSMS : Its Back! Now With XTEA. Retrieved from: https://www.fireeye.com/blog/threat-research/2014/03/android-misosms-its-back-now-with-xtea.html

Drake, J. J., Lanier, Z., Mulliner, C., Fora, P. O., Ridley, S. A., & Wicherski, G. (2014). *Android hacker's handbook*. Indianapolis, IN: John Wiley & Sons.

Enck, W., Gilbert, P., Han, S., Tendulkar, V., Chun, B. G., Cox, L. P., ... & Sheth, A.N. (2014). TaintDroid: an information-flow tracking system for realtime privacy monitoring on smartphones. *ACM Transactions on Computer Systems (TOCS), 32*(2), 5.

Enck, W., Octeau, D., McDaniel, P., & Chaudhuri, S. (2011). A study of Android application security. In *USENIX security symposium* (Vol. 2, p. 2). Berkeley, CA: Usenix.

Forristal, J. (2014). Android fake ID vulnerability. In *Black Hat USA, 2014. Black Hat Materials*, Retrieved from: https://www.blackhat.com/docs/us-14/materials/us-14-Forristal-Android-FakeID-Vulnerability-Walkthrough.pdf

Fuentes, M. & Gómez, J. (2014). Valoración de la plataforma ASEF como base para detección de malware en aplicaciones Android. *Ingenium*, *8*(21), 11-23.

Ghorbanzadeh, M., Chen, Y., Ma, Z., Clancy, T. C., & McGwier, R. (2013), A neural network approach to category validation of android applications. In *2013 International Conference on Computing, Networking and Communications (ICNC)*, (pp. 740-744), Piscataway, NJ: IEEE.

Gopan, D., & Reps, T. (2007). Low-level library analysis and summarization. In *Computer aided verification* (pp. 68-81). Berlin-Heidelberg, Germany: Springer.

Grace, M., Zhou, Y., Zhang, Q., Zou, S., & Jiang, X. (2012). Riskranker: scalable and accurate zero-day android malware detection. In *Proceedings of the 10th international conference on Mobile systems, applications, and services*, (pp. 281-294). New York, NY: ACM.

Khandelwal, S. (2014a, March 26). *Android malware 'dendroid' targeting Indian users* [blog The Hacker News]. Retrieved from: http://thehackernews.com/2014/03/android-malware-dendroid-targeting_26.html

Khandelwal, S. (2014b, May 6). Police ransomware malware targeting Android smartphones [blog The Hacker News]. Retrieved fromhttp://thehackernews.com/2014/05/police-ransomware-malware-targeting.html

Mutz, D., Robertson, W., Vigna, G., & Kemmerer, R. (2007). Exploiting execution context for the detection of anomalous system calls. In *Recent advances in intrusion detection*, (pp. 1-20. .Berlin-Heidelberg, Germany: Springer.

Nadji, Y., Giffin, J., & Traynor, P. (2011). Automated remote repair for mobile malware. In *Proceedings of the 27th Annual Computer Security Applications Conference*, (pp. 413-422). New York, NY: ACM.

Navarro, A., Sebastián, L., Urcuqui, C., Fuentes, M., & Gomez, J. (2014). Análisis y caracterización de frameworks para detección de aplicaciones maliciosas en Android. In *XIV Jornada Internacional de Seguridad Informática ACIS 2014,* (Art.1) [CD]. Available at http://52.0.140.184/typo43/index.php?id=2114

Oja, M., Kaski, S., & Kohonen, T. (2003). Bibliography of self-organizing map (SOM) papers: 1998-2001 addendum. *Neural Computing Surveys, 3*. Retrieved from http://www.cis.hut.fi/research/refs/NCS_vol3_1.pdf

Petsas, T., Voyatzis, G., Athanasopoulos, E., Polychronakis, M., & Ioannidis, S. (2014). Rage against the virtual machine: hindering dynamic analysis of android malware. In *Proceedings of the Seventh European Workshop on System Security*, (p. 5). New York, NY: ACM.

Pichai, S. (2014). *Google I/O 2014 - Keynote* [video. 6:43m]. Retrieved from: https://www.google.com/events/io

Portokalidis, G., Homburg, P., Anagnostakis, K., & Bos, H. (2010). Paranoid Android: versatile protection for smartphones. In *Proceedings of the 26th Annual Computer Security Applications Conference*, (pp. 347-356). New York, NY: ACM.

Smalley, S., & Craig, R. (2013). Security enhanced (SE) Android: Bringing flexible Mac to Android. In *20th Annual Network and Distributed System Security Symposium (NDSS'13)*, (pp. 20-38). Reston, VA: The Internet Society.

Tor Project (s.f). *Anonymity online*. Retrieved from: http://www.torproject.org/

Wei, T., Zhang, Y., Xue, H., Zheng, M., Ren, C., & Song, D. (2014). Sidewinder: Targeted attack against Android in the golden age of ad libraries. In *Black Hat 2014, Black Hat Materials*. Retrieved from: https://www.blackhat.com/docs/us-14/materials/us-14-Wei-Sidewinder-Targeted-Attack-Against-Android-In-The-Golden-Age-Of-Ad-Libs.pdf

Yadav, N. P. & Shivamurthy, R. C. (2013). Faamac: Forensic Analysis of Android Mobile Applications using Cloud Computing. *International Journal on Recent and Innovation Trends in Computing and Communication, 2*(5), 1069-1073.

Yan, L. K., & Yin, H. (2014). DroidScope: Seamlessly reconstructing the OS and dalvik semantic views for dynamic android malware analysis. In *USENIX security symposium*, (pp. 569--584).

Yerima, S. Y., Sezer, S., & Muttik, I. (2014). Android malware detection using parallel machine learning classifiers. In *2014 Eighth International Conference on Next Generation Mobile Apps, Services and Technologies (NGMAST)*, (pp. 37-42). Piscataway, NJ: IEEE.

Zhou, W., Zhou, Y., Jiang, X., & Ning, P. (2012). Detecting repackaged smartphone applications in third-party android marketplaces. In *Proceedings of the second ACM conference on Data and Application Security and Privacy*, (pp. 317-326). New York, NY: ACM.

Zhou, Y., & Jiang, X. (2012). Dissecting android malware: Characterization and evolution. In *2012 IEEE Symposium on Security and Privacy (SP),* (pp. 95-109). Piscataway, NJ: IEEE.

CURRICULUM VITAE

Sebastián Londoño Systems Engineer (emphasis in Management and Computing) and Master in Computing Management and Telecommunications from the Universidad Icesi (Cali-Colombia). As part of Informatics and Telecommunications research group [i2t] participated in "Safe Candy: analysis, validation and security configuration for Android apps" project / Ingeniero de Sistemas con énfasis en Administración e Informática y Máster en Gestión de Informática y Telecomunicaciones de la Universidad Icesi (Cali-Colombia). Como miembro del grupo de investigación en Informática y Telecomunicaciones [i2t] participó en el proyecto "Safe Candy: Plataforma de análisis, validación y configuración de seguridad de aplicaciones Android".

Christian Camilo Urcuqui Systems Engineer (emphasis in Management and Computing) and Master in Computing Management and Telecommunications student from the Universidad Icesi (Cali-Colombia). As part of Informatics and Telecommunications research group [i2t] participated in "Safe Candy: analysis, validation and security configuration for Android apps" project / Ingeniero de Sistemas con énfasis en Administración e Informática y estudiante de la Maestría en Gestión de Informática y Telecomunicaciones de la Universidad Icesi (Cali-Colombia). Como miembro del grupo de investigación en Informática y Telecomunicaciones [i2t] participó en el proyecto "Safe Candy: Plataforma de análisis, validación y configuración de seguridad de aplicaciones Android".

Manuel Fernando Fuentes Amaya Electronic Engineer from the Universidad del Cauca. He works for Password Consulting Services (Bogotá, Colombia) and was a member of developers team in "Safe Candy: analysis, validation and security configuration for Android apps" project / Ingeniero Electrónico de la Universidad del Cauca. Como parte del equipo de ingenieros de desarrollo de Password Consulting Services (Bogotá-Colombia) –empresa a la que continúa vinculado– participó en el proyecto "Safe Candy: Plataforma de análisis, validación y configuración de seguridad de aplicaciones Android".

Johan Alberto Gómez Girón Electronic Engineer from the Universidad del Cauca. He works for Password Consulting Services (Bogotá, Colombia) and was a member of developers team in "Safe Candy: analysis, validation and security configuration for Android apps" project / Ingeniero Electrónico de la Universidad del Cauca. Como parte del equipo de ingenieros de desarrollo de Password Consulting Services (Bogotá-Colombia) –empresa a la que continúa vinculado– participó en el proyecto "Safe Candy: Plataforma de análisis, validación y configuración de seguridad de aplicaciones Android".

Andrés Navarro Cadavid Electronic Engineer and Magister in Technology Management of the Universidad Pontificia Bolivariana (Medellín, Colombia) and Doctor of Engineering in Telecommunications of the Universidad Politécnica de Valencia (Spain). Full time professor and leader of the Informatics and Telecommunications research group (i2T) attached to the Information and Communications Department at the Universidad Icesi (Cali-Colombia). Counselor at the National Program of Electronics, Telecommunications and Informatics [ETI]. Spectrum Management and Cognitive Radio are two of his major interest areas / Ingeniero Electrónico y Máster en Gestión Tecnológica de la Universidad Pontificia Bolivariana y Doctor Ingeniero en Telecomunicaciones de la Universidad Politécnica de Valencia (Spain). Profesor de planta y líder del agrupo de investigación en informática y telecomunicaciones (i2T), adscrito al Departamento de Tecnologías de la Información y las Comunicaciones de la Universidad Icesi; y Consejero del Programa Nacional de Electrónica, Telecomunicaciones e Informática [ETI]. Dos de sus mayores áreas de interés son la gestión del espectro y la radio cognitiva.