REVISTA DIGITAL DE LAS TECNOLOGÍAS DE LA INFORMACIÓN Y LAS COMUNICACIONES
TELEM@TICA
Revista Telem@tica. Vol. 13. No. 1, enero-abril, 2014, p. 1-11 ISSN 1729-3804

IDENTIFICACIÓN DE TEXTO EN SEGURMATICA ANTIVIRUS
TEXT IDENTIFICATION IN SEGURMATICA ANTIVIRUS

Ing. Alain López Jiménez, Lic. Alejandro Rivero Pérez, Jorge Lodos Vigil
Segurmatica. Empresa de Consultoría y Seguridad Informática.
Zanja No. 651 Esq. A Soledad. Centro Habana. La Habana. Cuba.
Teléf. 8703536. Fax. 8735965. Código postal. 10300.
Ministerio de Informática y las Comunicaciones. Cuba.
alopez@segurmatica.cu, alex@segurmatica.cu, lodos@segurmatica.cu

RESUMEN: *Cuando se escanea un fichero con Segurmática Antivirus, no se busca en bruto con todos sus algoritmos, de ser así el proceso tardaría considerablemente si la cantidad de ficheros escaneados es grande. Realmente lo que hace es distribuir los algoritmos según el tipo de archivo que se escanea. Esta práctica resulta muy útil cuando el volumen de firmas y de métodos de identificaciones y descontaminaciones es muy grande. Cuando se trata de buscar en archivos de texto el resultado es aún mayor si se tiene en cuenta la cantidad de variantes de clasificaciones posibles, mientras aumenta la especialización disminuye la cantidad de algoritmos por especialidad o tipos.*
Es muy común el uso de lenguajes interpretados o scripts para explotar vulnerabilidades en los programas clientes (Internet Explorer, Mozilla Firefox, Chrome, etc.) y existen diversas técnicas para la identificación de tipos de textos, cada una con sus ventajas y desventajas. Es necesario entonces contar con una biblioteca que le permita al antivirus identificar distintos tipos de texto mediante el análisis del contenido de los archivos a partir de las técnicas existentes, además de extender su arquitectura de búsqueda y optimizar cada vez más el proceso.

Palabras Clave: identificación de texto, Segurmática Antivirus, análisis léxico, análisis sintáctico, identificación de patrones, regla.

ABSTRACT: *When scanning a file with Segurmática Antivirus, it not will search with all algorithms, if so the process could be delay considerably if the number of files scanned is large. Really what it does is distribute the algorithms depending on the type of file being scanned. This practice is very useful when the volume of signa-tures, identifications and decontaminations is very large. When it comes to search in text files the result is even greater if one takes into account the number of possible classifications and variants, while increase the speciali-zations decrease the number of algorithms by specialty or type.*
Is very common use interpreted languages or scripts to exploit vulnerabilities in clients programs (Internet Ex-plorer, Mozilla Firefox, Chrome, etc.) and there are several techniques for the identification of text types, each with its advantages and disadvantage. Therefore is necessary to have a library that could allow to the antivirus identify different types of text by the content analyze of the files with the existing techniques, as well as extend the existing search architecture and optimize the process every time more.

KeyWords: text identification, Segurmatica Antivirus, lexical analyze, syntactic analyze, patterns identification, rule.

1. INTRODUCCIÓN

Los antivirus son muy utilizados como herramientas para garantizar en alguna medida la protección de la información digital y evitar la infección mediante programas malignos, así como la propagación de la misma mediante dispositivos, redes, correos entre otras formas existentes.

Todos los días se analizan códigos malignos y se implementan algoritmos para su identificación y descontaminación, lo que incrementa los volúmenes de las detecciones por firmas y las detecciones genéricas o heurísticas [1], mejorando el nivel de protección pero aumentando considerablemente el tiempo de respuesta de los antivirus. Para impedir tal disparidad en los resultados, se emplean técnicas que permiten distribuir las búsquedas según el tipo de fichero que se analiza, así se especializan los algoritmos y las firmas por cada tipo clasificado.

Cuando se escanea un fichero con Segurmática Antivirus (SEGAV) se identifica a cual o cuales de las clasificaciones definidas pertenece y luego se ejecutan las secuencias de búsquedas con las firmas y los algoritmos establecidos para dichas clasificaciones. Debido al aumento considerable del tiempo de escáner de SEGAV en ficheros de texto, es necesario extender su arquitectura de búsqueda mediante clasificaciones de los archivos de texto para optimizar el proceso.

La clasificación de tipos en archivos de texto es complicada debido a que existen más de 250 extensiones de texto [2] y se utilizan diferentes lenguajes de codificación de programas para ejecutar acciones malignas con estos ficheros. Los lenguajes más utilizados son los interpretados o scripts que permiten explotar vulnerabilidades en los intérpretes o programas clientes [3] como Internet Explorer, Mozilla Firefox, Chrome, etc.

Algunos de estos lenguajes presentan estructuras y sintaxis en los códigos que son comunes, lo que constituye otro problema más en la identificación. Un ejemplo representativo (ver Figura. 1) lo constituyen JavaScript (JS) y Visual Basic Script (VBS) [4].

```
1
2  <script language="javascript">
3      function Write()
4      {
5          document.write();
6      }
7  </script>
8
9  <script language="vbcript">
10     sub Write()
11         document.write();
12     end sub
13 </script>
```

Figura. 1: Sintaxis JS y VBS

Es importante mencionar además la flexibilidad de los archivos de texto en cuanto a modificaciones del contenido sin afectar su esencia (permite mucha información irrelevante, líneas vacías, espacios, etc.), lo que agrava aún más las dificultades existentes en la clasificación de texto.

Existen algunas técnicas para identificar este tipo de fichero:

• La identificación por extensión (ver Tabla I): no garantiza un alto grado de fidelidad por la facilidad con que pueden ser cambiadas.

Tabla I: Extensiones de archivos de texto *

Extensión	Descripción
txt	Archivo de texto sin formato.
doc	Documento de WordPad y Microsoft Word.
eml	Mensaje de correo electrónico.

• La identificación por números mágicos o marcas (Tabla II) en las cabeceras de los archivos: que resulta tedioso detectarlas porque suelen asociarse al programa donde se creó el fichero. Su cálculo puede ser complejo y habría que analizar cada archivo para cada uno de los posibles números mágicos.

Tabla II: Firmas en ficheros de texto **

Hexadecimal	ASCII	Extensión
0D 44 4F 43	.DOC	doc
52 65 74 75 72 6E 2D 50	Return-P	eml

• La clasificación por análisis del contenido: es mucho más certera, pero el tiempo de respuesta y el costo computacional crecen directamente proporcionales al tamaño del fichero que se analiza.

**Puede consultar fileinfo.com donde aparecen la mayoría de las extensiones de archivos de texto existentes acompañadas de una breve descripción.*

***En garykessler.net y filesignatures.net puede encontrar las firmas hexadecimales y ASCII de las extensiones más comunes.*

Es necesario contar con una biblioteca que solucione mediante las técnicas descritas, las problemáticas que surgen en la identificación de archivos de texto por SEGAV, manteniendo así la proporción entre el nivel de detección y la velocidad de escaneo.

2. DESCRIPCIÓN Y ESTRUCTURA

2.1 Elementos generales

El proceso de clasificación de texto se puede dividir en dos partes:

a. Determinar si un archivo es texto para lo que el antivirus cuenta con una biblioteca de identificación de tipos.

b. Clasificar los textos según los lenguajes interpretados que contenga el mismo.

La solución propuesta resuelve la segunda parte del proceso. Utiliza el análisis léxico y sintáctico del contenido de los ficheros con el uso de patrones. Busca cadenas de caracteres o sintaxis malignas1, definidas en los lenguajes interpretados (Tabla III) más utilizados y denota las clasificaciones según los scripts reconocidos. Además resulta compleja en su ejecución, a medida que crece el tamaño del archivo y la cantidad de caracteres a analizar aumenta también el tiempo de clasificación, haciendo prácticamente imposible revisar todo el contenido de los archivos.

Tabla III: Caracteres y sintaxis de algunos scripts

Caracteres	Sintaxis	Lenguaje script
<, html, >, /	<html></html>	HyperText Markup Language (HTML)
<, script, >, /	<script></script>	JavaScript (JS)
#, /, usr, bin, perl	#!/usr/bin/perl	PERL
#, /, usr, bin, python	#!/usr/bin/python	PYTHON

La figura 2 ilustra la secuencia completa de la biblioteca para la identificación de texto.

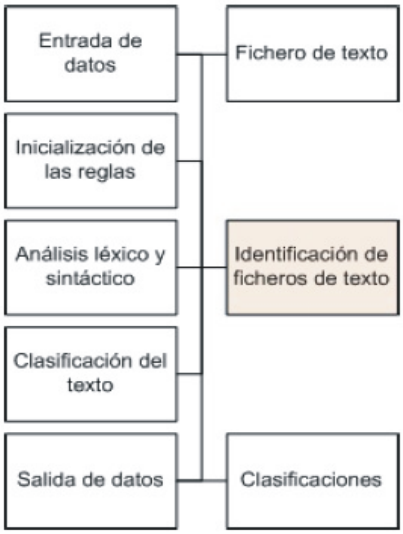

Figura. 2: Arquitectura de identificación de texto

2.1.1 Análisis léxico del contenido

El análisis léxico (Figura. 3) permite identificar secuencias de caracteres [5] específicos de los lenguajes interpretados, que cumplan con un determinado patrón. Esta secuencia corresponde a las cadenas de caracteres definidas en el lenguaje (ver Tabla III) y el patrón define la regla que la reconoce.

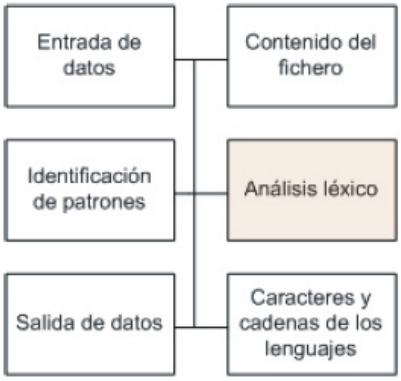

Figura. 3: Arquitectura del análisis léxico

2.1.2 Análisis sintáctico del contenido

En el análisis sintáctico (Figura. 4) se identifican secuencias de cadenas de caracteres específicas de los lenguajes scripts que cumplen con un determinado patrón. Dicha secuencia corresponde a las sintaxis del lenguaje (Tabla III) y el patrón define la gramática que la reconoce.

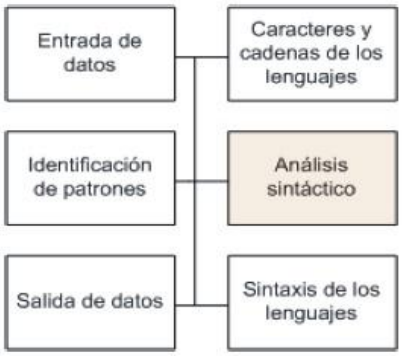

Figura. 4: Arquitectura del análisis sintáctico

2.1.3 Creación de patrones

Los patrones se crean combinando operadores, caracteres y cadenas de caracteres correspondientes a los lenguajes interpretados que se desean buscar [6] - [7]. Cuando se combinan caracteres y operadores se representan mayormente reglas rápidas que se utilizan en el análisis léxico. Cuando el patrón se compone de operadores y cadenas de caracteres se representan reglas secundarias que se emplean en el análisis sintáctico. Mientras más compleja sea la estructura del patrón que se define, más certera es la búsqueda y por consiguiente las clasificaciones derivadas. Esto influye directamente en el grado de especialización de los algoritmos y la cantidad a ejecutar por cada tipo detectado.

2.2 Estructura de la biblioteca

La biblioteca fue implementada en el lenguaje de programación C++ mediante algoritmos disponibles en el mismo. Esto permite su utilización indepen-dientemente del sistema operativo en que se instale el antivirus. Se aplicaron paradigmas actuales como la programación orientada a objetos (POO) y la programación genérica lo que permitió el encapsulamiento y la reutilización del código, así como la escalabilidad del mismo, en función de la posibilidad de futuras actualizaciones por la dinámica y evolución que traen consigo los programas antivirus. Consta de una estructura dividida en 5 partes principales (Figura. 5).

Figura. 5: Partes principales

También se diseñaron pruebas para cada parte de la biblioteca y se chequearon los posibles casos que pudieran generar resultados incorrectos en el funcionamiento esperado de cada sección. Se implementó un proyecto que ejecuta las pruebas lo que permite modificar el código fuente con una alta garantía de no afectar negativamente el resultado.

2.2.1 Definición de las reglas

En la definición de las reglas se permite el uso de patrones complejos que son muchos más certeros en la identificación pero más costosos y ralentizan el proceso. Para resolver este problema, la biblioteca posibilita asociar identificadores a los patrones y registrar las reglas como rápidas o secundarias en dependencia de la complejidad de los patrones (ver Figura. 6).

- Reglas rápidas: están definidas por patrones que reconocen secuencias de caracteres y algunas sintaxis simples.

- Reglas secundarias: están definidas por secuencias de cadenas de caracteres que re-conocen estructuras de lenguajes más complejas.

Para identificar los lenguajes scripts que presentan sintaxis comunes (Figura. 1), la biblioteca cuenta con la opción de definir cuáles reglas van a ser comunes y cuáles van a ser únicas, para que no se solapen y se puedan detectar ambas (Figura. 6).

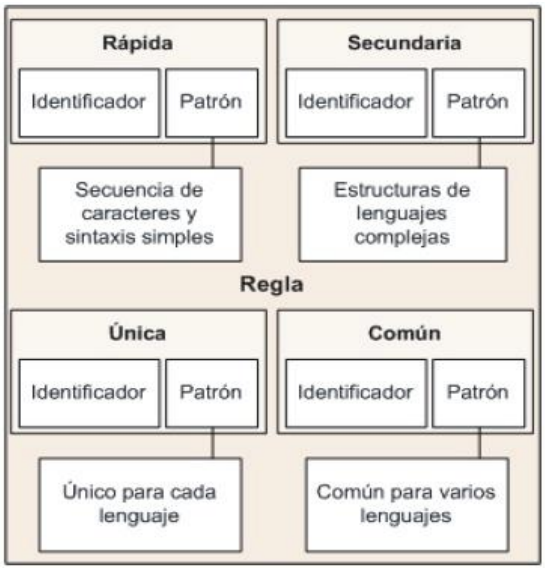

Figura. 6: Definición de regla

2.2.2 Exportación de las reglas

Este proceso sigue el orden lógico y estructurado de la inicialización de las reglas, para estandarizar y compatibilizar ambos procesos. Posibilita además la salva de todos los datos necesarios para que la biblioteca pueda iniciar las reglas y las clasificaciones. Permite incluir en la actualización del antivirus los cambios que sean necesarios en los patrones de búsqueda haciendo extensible hacia todos los productos las modificaciones o correcciones hechas al código.

2.2.3 Inicialización de las reglas

La inicialización de las reglas agrupa en un mismo proceso la asignación de los valores a todos los identificadores y los patrones que van a ser utilizados luego por el analizador. Estos valores son constantes para todas las búsquedas, por lo que la biblioteca ofrece la posibilidad de leerlos a partir del fichero que se exporta. Así se inicializa todo solo una vez al comienzo del sistema operativo cuando el antivirus carga los datos. De esta forma no es necesario ejecutar este proceso cada vez que se clasifique un fichero, lo que mejora considerablemente el consumo de recursos durante el análisis de grandes volúmenes de archivos.

2.2.4 Análisis de los ficheros

El análisis de los ficheros se divide en dos niveles (Figura. 7):

- Primer nivel: permite clasificar en una primera instancia los diferentes tipos a partir de búsquedas con las reglas rápidas y descartando archivos con información irrelevante.

- Segundo nivel: valida con búsquedas de reglas secundarias si el archivo corresponde realmente a la clasificación y descarta posibles falsos resultados.

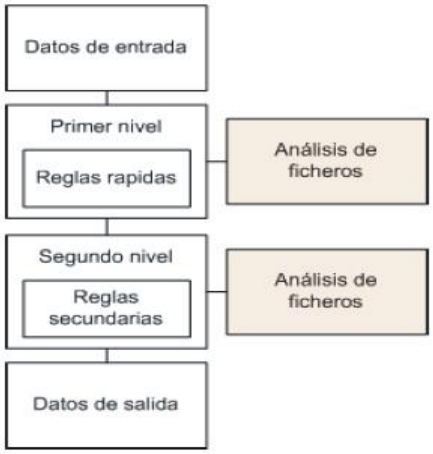

Figura. 7: Análisis de los ficheros

Además se emplean las reglas únicas y comunes para diferenciar las sintaxis, evitar el solapamiento de las reglas y no confundir u omitir alguna clasificación. Este diseño establece un equilibrio entre velocidad y certeza por lo que mejora considerablemente los tiempos de respuesta.

2.2.5 Clasificación del fichero

La clasificación del fichero obtiene los identificadores de cada patrón detectado durante el análisis del archivo y los almacena. Comprueba si existen datos repetidos y los elimina para evitar repetir búsquedas. Así el antivirus puede acceder a las clasificaciones y distribuir las secuencias de escaneo de los ficheros de texto según los tipos identificados y disminuyendo el volumen de firmas y algoritmos utilizados por cada archivos.

2.2.6 Resultados

Debido al impacto directo que genera el empleo de la biblioteca sobre SEGAV, se ejecutó un amplio proceso de pruebas, donde se escanearon grandes volúmenes de muestras con diferentes herramientas disponibles. Se compararon los resultados de tiempo y efectividad arrojados por las mismas antes y después de hacer uso de la biblioteca. Luego de varios análisis y optimizaciones, se logró alcanzar un equilibrio en los resultados válido para el antivirus. Las Figuras 8 y 9 ilustran gráficamente los valores obtenidos con una de las herramientas utilizadas:

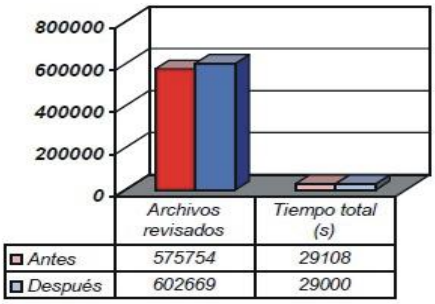

Figura. 8: Resultados de una búsqueda antes y después de utilizar la biblioteca

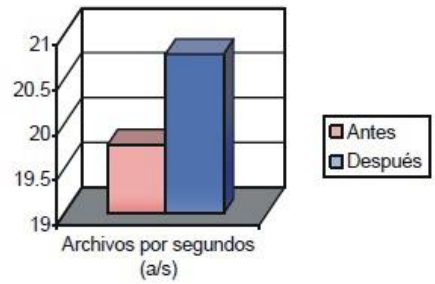

Figura. 9: Velocidad media de la búsqueda antes y después de utilizar la biblioteca

Se especializaron los algoritmos existentes para la detección de amenazas en ficheros de texto lo que fortaleció las estrategias de búsquedas. Se extendió la arquitectura de escáner aumentando la velocidad de escaneo del antivirus que permitió elevar el nivel de detección sin afectar el tiempo resultante.

Ahora los clientes del producto gozan de una mayor seguridad aunque no es visible a simple vista, además de una mejora notable de los tiempos de respuesta del antivirus.

CONCLUSIONES

Los resultados obtenidos demuestran la efectividad del análisis léxico y sintáctico en la identificación de lenguajes scripts para la clasificación de los diferentes ficheros de texto, además de la importancia de la biblioteca para el rendimiento del producto SE-GAV.

Quedan resueltas las problemáticas que genera todo el proceso de identificación de texto. El antivirus obtiene la especialización apropiada en las búsquedas a partir del uso de la biblioteca y se enriquece aún más su arquitectura de detección.

Una vez alcanzados con éxitos los objetivos trazados, es imprescindible retomar las investigaciones en el tema para encontrar nuevas problemáticas y generar buenas soluciones que garanticen una constante y creciente mejora en la identificación de texto en Segurmática Antivirus.

REFERENCIAS

1. Pérez, Alejandro Rivero. Emulador: más allá de la detección estática. Noviembre de 2010.

2. Text File Types. FileInfo.com. [En línea] 2012. [Citado el: 10 de Octubre de 2012.] http://www.fileinfo.com/filetypes/text.

3. Grossman, Jeremiah. XSS Attacks: Cross-site Scripting Exploits and Defense. [ed.] Syngress Media. 2007. 1597491543, 9781597491549.

4. Aumaille, Benjamin. JavaScript y VBScript. Barcelona : Ediciones ENI, 2000. ISBN:2-7460-0982-X.

5. Compiladores1, Segundo Semestre. Universidad de San Carlos. Guatemala : s.n., 2010.

6. Google. Acerca de las expresiones regulares. Página de ayuda de Google Analytics. [En línea] 8 de Marzo de 2008. [Citado el: 10 de Octubre de 2012.] http://support.google.com/analytics/bin/answer.py?hl=es&answer=1034324.

7. Mozilla Developer Network. Mozilla Developer Center. Escribir un patrón de expresión regular. [En línea] 24 de Mayo de 2007. [Citado el: 12 de Octubre de 2012.] Https://developer.mozilla.org/es/docs/Gu%C3%ADa_JavaScript_1.5/Escribir_un_patr%C3%B3n_de_expresi%C3%B3n_regular.