Detección de malware con modelo de lenguaje y su clasificación mediante SVM

Alex I. Valencia-Valencia[1], Sofía N. Galicia-Haro[2]

[1] UNAM, Posgrado en Ciencias e Ingeniería de la Computación, México

[2] UNAM, Facultad de Ciencias, México

letras_vivas@comunidad.unam.mx, sngh@fciencias.unam.mx

Resumen. La detección de malware representa una tarea cada vez más compleja, debido a las avanzadas técnicas de evasión a la detección; empleadas por los desarrolladores de malcode: desde el ofuscado de código, uso de polimorfismo, hasta variantes de malware que destruyen el disco duro si detectan que están siendo analizados. En la presente investigación se realiza un análisis dinámico de seis tipos de malware: Troyanos, Gusanos, Virus, Troyanos Espía, Puertas Traseras y Rootkits, además de un conjunto de Whiteware empleando el modelo de lenguaje de n-gramas en las llamadas al sistema del tipo WinAPI para cada muestra. Finalmente utilizamos el método de Máquinas de Soporte Vectorial (SVM) con kernel polinomial como algoritmo de aprendizaje automatizado para predecir la clasificación de malware, en nuestro caso con un rendimiento promedio de 70% y de 100% para el Whiteware, lo cual nos permite concluir que el modelo determina si la muestra es maliciosa y de esta manera puede llevar a cabo la detección de malware.

Palabras clave: Detección de malware, clasificación de malware, N-grama, SVM.

Language Model for Malware Detection and Classification based on SVM

Abstract. Malware analysis is a more difficult task each time, due to malware developers are taking care of avoiding and detection techniques, from obfuscation to destroying, hard drive (if malware detect that has been analyzed). In this paper we release a malware dynamic analysis of six types of malware: Trojans, Worms, Virus, Trojan-Spys, Backdoors y Rootkits, moreover a set of Whiteware using an n-grams language model in Win API calls for every sample. Finally, we used a Polynomial Kernel SVM to malware classification prediction, where we have obtained 70% of performance and malware classification and 100% for Whiteware classification in which case determine if is a malicious sample and in this form we can do malware detection.

Keywords: Malware detection, malware classification, N-gram, SVM.

1. Introducción

Al día de hoy las computadoras se han convertido en grandes herramientas de procesamiento y en este sentido los códigos maliciosos han evolucionado tanto en el daño causado cómo en las características que permiten ocultarlos del análisis. Respecto a las estadísticas la firma Symantec en su reporte de Amenazas de Seguridad en Internet de 2015 manifiesta que en el 2014 fueron introducidas 317 millones de variantes de malware. Mantenerse al día con la gran cantidad de variantes es desalentador para las organizaciones [1]. El método de análisis de malware es uno de los problemas clave en la técnica de detección de intrusos. En la literatura, los principales métodos de análisis de malware se basan en análisis de contenido estático y en el comportamiento dinámico [2]. En el análisis estático, las características se extraen del código binario de los programas y se usan para crear modelos que los describan. Los modelos se usan para distinguir entre malware y software legítimo [3]. Sin embargo el análisis estático falla en diferentes técnicas de ofuscación de código que se usa por los desarrolladores de códigos maliciosos y también en malcodes metamórficos y polimórficos [4].

En las técnicas de ofuscación el código fuente y el código binario se transforman de tal manera que tanto el proceso de decompilación sea más difícil como la lectura y análisis del mismo. Aunado a lo anterior el polimorfismo de malware consiste en cambiar la apariencia del malware a través de métodos de cifrado, de agregación de datos o eliminación de datos [5]. Debido a que el análisis dinámico de malware consiste en el estudio del mismo a través de su ejecución en un entorno controlado, su principal ventaja es que dicho análisis de comportamiento no puede ser ofuscado [6-7]. Sin embargo, hay algunas limitaciones en el análisis dinámico. Ya que cada muestra de malware debe ejecutarse dentro de un entorno seguro por un tiempo específico para monitorear el comportamiento. El proceso de monitoreo consume tiempo y debe asegurarse que la ejecución del malware no infecta la plataforma [8]. Los entornos seguros discrepan sólo un poco de un entorno de ejecución real y el malware puede comportarse de diferente manera en los dos entornos, causando una bitácora inexacta del comportamiento del malware [9]. Además de que algunas acciones del software malicioso se activan o ejecutan bajo ciertas condiciones (la fecha y hora del sistema o alguna entrada particular proporcionada por el usuario) puede no detectarse por el entorno virtual seguro [10].

Sin embargo, el análisis dinámico es un complemento necesario al enfoque estático como una medida preventiva de ofuscación de código. El análisis dinámico de malware basado en comportamiento consiste de una colección de muestras de malware, la ejecución de dichas muestras, entornos de monitoreo y la determinación de un modelo de características de comportamiento y análisis de comportamiento (clustering, clasificación, reconocimiento, etc.). La investigación de recolección de malware, ejecución de malware y métodos de monitoreo ha alcanzado un resultado maduro. Algunos sistemas de recolección y métodos se proponen basados en honeypot, como: Dionaea [11] y Kippo [12]. Los sistemas típicos tales como Anubis [13-14], CWSandbox [15], CuckooSandBox [16] ejecutan el malware en un ambiente controlado y monitorean el comportamiento del mismo (dicho método se nombra Sand-boxing), finalmente genera un reporte de comportamiento para cada muestra. El análisis profundo es necesario para revelar características del comportamiento y la detección de malware. Dos principales conceptos para el análisis automático de comportamiento se

han propuesto: clustering y clasificación [17-18]. A diferencia de muchos artículos en la literatura, que realizan el análisis de comportamiento con reportes de syscalls utilizando modelos de secuencias y/o los resultados del análisis estático realizados por la herramienta SandBox [4], [7], [13-19], en la presente investigación usamos las llamadas al sistema del tipo WinAPI y un modelo de lenguaje basado en n-gramas, para que el modelo considere la detección de malware tomamos en cuenta un conjunto de Whiteware o "malware benigno" con programas en la carpeta System32.

El artículo se organiza como sigue: en la segunda sección presentamos los elementos que se utilizan como datos para nuestra base de conocimiento la cual consiste de los 5-gramas de Win API calls de las muestras de malware. En la tercera sección describimos los parámetros con que se realizó el proceso de aprendizaje automatizado a través de SVM en Weka. En la cuarta sección primero describimos el comportamiento de los diferentes tipos de malware que se utilizaron en el experimento y posteriormente presentamos las estadísticas de los 5-gramas que se emplearon para el método de clasificación. En la quinta sección damos los resultados correspondientes obtenidos por Weka al realizar 10 iteraciones de la generación del modelo por medio de SVM.

Finalmente presentamos nuestras conclusiones y planteamos las líneas de trabajo futuro.

2. N-gramas en API Windows Calls

En los campos de probabilidad y lingüística computacional, un n-grama es una secuencia contigua de n elementos de una determinada secuencia de texto o de habla. Los elementos pueden ser fonemas, silabas, letras de acuerdo a su aplicación. Los n-gramas típicamente se recolectan de un corpus de texto o de habla. Cuando los elementos son palabras los n-gramas también pueden ser llamadas "shingles" [20].

Un n-grama de tamaño 1 se nombra "unigrama", el de tamaño 2 "bigrama", 3 "trigrama". Los n-gramas de mayor tamaño son nombrados por el valor de n, es decir; "4-grama", "5-grama", etc.

La siguiente tabla muestra diferentes ejemplos de secuencias y su modelo de secuencia de n-grama presentada en [21]:

Tabla 1. Ejemplos de n-gramas.

Campo	Unidad	Ejemplo de secuencia	Secuencia 1-grama	Secuencia 2-grama	Secuencia 3-grama
Secuencia de Proteinas	Amino ácido	…Cys-Gly-Leu-Ser-Trp …	…, Cys, Gly, Leu, Ser, Trp, …	…, Cys-Gly, Gly-Leu, Leu-Ser, Ser-Trp, …	…, Cys-Gly-Leu, Gly-Leu-Ser, Leu-Ser-Trp, …
Secuencia DNA	Pares base	…AGCTTCGA…	…, A, G, C, T, T, C, G, A, …	…, AG, GC, CT, TT, TC, CG, GA, …	…, AGC, GCT, CTT, TTC, TCG, CGA, …

Para el objetivo de este experimento utilizamos un análisis de 5-gramas en el reporte de WinAPI calls, el cual se llevó a cabo por la SandBox Cuckoo [16] dicho reporte está relacionado con los resultados del monitor de procesos de Windows, el cual se ejecuta durante el análisis de cada muestra de malware.

La API de Windows también llamada WinAPI es el conjunto núcleo de las interfaces de programación de aplicaciones (APIs) de Microsoft disponibles en los sistemas operativos de Microsoft Windows. Con dichas APIs de Windows pueden desarrollarse aplicaciones que se ejecuten exitosamente en todas las versiones de Windows mientras se aprovecha de las características y capacidades únicas de cada versión. Notar que esta fue formalmente llamada la API Win32. El nombre de Windows API refleja de manera más precisa su raíz en Windows de 16-bit y su soporte en Windows de 64-bit.

La siguiente lista es una referencia de contenido para la API de Windows de 32-bit tanto para aplicaciones de escritorio como de servidores, y dentro de cada categoría se encuentran las Win API calls que pudieron ser ejecutadas por cada muestra de malware [22]:

1. User Interface
2. Windows Environment (Shell)
3. User Input and Messaging
4. Data access and storage
5. Diagnostics
6. Graphics and Multimedia
7. Devices
8. System Services
9. Security and Identity
10. Application Installation and Servicing
11. System Admin and Management
12. Networking and Internet
13. Deprecated or legacy APIs

En la siguiente tabla se muestran ejemplos de los 5-gramas obtenidos de las muestras de malware:

Tabla 2. Ejemplos de 5-gramas de Win API calls obtenidos.

No.	5-grama de Win API Call
1	{NtOpenFile,NtCreateSection,NtCreateFile,NtQueryInformationFile,NtSetInformationFile}
2	{LdrGetProcedureAddress,NtQueryInformationFile,GetSystemMetrics,NtCreateSection,ZwMapViewOfSection}
3	{LdrGetProcedureAddress,LdrGetProcedureAddress,LdrGetProcedureAddress,NtFreeVirtualMemory,GetSystemMetrics}
4	{OpenServiceW,RegQueryValueExA,CreateThread,RegCloseKey,RegCloseKey}
5	{NtDelayExecution,ZwMapViewOfSection,DeleteFileA,NtDelayExecution,DeleteFileA}

3. Máquinas de soporte vectorial

Las Máquinas de Soporte Vectorial son un conjunto de algoritmos de aprendizaje supervisado desarrollados por Vladimir Vapnik y su equipo en los laboratorios AT&T. Estas construyen un hiperplano o conjunto de hiperplanos en un espacio de dimensionalidad muy alta (o incluso infinita) que puede ser utilizado en problemas de clasificación o regresión. Una buena separación entre las clases permitirá una clasificación correcta [23]. Dicho lo anterior y al obtener la similitud vectorial entre cada registro de la tabla binaria generada con los 5-gramas se determinó que sólo la clase Whiteware era la única diferente a todas las clases. Para llevar a cabo el experimento se utilizó el software Weka en su versión 3.7.13 y se realizaron 10 separaciones aleatorias de datos de entrenamiento y datos de prueba, además de los siguientes parámetros del clasificador SVM:

Tipo de experimento: División porcentual de conjunto de entrenamiento y prueba (con datos aleatorizados).
Porcentaje de entrenamiento: 70
Clasificador: Función SMO
Tamaño del lote: 1124
C: 1.0
Épsilon: 1.0E-12
Tipo de Filtro: Normalizar los datos de entrenamiento
Kernel: PolyKernel -E 1.0 -C 250007
Número de cifras decimales: 6

4. Diseño del experimento

En el experimento descargamos las 900 muestras de los 6 diferentes tipos de malware de malwr.com [24] considerando diferentes familias de cada tipo [25]. Dichos tipos de malware corresponden a los siguientes:

1. **Puertas traseras o Backdoors:** son un método externo en el proceso de autenticación o en otros controles de seguridad con el fin de acceder a un sistema de cómputo o a los datos contenidos en el mismo [8].
2. **Troyanos:** es un tipo de software malicioso que se empaqueta junto con una pieza útil del software o se hace pasar por una pieza de software útil. Una vez que el troyano es activado, que por lo general pasa desapercibido por el usuario, se libera una carga útil de ellos ya sea como un virus o una puerta trasera que puede permitir a un usuario acceder remotamente al sistema [6].
3. **Troyanos Espía:** software que obtiene información de una persona u organización sin su conocimiento y que puede enviar tal información a otra entidad sin el consentimiento del cliente, o que impone el control sobre una computadora sin el conocimiento del cliente [3].
4. **Gusanos:** Un programa usualmente pequeño que auto replica su contenido en sí mismo y que invade computadoras en una red y generalmente realiza acciones destructivas [10].

5. **Rootkits:** programas maliciosos que se ocultan en el sistema a través de modificaciones en las herramientas del sistema, filtrando activamente información de estado del sistema de los usuarios, enmascarando la presencia de archivos, servicios y canales de comunicación maliciosos [9].
6. **Virus:** Un programa que está usualmente oculto dentro de otro programa aparentemente inocuo y que produce copias de sí mismo e inserta otros programas y usualmente realiza una acción maliciosa (como destruir datos) [7].

Como parte del pre procesamiento de los datos se obtuvieron los 5-gramas por cada muestra de los seis tipos de malware incluyendo el whiteware consiguiendo un total de 85,013 5-gramas tal cómo se muestra en la Tabla 3, de los cuales 63,204 son únicos, y con estos últimos se generó una tabla binaria en representación de la presencia de cada 5-grama para cada muestra de clase, dicha tabla sirvió como entrada para el algoritmo de máquinas de soporte vectorial cuyos resultados se presentan en el siguiente punto. Para los diferentes tipos de malware se utilizaron 150 muestras de diferentes variantes del mismo y para Whiteware se utilizaron 224 muestras. Resultando una tabla de 63205x1124 campos (incluyendo la variable categórica de clase).

Tabla 3. Número de 5-gramas por cada tipo de malware.

Tipo de malware	Número de 5-gramas
Backdoors	6,343
Troyanos	15,378
Troyanos-Espía	11,621
Gusanos	33,888
Rootkits	606
Virus	12,212
Whiteware	4,965
Total	85,013

5. Resultados

Al realizar diez iteraciones con diferentes algoritmos de clasificación mostrados en la Tabla 4, puede apreciarse que las SVM con Kernel Polinomial se posicionan en el segundo mejor promedio de sensibilidad después de Random Forest y su tiempo de construcción del modelo es menor que este.

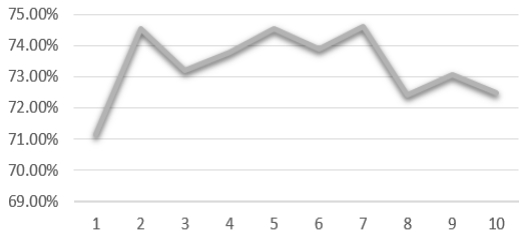

Fig. 1. Rendimiento del clasificador SVM.

Una vez determinado el algoritmo de Máquinas de Soporte Vectorial con Kernel Polinomial se ejecutó con los parámetros ya mencionados con el software Weka, sobre la misma base de conocimiento con los seis tipos de malware y whiteware, el rendimiento promedio fue de 70.16%, alcanzando un máximo de 75.65% y un mínimo de 71.13%.

Tabla 4. Tabla de comparación de sensibilidad entre algoritmos por tipo de malware.

	Logit Boost	**Naive Bayes Multinomial**	**Random Forest**	**SVM RBF Kernel**	**SVM Poly Kernel**	**SVM Poly Kernel Normalized**
Backdoor	44.2%	55.8%	58.1%	53.5%	67.4%	58.1%
Trojan	47.2%	44.4%	55.6%	50.0%	50.0%	55.6%
Trojan-Spy	72.1%	69.8%	72.1%	53.5%	67.4%	74.4%
Worm	37.0%	57.4%	57.4%	50.0%	48.1%	48.1%
Rootkit	93.2%	97.7%	90.9%	93.2%	95.5%	95.5%
Virus	44.2%	48.1%	65.4%	50.0%	57.7%	51.9%
Whiteware	100.0%	100.0%	100.0%	100.0%	100.0%	100.0%
Promedio	64.1%	69.4%	73.0%	66.2%	70.9%	70.1%
*Tiempo	251.02	0.85	91.12	56.46	57.73	61.28

*Tiempo significa Tiempo de construcción del modelo en segundos.

Es importante mencionar que la distribución promedio en el conjunto de prueba fue la siguiente: Backdoors: 42, Troyanos: 41, Troyanos Espía: 39, Gusanos: 46, Rootkits: 54 Virus: 45 y Whiteware: 65. Con dicha información se obtuvo la siguiente Tabla de resultados por clase, con el mismo número de iteraciones ya mencionadas, dónde puede apreciarse que el mejor rendimiento fue obtenido al clasificar Rootkits además del Whiteware y en último lugar los Gusanos.

Tabla 5. Rendimiento del clasificador por clase.

Clase	**TP**	**FP**	**Prec**	**F-M**	**MCC**	**ROC**	**PRC**
Backdoor	67%	5%	64%	66%	61%	87%	54%
Troyano	50%	6%	49%	49%	43%	83%	37%
Troyano Espía	67%	2%	81%	73%	70%	90%	65%
Gusano	48%	5%	65%	55%	49%	75%	43%
Rootkit	96%	3%	82%	88%	87%	97%	81%
Virus	58%	6%	64%	61%	54%	82%	47%
Whiteware	100%	6%	80%	89%	87%	97%	80%
Promedio	79%	5%	70%	70%	66%	87%	59%

Dónde: TP significa Tasa de Verdaderos Positivos, FP – Tasa de Falsos Positivos, Prec – Precisión, F-M - Valor-F, MCC – Coeficiente de Correlación de Matthews, ROC - Área ROC y PRC - Área PRC.

Como parte de la investigación se obtuvo la similitud vectorial entre cada registro de la tabla (similitud en términos de presencia de 5-gramas por muestra) utilizando la medida del coseno [26] algunos de los resultados se muestran en la Tabla 6.

Tabla 6. Tabla de porcentajes de similitud vectorial entre clases.

	Backdoor	**Troyano**	**Troyano Espía**	**Gusano**	**Rootkit**	**Virus**	**Whiteware**
Backdoor	20.7%	11.4%	22.9%	10.5%	18.2%	16.6 %	0%
Troyano	11.4%	9.5%	13.4%	8%	12.3%	10.1 %	0%
Troyano Espía	22.9%	13.4%	29.8%	12.5%	22.5%	19.6 %	0%
Gusano	10.5%	8%	12.5%	9%	0%	0%	0%
Rootkit	18.2%	12.3%	22.5%	0%	19.4%	0%	0%
Virus	16.6%	10.1%	19.6%	0%	0%	13.3 %	0%
Whiteware	0%	0%	0%	0%	0%	0%	1%
Promedio	14.3%	7.6%	12%	1.2%	10.3%	1.9%	0

6. Conclusiones y trabajo futuro

Cómo puede apreciarse en la gráfica, el modelo tiene un rendimiento que logra mantenerse en el 70%, y además cabe mencionar que el nivel de contribución de 5-gramas para el llenado de la tabla por cada tipo de malware tiene correlación con el rendimiento de su clasificación y ésta es inversamente proporcional; siendo que de los 85,013 5-gramas (incluyendo las repeticiones) los Gusanos tienen un mayor número de 5-gramas, es decir, la entropía de información proporcionada por su número de 5-gramas es muy alta y esto se muestra en la Tabla 6 que comparte similitud con dos tipos de malware, ergo tienen procesos con mayor cantidad de Win API calls lo cual hace más difícil su detección con este enfoque y análogamente los Rootkits con tan sólo 606 5-gramas nos dice que la variación entre cada muestra de este tipo es pequeña respecto a sus 5-gramas lo cual se manifiesta en la Tabla 6 ya que las muestras de este tipo comparten el tercer mayor coeficiente de similitud y por tanto más fácil de detectarse cómo puede apreciarse en la Tabla 4. Adicionalmente respecto a trabajos relacionados el costo computacional que representa obtener los 5-gramas, es mucho menor tanto en tiempo como en espacio respecto a otros algoritmos de minado de secuencias. Y por otro lado en este experimento se utiliza un conjunto de Whiteware a manera de estrategia de detección de malware, y debido a que la sensibilidad del Whiteware es del 100% sin importar el algoritmo esto nos permite decir que el enfoque propuesto detecta la presencia de malware.

Finalmente, como trabajo futuro creemos que el rendimiento del enfoque actual se puede mejorar aumentando el número de muestras y de fuentes de información por cada tipo de malware; y de esta manera la clasificación sería más precisa y no sólo se contemplaría el tipo de malware sino también la familia a la que pertenece. Por otro

lado se puede considerar otros experimentos de clases no balanceadas de malware, es decir con variantes de malware con una similitud vectorial menor.

Referencias

1. Shijo, P.V., Salim, A.: Integrated Static and Dynamic Analysis for Malware Detection. Procedia Comput. Sci., Vol. 46, No. Icict, pp. 804–811 (2015)
2. Moser, A., Kruegel, C., Kirda, E.: Limits of Static Analysis for Malware Detections.
3. Zhao, H., Xu, M., Zheng, N., Yao, J., Hou, Q.: Malicious executables classification based on behavioral factor analysis. In: IC4E, International Conference on e-Education, e-Business, e-Management and e-Learning, pp. 502–506 (2010)
4. Ahmadi, M., Sami, A., Rahimi, H., Yadegari, B.: Malware detection by behavioural sequential patterns. Comput. Fraud Secur., Vol. 2013, No. 8, pp. 11–19 (2013)
5. Shcherbina, V.S., Zakharov, V.A.: Using algebraic models of programs for detecting Metamorphic Malwares. r . I . Podlovchenko, n . N . Kuzyurin. Vol. 172, No. 5, pp. 740–751 (2011)
6. Wang, C., Pang, J., Zhao, R., Fu, W., Liu, X.: Malware detection based on suspicious behavior identification. Proceedings of the 1st International Workshop on Education Technology and Computer Science, ETCS, Vol. 2, pp. 198–202 (2009)
7. Tian, R., Islam, R., Batten, L., Versteeg, S.: Differentiating malware from cleanware using behavioural analysis. In: Proceedings of the 5th IEEE International Conference on Malicious and Unwanted Software, Malware, pp. 23–30 (2010)
8. Egele, M., Scholte, T., Kirda, E., Kruegel, C.: A survey on automated dynamic malware-analysis techniques and tools. ACM Computing Surveys, Vol. 44, No. 2. pp. 1–42 (2012)
9. Islam, R., Tian, R., Batten, L.M., Versteeg, S.: Classification of malware based on integrated static and dynamic features. Journal of Network and Computer Applications, Vol. 36, No. 2. pp. 646–656 (2013)
10. Jonathan, P., Vázquez, B.: PoC : Captura de malware con el honeypot Dionaea - Parte I. Rev. .Seguridad UNAM CERT (2015)
11. Sochor, T., Zuzcak, M.: Study of Internet Threats and Attack Methods Using Honeypots and Honeynets. Computer Networks SE-12, Vol. 431, Kwiecień, A., Gaj, P., Stera, P. Eds. Springer International Publishing, pp. 118–127 (2014)
12. Wa, R., Hunt, G., Brubacher, D.: Detours: Binary Interception of Win32 Functions. Proc. 3rd USENIX Wind. NT Symp., pp. 135–143 (1999)
13. Bayer, U., Kruegel, C., Kirda, E.: TTAnalyze : A Tool for Analyzing Malware.
14. Bayer, U., Moser, A., Kruegel, C., Kirda, E.: Dynamic analysis of malicious code. pp. 67–77 (2006)
15. Willems, G., Holz, T., Freiling, F.: Toward automated dynamic malware analysis using CWSandbox. IEEE Secur. Priv., Vol. 5, No. 2, pp. 32–39 (2007)
16. Guarnieri, C.: CuckooSandbox. [Online]. Available: http://www.cuckoosandbox.org/about.html. [Accessed: 01-Jun-2015].
17. Gheorghescu, M.: An Automated Virus Classification System. Virus Bull. Conf., no. October, pp. 294–300 (2005)
18. Rieck, K., Trinius, P., Willems, C., Holz, T.: Automatic Analysis of Malware Behavior using Machine Learning. pp. 1–30 (2011)
19. Wysopal, C., Shields, T.: Static Detection of Application Backdoors Detecting both malicious software behavior and malicious, Info.
20. Syntactic Clustering of the Web. [Online]. Available: http://www.hpl.hp.com/techreports/Compaq-DEC/SRC-TN-1997-015.pdf. [Accessed: 15-Dec-2015].

21. Sidorov, G.: Syntactic Dependency Based N-grams in Rule Based Automatic English as Second Language Grammar Correction. Int. J. Comput. Linguist. Appl., Vol. 4, No. 2, pp. 169–188 (2013)
22. Windows API Index (Windows). [Online]. Available: https://msdn.microsoft.com/en-us/library/windows/desktop/ff818516(v=vs.85).aspx. [Accessed: 16-Dec-2015]
23. Support-Vector Networks. [Online]. Available: http://image.diku.dk/imagecanon/material/cortes_vapnik95.pdf. [Accessed: 15-Dec-2015].
24. Cuckoo Sandbox: Malware repository. [Online]. Available: https://malwr.com/. [Accessed: 02-Jun-2015]
25. Bontchev, V., Software, F., Thverholt, I.: Current Status of the CARO Malware Naming Scheme.
26. Narouei, M., Ahmadi, M., Giacinto, G., Takabi, H., Sami, A.: DLLMiner : structural mining for malware detection. No. April, pp. 3311–3322 (2015)