Sistema antivirus multiplataforma en tiempo real usando técnicas heurísticas y proactivas

Ramiro P. Laura Murillo* , Ernesto N. Tumi Figueroa†
Recibido 23 de marzo de 2015, aceptado 19 de junio de 2015
Received: March 23, 2015 Accepted: June 19, 2015

RESUMEN

La proliferación y propagación de una gran variedad de virus informáticos a través de los dispositivos de almacenamiento extraíbles como son: la contaminación lógica del sistema operativo, la destrucción de archivos y carpetas, la posibilidad de daños en los medios de almacenamiento y otros. La presente investigación tuvo como objetivo el desarrollar un Sistema Antivirus con detección en tiempo real usando técnicas heurísticas, proactivas y auto-actualizable, aprovechando la conexión a Internet. Cumpliendo con este objetivo de desarrollo el sistema antivirus denominado AV-Fenix, realiza las operaciones de detección, eliminación y prevención de software malicioso. La metodología que se usó fue de la Programación Extrema (XP) por su flexibilidad en el desarrollo de aplicaciones pequeñas y eficaces sobre todo porque el antivirus es modificado e incrementado con mejoras necesarias por la constante evolución de los malwares. El modelamiento basado en el Lenguaje de Modelamiento Unificado (UML) y la Métrica de Validación ISO/IEC 9126. Finalmente se ha analizado, diseñado e implementado el Software Antivirus que detecta una considerable variedad de Malwares de acuerdo a las pruebas en un 80%, siendo 50% detección por firmas de virus y el 30% por el escáner heurístico proactivo, además el actualizador permite que una versión del antivirus pueda ser actualizada, se ha escrito rutinas que permitieron que el antivirus pueda ejecutarse en distintos sistemas operativos y de modo autónomo para una detección en modo pasivo. La proactividad ha permitido escribir el detector en tiempo real para las unidades de almacenamiento extraíble para prevenir infecciones.

Palabras clave: Interfaces; antivirus; heurísticas, virus, malwares, detección.

* Magister en Informática. Universidad Nacional del Altiplano. Email: rplm.mx@gmail.com
† Magister en Informática. Universidad Nacional del Altiplano.

ABSTRACT

The proliferation and propagation of a great variety of computer viruses trough removable storage devices are as follows: logic contamination of operating system, destruction of files and folders, probable damages in storage means, and others. The objective of the present research is To Develop an Antivirus System with real-time detection using heuristic, proactive, and self-updating techniques, by using the Internet. To comply with this development objective, the Antivirus System called AV-Fenix, performs the operations of detection, removal, and prevention of malicious software. The methodology used was Extreme Programming (XP) because of its flexibility in developing small and effective applications mainly because the antivirus application is modified and increased with improvements due to the constant evolution of malware. The modeling based on the Unified Modeling Language (UML) and the Metrics of Validation ISO / IEC 9126. Finally, we have analyzed, designed, and implemented the Antivirus Software that detects a considerable amount of Malware according to tests at 80%, being 50% virus signature detection and 30% by the Proactive Heuristic Scanner. The updater also allows a version of the antivirus to be updated, routines have been written which let the antivirus to be run on different operating systems and autonomously for detection in passive mode. Proactivity has allowed to write the detector in Real time for removable storage units to prevent infection.

Keywords: Interfaces; antivirus; heuristics, virus, malware, detection.

INTRODUCCIÓN

La seguridad informática como investigación tiene como principal labor la seguridad de la información y estabilidad del sistema operativo host, evitar la infiltración de programas sospechosos y/o potencialmente peligrosos, todas estas labores usualmente se encargan a un experto en seguridad informática.

El usuario normal requiere un Software experto en seguridad informática, los algoritmos heurísticos y proactivos darán la inteligencia necesaria a nuestro sistema antivirus, en la primera etapa aprende, posteriormente aplica el conocimiento adquirido, en la segunda etapa la de proactividad debe adelantarse a situaciones que puedan ser peligrosas, los intentos de infiltración bajo archivos ejecutables, guiones, programas ejecutables y dispositivos extraíbles. Sucederán ocasiones post-infección en los que se requiere una vacuna que no dependa del sistema operativo ya infectado, para ello un antivirus provisto de su propio sistema operativo proporcionara mayor seguridad en las tareas de detección y eliminación, esto le dará al producto antivirus una mayor eficiencia frente a sus competidores, sumando la capacidad de poder usarlo en diversos sistemas operativos.

El objetivo de esta investigación ha sido: Desarrollar un sistema antivirus con detección en tiempo real usando técnicas heurísticas, proactivas y auto-actualizable, aprovechando la conexión a Internet

MATERIALES Y MÉTODOS

LOCALIZACIÓN

El estudio fue realizado en el Departamento de Puno, Provincia de Puno, Distrito de Puno, a 3820 m.s.n.m. en la Universidad Nacional del Altiplano, lugar en el cual se obtuvo los datos necesarios, instrumentos y población para poder realizar los estudios correspondientes.

METODOLOGÍA

Para el desarrollo del sistema de aleatorización de exámenes se aplicó la metodología de la Programación Extrema (Extreme Programming - XP) que se adapta hoy en día perfectamente al desarrollo del ciclo de vida del sistema y para el modelado del sistema se usará el Lenguaje de Modelamiento Unificado (Unified Modeling Language - UML).

Ventajas:

- Programación organizada.
- Menor tasa de errores.
- Satisfacción del programador.
- Solución de errores de programas.
- Versiones nuevas.
- Implementa una forma de trabajo donde se adapte fácilmente a las Circunstancias.

Desventajas:

- Es recomendable emplearlo solo en proyectos a corto plazo.
- Altas comisiones en caso de fallar.
- Imposible prever todo antes de programar.
- Demasiado costoso e innecesario.

DESARROLLO DEL SOFTWARE

Es el proceso mediante el cual se produjo el Software final, consta de las siguientes etapas: Análisis, Diseño, Codificación y Validación del Sistema. Todas las etapas se desarrollan a continuación.

Análisis: La metodología XP plantea en análisis como un permanente diálogo entre la parte empresarial y técnica del proyecto, en la que los primeros decidirán el alcance ¿Qué es lo realmente necesario del proyecto?

Diseño: El Propósito del diseño es de crear una arquitectura para la naciente implementación, el diseño arquitectural sólo puede comenzar una vez que el equipo tenga un entendimiento razonable de los requerimientos del sistema. El diseño se enfoca en la estructura, estática y dinámica, su propósito principal es de crear el esqueleto concreto del sistema sobre el cual todo el resto de la implementación se basa (Grady Booch, 1999).

Codificación: Esta etapa debe reunir las siguientes características o cualidades:

- El cliente está siempre disponible
- Se debe escribir código de acuerdo a los estándares
- Desarrollar la unidad de pruebas primero
- Todo el código debe programarse por parejas
- Integrar frecuentemente
- Todo el código es común a todos

Validación del Sistema: Cuando el experto y el ingeniero del conocimiento están convencidos de que el prototipo de sistema está terminado, hay que probarlo de acuerdo con el criterio del funcionamiento. Este también es el momento de invitar a otros expertos y/o usuarios a que prueben el sistema.

LENGUAJE DE MODELAMIENTO

Para modelar el análisis y el diseño del sistema se utilizó el Lenguaje de Modelamiento Unificado UML.

REQUERIMIENTOS DEL SISTEMA

Requerimientos funcionales: Se definieron para el sistema, los siguientes puntos más relevantes que el software debe poder realizar:

R1: Debe ser capaz de detectar software maliciosos y versiones variantes de virus anteriormente catalogados.

R2: Requiere un módulo heurístico y proactivo para la implementación y ejecución del escáner..

R3: Se debe implementar un detector a tiempo real que permita detectar cuando un dispositivo se conecta al computador y ejecutar la vacuna.

R4: Debe ser autónomo en su ejecución para ofrecer protección mientras el usuario trabaja cómodamente.

R5: Debe ser capaz de ejecutarse en otros sistemas operativos además de Windows en este caso la versión experimental para Linux y Android OS.

Requerimientos no funcionales: Interfaz agradable para un fácil entendimiento del software.

- Disponibilidad del sistema de encontrarse disponible todos los días.
- Estabilidad del sistema que soporta varios usuarios a la vez.
- Rendimiento el sistema brindará un servicio óptimo ya que es diseñado para que funcione en ambientes distribuidos.
- Mantenimiento y Escalabilidad diseñado pensando en el crecimiento del sistema.

RESULTADOS Y DISCUSIÓN

DESARROLLO

Implementación de código

Ejemplo 9 (Uso de librería VESA KrystalView en AV-Fenix).

```
/*************************************************************
 * Kronoz Project - Av-Fenix Antivirus
 * Modulo : KrystalView Environment + [KronozVideo Agent]
 *
 * Compilar : bcc -ml -B -3 kvFenix.cpp
 *************************************************************/

#include "kvGraph.h"
#include "kvFonts.h"
#include "kvWidgts.h"

class KvFenixApp : public KApp {
    private:
        BOOL OnInit();
        void OnExit();
};
```

Infección Controlada	Infectados	A Prueba	Limpiados
Gusanos	14	14	13
Carpetas Dañadas	4	4	4
%	--	100%	90%

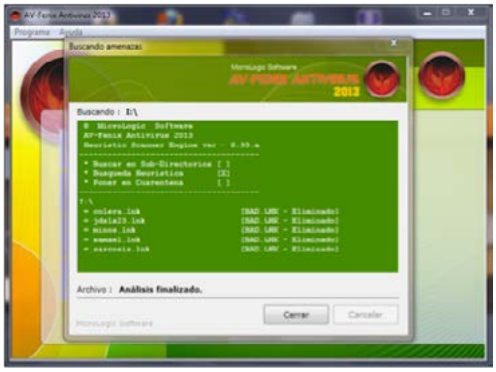
Desarrollo de Interfaz Gráfica de Usuario

Completamente desarrollado sobre wxWidgets Windows y Linux

PRUEBAS

Apertura de archivos: Se implementó un contenedor para los archivos que se desee agregar para el proceso de aleatorización.

Comprobación de Contenido: En la ventana de Historial se registraron la cantidad de detecciones realizadas.



Para la muestra se infectó bajo control la carpeta C:\PerfLogs\DEMO con 14 variedades de gusanos y dañado 4 subdirectorios.

MODELAMIENTO DEL SISTEMA

Diagrama de Clases: La clase Interfaz Principal contiene los elementos de comunicación.

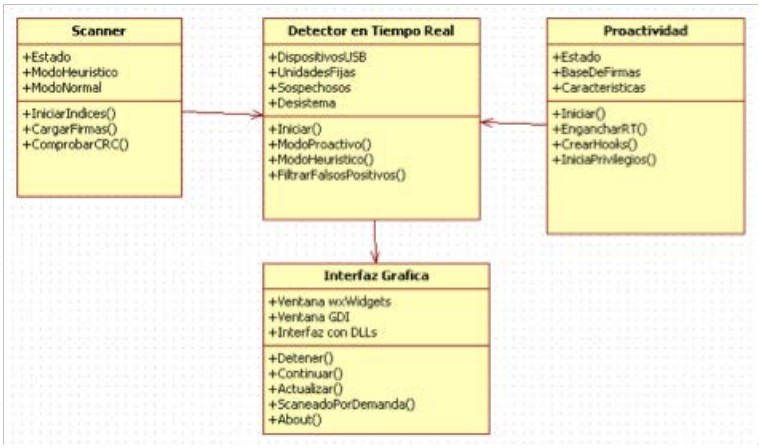


Diagrama de Secuencia:

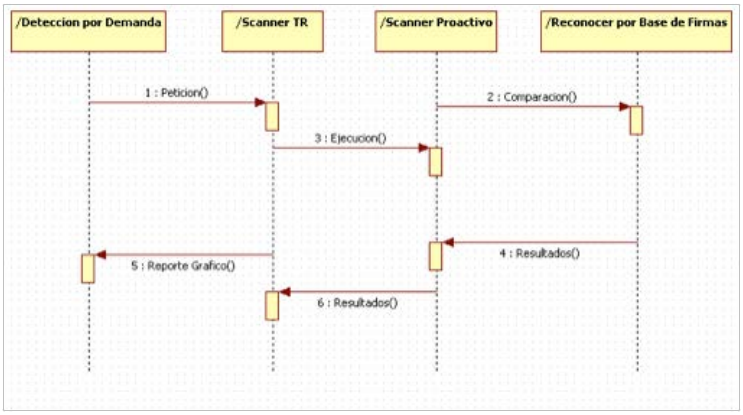
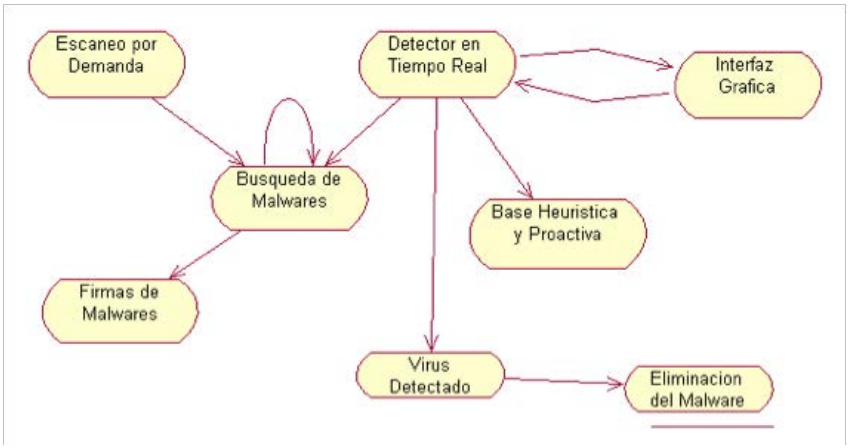


Diagrama de Actividades: Aquí las actividades realizadas.



METRICAS DE VALIDACIÓN DE SOFTWARE

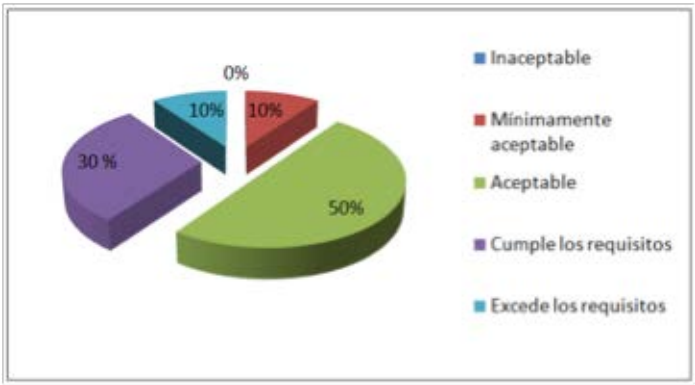
La metodología que se aplicó para validar el software es la descrita en la norma estándar ISO 9126 tomando de ella cuatro características de calidad cuantificables por medio de preguntas realizadas a los actores del Sistema.

Las características y aspectos a considerarse fueron propuestos por Abran, Khelifi, Suryn & Sefiah (2003); aspectos considerados como los más extensos desarrollados hasta la fecha.

PROMEDIO GENERAL DE LAS FICHAS DE EVALUACIÓN ISO-9126

Cuadro de decisiones ISO – 9126
Fuente: Cuadro de decisiones ISO - 9126

Clasificación	Intervalo	N°	%
Inaceptable	[27 – 54 >	0	0
Mínimamente aceptable	[54 – 81 >	5	10
Aceptable	[81 – 95 >	25	50
Cumple los requisitos	[95 - 122 >	15	30
Excede los requisitos	[122 - 135]	5	10
Total		50	100



Decisión: De acuerdo a los resultados de la calidad del software se concluyó que el Sistema Antivirus, cumple los requisitos con un promedio de 50 puntos del total de 100 puntos que se considera en el cuadro de decisiones del ISO - 9126.

Selección de métricas: La selección de métricas se obtiene a partir de los atributos especificados en el Modelo de Calidad. Se agruparon en:

- Funcionalidad
- Fiabilidad
- Utilidad
- Eficiencia

CONCLUSIONES

- Se ha analizado, codificado e implementado el esquemas más óptimos para desarrollar el Software Antivirus denominado “AV-Fenix Antivirus 2013”, proporciona protección contras los virus informáticos, que se propagan en los dispositivos de almacenamiento extraíbles (Memorias USB, SD, FlashDrives, etc.).

- Concluimos que la mejor forma de proteger el computador de las amenazas informáticas latentes en los dispositivos extraíbles, es desarrollando un detector a tiempo real para los dispositivos y notificar al escáner para que este verifique el contenido y pueda informarnos de los resultados, sin tener que interrumpir al usuario.

- Se escribió rutinas de carga y enganche (Hooking) con las APIs de Win32 que permitan notificarnos cuando una instrucción privilegiada esta por ejecutarse, de modo que el detector a tiempo real pueda actuar sin que nosotros intervengamos ello permite que el programa actué por nosotros así poder protegernos de Malwares.

- Se ha implementado un sitio Web ubicado en la dirección URL http://av-fenix.com, que permitirá a los usuarios del Software Antivirus mantener las actualizaciones y estas sean descargadas, la actualización contendrá nuevas firmas de virus informáticos, como también la descarga de versiones mejoradas del Software Antivirus, proveyendo una protección general a todos los usuarios de este Software.

Referencias

A. Borghello, C. F. (2001). *Seguridad informática sus implicancias e implementación.*

Abel, Peter. (2002). *El Universo Digital del IBM PC 80386 y Superiores. Madrid: Anaya Multimedia*

Andrew S., Tanenbaum, A. (2006). *Operating Systems Design and Implementation.* Prentice Hall.

Burns, K. (2003). *TCP/IP Analysis and Troubleshooting Toolkit. Computer network protocol.* USA: Wiley, Hoboken,

Chromatic (2004). *Guía de Bolsillo de Programación Extrema,* p.10. O'Reilly, 1th edición.

Celis, C. G. (1997). *El Universo Digital del IBM PC AT y PS/2.* Grupo Informático 1992-1997. Anaya Multimedia, Casa del Estudiante, 4ta edición.

Juares Henry. (2004). *Protección de programas ejecutables usando autómatas celulares de Wolfgang,* Tesis de grado, Facultad de Ingeniería Estadística e Informática, UNA Puno.

Kenneth Calvert, M. D. (/2001). *TCP/IP Sockets in Java - Practical Guide for Programmers.* Morgan Kaufmann, Burlington, MA, USA, 1st edition.

Knuth, D. E. (1984). *The TEX Book. Computers & Typesetting. Addison-Wesley, Reading, Massachusetts,* 15th edition. Reprinted as Vol. A of Computers & Typesetting,

Rogers, D. T. (2003). *Un Framework para la Subversión Dinámica. Glosario de Multics acrónimos y términos.* 1st edition. Seguridad y Anillos.

Rosello, M. A. R. (1988). *Programación Ensamblador en entorno MS-DOS.* Madrid: Anaya Multimedia

Julian Smart, K. H. (2006). *Cross-Platform GUI Programming with wx-Widgets.* México: Prentice Hall PTR, Pearson Education, Inc., 1st edition.

Walker, A. (2006). *Seguridad, Spam, Spyware y Virus. Seguridad Inform1atics.* Madrid: Anaya Multimedia-Anaya Interactiva.