Analisis Perbandingan Deteksi *Trojan* Menggunakan Metode *Static-Dynamic* dengan Metode *Behaviour* Pada Sistem Operasi Windows

Dania M.P.M[1], Parman Sukarno[2], Erwid Mushofa Jadied[3]

[1,2,3]Fakultas Informatika, Universitas Telkom, Bandung
[1]daniamartha@students.telkomuniversity.ac.id, [2]psukarno@telkomuniversity.ac.id,
[3]jadied@telkomuniversity.ac.id

Abstrak

Windows adalah sistem operasi yang paling banyak digunakan oleh pengguna komputer. Karenanya, pengguna Windows menjadi sasaran utama dalam penyebaran *malware* terutama *trojan. Trojan* adalah *malware* yang menyebar pada komputer korban yang mempunyai sifat merusak hingga merugikan. Oleh karena itu banyak penelitian mengenai metode untuk mendeteksi *trojan.* Dari penelitian yang telah dilakukan sebelumnya, metode *Static-Dynamic* lebih sering digunakan. Namun metode tersebut belum cukup akurat dalam hal mendeteksi *trojan* dibandingkan dengan metode *Behaviour.* Penelitian ini mengimplementasikan metode *Static-Dynamic* dan metode *Behaviour* dengan melakukan percobaan menggunakan *threshold* yang berbeda pada pengujiannya. Didapatkan kesimpulan bahwa setiap perubahan pada *threshold* akan berpengaruh pada tingkat akurasi dari setiap metode. Dari dua belas kali pengujian metode *Static-Dynamic* mempunyai rata-rata akurasi sebesar 32,70%. Sedangkan metode *Behaviour* dengan dua puluh tiga kali percobaan mempunyai rata-rata akurasi sebesar 32,54%.

Kata kunci : *Trojan,* Metode *Static,* Metode *Dynamic,* Metode *Behaviour*

Abstract

Windows is the operating system that is most widely used by computer users. Therefore, Windows' users are the main target of the malware spreading, especially trojan. Trojans are malware that spreads on a victim's computer that destructive and harmful users' properties. There is a lot of research about methods for detecting trojans. From previous research, the Static-Dynamic method is used more often. But this method is not accurate enough in detecting trojan compared to the Behavior method. This study implements the Static-Dynamic method and the Behavior method. It was concluded that every change of *threshold* will have different result. Out of twelve threshold of Static-Dynamic method, the average accuracy is 32,70%. And Behaviour method with twenty three threshold has 32,54% of average accuracy.

Keywords: Trojan, Static Method, Dynamic Method, Behaviour Method

1. Pendahuluan

Latar Belakang

Pada era teknologi sekarang ini, hampir seluruh orang bisa mengakses internet dengan mudah. Kebanyakan komputer yang digunakan berbasis Windows sebagai sistem operasinya [1]. Hal itu membuat Windows menjadi sasaran utama penyebaran *malware. Malware* adalah program yang dibuat dengan tujuan untuk merusak, mencuri dan melakukan aktifitas illegal pada komputer korban. Salah satu jenis *malware* adalah *trojan. Trojan* adalah *malware* yang bisa dengan mudah menyebar tanpa diketahui oleh pemilik komputer. Kebanyakan pengguna secara tidak sengaja mengeksekusi *trojan* tanpa mengetahui dampaknya. *Trojan* biasanya menyamar menjadi aplikasi umum yang sering digunakan, maka dari itu banyak yang tidak menyadari bahwa mereka telah menjalankan *trojan.*

Dari permasalahan diatas, telah dilakukan penelitian tentang metode deteksi *trojan* pada sistem operasi Windows [2]. Pada penelitian tersebut menggunakan metode *Static-Dynamic* untuk deteksinya. Pada prosesnya, metode *Static* dilakukan sebelum *trojan* dieksekusi, dan metode *Dynamic* dilakukan setelah *trojan* dieksekusi. Namun, metode tersebut pada beberapa bagian masih menggunakan analisis manual dan memiliki tingkat akurasi yang rendah.

Perumusan Masalah

Berdasarkan latar belakang yang telah dijelaskan diatas, perumusan masalah yang diambil adalah bagaimana metode *Behaviour* mempunyai akurasi lebih tinggi dalam mendeteksi *trojan* dibandingkan dengan metode *Static-Dynamic?*

Adapun batasan masalah untuk penelitian ini agar permasalahan yang diambil tidak meluas dan untuk menyesuaikan kebutuhan, yaitu :

1. *Trojan* yang digunakan sebagai sampel adalah *Bozok*, *njRAT*, *BlackShades*, *DarkComet* dan sebuah *DataSet.*
2. Perspektif yang digunakan adalah *victim side.*

Tujuan

Tujuan pada penelitian ini adalah menemukan karakteristik dari sampel *trojan* dengan melakukan implementasi metode *Static*, mengetahui siklus hidup *trojan* pada komputer dengan melakukan implementasi metode *Dynamic*, membandingkan tingkat akurasi metode otomatisasi *Static-Dynamic* dengan metode *Behaviour*.

Organisasi Tulisan

Penulisan tugas akhir ini tersusun dalam beberapa bagian, yaitu :

1. **Pendahuluan** Menjelaskan latar belakang, perumusan masalah, batasan masalah dan tujuan dari topik yang diambil.
2. **Studi Terkait** Berisi penelitian sebelumnya yang digunakan sebagai literatur dalam pembuatan tugas akhir ini.
3. **Skenario Pengujian** Menjelaskan skenario pengujian yang dilakukan dalam implementasi untuk dijadikan perbandingan pada hasil analisis.
4. **Evaluasi** Menjelaskan hasil pengujian yang dilakukan dan analisis terhadap hasil dari pengujian skenario pengujian.
5. **Kesimpulan** Hasil akhir dari penelitian serta saran untuk penelitian selanjutnya.

2. Studi Terkait

Pada penelitian dengan judul *Malware Analysis Pada Sistem Operasi Windows untuk Mendeteksi Trojan* [2], jurnal tersebut menggunakan metode *Static* dan metode *Dynamic* untuk melakukan analisis terhadap sampel *trojan* yang komputer penggunanya berbasis Windows. Namun tingkat akurasi dari kedua metode tersebut tidak bisa dinyatakan *valid* karena hanya menggunakan 4 buah file *trojan* sebagai sampelnya.

Pada penelitian dengan judul *A Trojan Horse Detection Technology Based on Behavior Analysis* [3], jurnal tersebut meneliti tentang bagaimana metode *Behaviour* digunakan untuk mendeteksi *trojan.* Metode ini mempelajari sifat sebuah *trojan* secara garis besar, bukan hanya *signature* tertentu.

3. Skenario Pengujian

Pada penelitian ini menggunakan Windows sebagai sistem operasi karena pengguna komputer lebih banyak menggunakan sistem operasi ini dibandingan dengan yang lain [1].

Dilakukan beberapa skenario pengujian untuk mencapai tujuan, yaitu :

3.1 Implementasi Metode *Static*

Metode *Static* adalah metode yang digunakan untuk menentukan sebuah file yang dicurigai sebagai *trojan* dengan cara menganalisis file tersebut sebelum dieksekusi. Metode ini digunakan untuk menemukan karakteristik dari file. Pada metode ini terdapat empat buah sampel yang digunakan yaitu *Bozok* [4], *njRAT* [5], *BlackShades* [6] dan *DarkComet* [7]. Tahapan ini dilakukan untuk mengimplementasikan ulang penelitian sebelumnya [2] dan memastikan kebenaran dari penelitian tersebut.

Tool yang pertama digunakan adalah *OllyDbg* [8]. *Tool* ini digunakan untuk melihat *dll* apa saja yang terdapat dalam sampel. Selanjutnya *tool* yang digunakan adalah *DependencyWalker 2.2* [9]. Lebih rinci dari *OllyDbg*, *tool* ini juga menampilkan *function* apa saja yang berada dalam setiap *dll.* Dari situ, bisa dilakukan analisis *function* mana saja yang dianggap berbahaya dilihat dari parameter [14].

3.2 Implementasi Metode *Dynamic*

Metode *Dynamic* adalah metode yang digunakan untuk mempelajari tingkah laku dari *trojan* ketika file tersebut sudah dieksekusi dalam sistem. Metode ini juga menggunakan sampel yang sama. *Tool* yang pertama digunakan adalah *Process Explorer v16.21* [10]. *Tool* ini digunakan untuk melihat *access* dari sampel. Selain itu disaat bersamaan dilihat bagian *Company Name* dan *Description* dari sampel yang sedang dieksekusi. File tersebut bisa dianggap mencurigakan jika tidak berisikan keterangan *official.* Selanjutnya menggunakan *Process Monitor v3.50* [11] untuk melihat *path* yang ditinggalkan dari file sampel setelah dieksekusi.

3.3 Otomatisasi Metode *Static-Dynamic*

Pada bagian ini, terdapat perbedaan dengan penelitian yang sebelumnya [2] dalam hal penerapan metode *Static-Dynamic.* Metode *Static-Dynamic* manual tidak bisa digunakan untuk mendeteksi akurasi yang akan diterapkan pada sampel berupa *DataSet* [12]. Walaupun mengacu pada penelitian sebelumnya, jumlah sampel yang terlalu sedikit tidak bisa dinyatakan *valid*. Maka dari itu, untuk menyederhanakan metode tersebut dibuatlah otomatisasi metode *Static-Dynamic.* Pada bagian ini dibuat aplikasi sederhana dengan menggunakan *python* sebagai bahasa pemrogramannya. Aplikasi ini dibuat agar bisa membaca *DataSet* sebagai sampel, dan akhirnya akan mengeluarkan *output* berupa adalah jumlah TP, FP, TN, FN, Presisi, *Recall* dan akurasi. Pada *DataSet* ditambahkan *countsd* pada akhir tabel untuk melihat dari sebuah sampel ada berapa buah *function* yang muncul dari 13 parameter [14]. *Threshold* adalah jumlah kemungkinan minimal sebuah *function* muncul hingga sebuah sampel bisa dikatakan *trojan* atau tidak.

3.4 Implementasi Metode *Behaviour*

Metode *Behaviour* adalah metode yang menggunakan sifat dari *trojan* sebagai acuan untuk memutuskan sebuah *file* bisa dianggap *trojan* atau tidak [3]. Sama seperti bagian otomatisasi metode *Static-Dynamic*, bagian ini dibuat aplikasi sederhana dengan konsep metode *Behaviour* untuk mendeteksi *trojan. Output* dari aplikasi ini adalah jumlah TP, FP, TN, FN, Presisi, *Recall* dan akurasi. Pada *DataSet*, ditambahkan *countsd* pada akhir tabel untuk melihat dari sebuah sampel ada berapa buah *function* yang muncul dari 24 parameter. *Threshold* adalah jumlah kemungkinan minimal sebuah *function* muncul hingga sebuah sampel bisa dikatakan *trojan* atau tidak.

3.5 *DataSet*

Pada pengujian bagian ini menggunakan file *DataSet* sebagai sampel. *DataSet* tersebut mempunyai 10.868 total sampel. ***Class*** pada *DataSet* mempunyai angka 1-9, yaitu ***Ramit, Lollipop, Kelihos_ver3, Vundo, Simda, Tracur, Kelihos_ver1, Obfuscator.ACY, Gatak.*** *Malware* yang termasuk *trojan* sebanyak 7.230 dari total sampel. Yang tidak termasuk sebagai *trojan* sebanyak 3.638 yaitu sampel dari ***Lollipop*** dan ***Obfuscator.ACY*** [15]. Mengacu pada *function* yang dianggap sebagai sebuah *trojan behaviour's treat* [14], berikut adalah beberapa function yang ada pada data sampel :

Tabel 1. Sampel *DataSet*

RegCreateKeyExW	CreateDirectoryW	InternetConnectA	SHGetSpecialFolderPathA	HttpOpenRequestA	Class
0	0	0	0	0	2
0	0	0	0	0	5
0	0	0	0	0	3
3	0	0	0	0	1

3.6 Perbandingan Tingkat Akurasi

Dari hasil *output* kedua aplikasi tersebut akan didapatkan jumlah TP, FP, TN, FN, Presisi, *Recall* dan akurasi yang berbeda tergantung dengan metode yang digunakan. Parameter tersebut akan digunakan untuk analisis perbandingan dalam menentukan metode mana yang lebih baik digunakan dalam mendeteksi *trojan.* Berikut adalah rumus yang digunakan :

		Nilai sebenarnya	
		TRUE	FALSE
Nilai prediksi	TRUE	TP (True Positive) *Corect result*	FP (False Positive) *Unexpected result*
	FALSE	FN (False Negative) *Missing result*	TN (True Negative) *Corect absence of result*

$$precision = \frac{TP}{TP + FP}$$

$$recall = \frac{TP}{TP + FN}$$

$$accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

Gambar 1. Rumus perhitungan presisi, recall dan akurasi

4. Evaluasi

4.1 Hasil Pengujian

Berikut adalah hasil pengujian berdasarkan skenario yang telah dijelaskan diatas :

4.1.1 Hasil Pengujian Metode *Static*

Hasil pengujian metode *Static* didapatkan beberapa *signature trojan* dilihat dari *function* mencurigakan yang berada dalam *dll*. Hasil dari pengujian ini nantinya akan dijadikan *rules* pada penerapan otomatisasi metode *Static-Dynamic*. *Rules* yang akan digunakan diambil dari *function* yang tertera dari tabel berikut :

Tabel 2. *Output* pengujian metode *Static*

No	Fungsionalitas	DLL	Function	DarkComet	Bozok	Njartz	BlackShades
1	Infeksi	Kernel32	CreateMutex			✓	
			CopyFile				
			GetModuleFileName		✓	✓	✓
			GetTempPath				
			CreateFile	✓	✓	✓	
			CreateThread				
			GetWindowsDirectory	✓		✓	
			CreateDirectory				
			CreateProcess				✓
			WriteFile	✓	✓	✓	✓
			Sleep	✓	✓	✓	✓
		Shell32	SHGetFolderPathA		✓		
			ShellExecute		✓		
			WinExec				
			ShGetSpecialFolderPath		✓		
2	Registry Activity	AdvAPI32	RegCloseKey	✓	✓	✓	✓
			RegSetValueEx	✓	✓		✓
			RegDeleteKeyEx	✓	✓		✓
			RegQueryValueEx	✓	✓	✓	✓
			RegCreateKeyEx	✓	✓		✓
			RegDeleteValue	✓	✓		✓
			RegOpenKeyEx	✓	✓		✓
3	Komunikasi C&C	WiniNet	HttpOpenRequest				
			InternetReadFile	✓			✓
			InternetOpenURL	✓			✓

			HttpSendRequest				
			InternetConnect	✓			✓
			InternetOpenURL	✓			✓
4	Download Execute	Shell32	SHGetFolderPathA		✓		
			ShellExecute	✓	✓		✓
			WinExec				
			ShGetSpecialFolderPath		✓		
		URLMON	UrlDownloadToFile				
		Kernel32	CreateDirectory	✓			✓
			GetWindowsDirectory	✓		✓	✓
			GetTempPath				
5	Visit Website	Shell32	ShellExecute	✓	✓		✓
6	KeyLogger	USER32	GetKeyState				
			GetKeyboardType				
			MapVirtualKey				
7	Update	AdvApi32	RegCreateKeyEx	✓	✓		✓
			RegDeleteKey	✓	✓		✓
		Kernel32	GetTempPath				
			GetWindows Directory	✓		✓	
		Shell32	ShGetFolderPathA		✓		
			ShellExecute	✓	✓		✓
			WinExec				
			ShGetSpecialFolderPath		✓		
		URLMON	UrlDownloadToFile				
8	Uninstal	AdvApi32	RegCreateKeyEx				
			RegDeleteKey	✓	✓		✓
		Kernel32	ExitProcess	✓	✓	✓	✓
			TerminateProcess				
			GetWindowsDirectory	✓		✓	

4.1.2 Hasil Pengujian Metode *Dynamic*

Hasil pengujian metode *Dynamic* dengan menggunakan beberapa *tools* memperlihatkan *access* apa saja yang dimiliki oleh *trojan.* Pada file *non-trojan*, *access* yang dimiliki hanya *read* dan *write* saja. Sedangkan seluruh file sampel memiliki hak *access* penuh pada sistem seperti *modify, read, write* dan *read & execute.* Virustotal adalah salah satu analisis yang ada pada *tool* untuk dihubungkan dengan *website* virustotal, dimana dari 66 hingga 68 *antivirus* yang ada, angka disebelah kiri adalah cuman file tersebut dideteksi sebagai *trojan* dari total *antivirus.*

Tabel 3. *Output* pengujian metode *Dynamic*

No	Access	Bozok	DarkComet	BlackShades	Njratz
1	Modify	✓	✓	✓	✓
2	Read	✓	✓	✓	✓
3	Write	✓	✓	✓	✓
4	Read & Execute	✓	✓	✓	✓
5	VirusTotal	26/66	53/68	46/67	48/67

4.1.3 Hasil Pengujian Otomatisasi Metode *Static-Dynamic*

Berikut adalah hasil pengujian menggunakan metode otomatisasi *Static-Dynamic* :

Tabel 4. *Output* pengujian otomatisasi metode *Static-Dynamic*

Threshold	Trojan Menurut Aplikasi	TP	FP	TN	FN	Akurasi	Presisi	Recall
1	8277	4726	3551	155	2436	44,91%	57%	66%
2	5998	2582	3416	290	4580	26,43%	43%	36%
3	4304	1549	2755	951	5613	23%	36%	22%
4	2408	1008	1400	2306	6154	30,49%	42%	14%
5	1296	3553	943	2763	6809	28,67%	27%	5%
6	556	218	338	3363	6944	33%	39%	3%
7	249	150	99	3607	7012	34,57%	60%	2%
8	76	61	15	3691	7101	34,52%	80%	1%
9	51	43	8	3698	7119	34,42%	84%	1%
10	28	23	5	3701	7139	34,27%	82%	0%
11	11	8	3	3703	7154	34,15%	73%	0%
12	2	1	1	3705	7161	34%	50%	0%

4.1.4 Hasil Pengujian Metode *Behaviour*

Berikut adalah hasil pengujian menggunakan metode *Behaviour* :

Tabel 5. *Output* pengujian otomatisasi metode *Behaviour*

Threshold	Trojan Menurut Aplikasi	TP	FP	TN	FN	Akurasi	Presisi	Recall
1	9577	6064	3513	193	1098	57,57 %	63%	85%
2	7362	4054	3308	398	3108	40,96 %	55%	57%
3	4809	2226	2583	1123	4936	30,82 %	46%	31%
4	3867	1902	1965	1741	5260	33,52 %	49%	27%
5	2902	1629	1273	2433	5533	37,38 %	56%	23%
6	2172	1019	1153	2553	6143	32,87 %	47%	14%
7	1884	808	1076	2630	6354	31,63 %	43%	11%
8	1575	593	982	2724	6569	30,52 %	38%	8%
9	1333	425	908	2798	6737	29,66 %	32%	6%
10	1108	279	829	2877	6883	29,4 %	25%	4%
11	937	188	749	2957	6974	28,94 %	20%	3%

12	763	101	662	3044	7061	28,94 %	13%	1%
13	677	77	600	3106	7085	29,29 %	11%	1%
14	181	52	129	3577	7110	33,39 %	29%	1%
15	114	34	80	3626	7128	33,68 %	30%	0%
16	71	27	44	3662	7135	33,94 %	38%	0%
17	49	23	26	3680	7139	34,07 %	47%	0%
18	28	21	7	3699	7141	34,23 %	75%	0%
19	14	13	1	3705	7149	34,21 %	93%	0%
20	9	9	0	3706	7153	34,18 %	100%	0%
21	2	2	0	3706	7160	34,12 %	100%	0%
22	1	1	0	3706	7161	34,11 %	100%	0%
23	1	1	0	3706	7161	34,12 %	100%	0%

4.2 Analisis Hasil Pengujian

TP adalah *True Positive*, dimana jumlah tersebut adalah jumlah *trojan* menurut *DataSet* dan memang terdeteksi sebagai *trojan* oleh aplikasi. *Behaviour* mempunyai rata-rata TP 850 sedangkan *Static-Dynamic* sebanyak 1160. Karena metode *Behaviour* mengacu pada parameter yang *direct* mengarah dan mengekseskusi sesuatu di sistem, maka parameter menjadi lebih *strict.* Maka dari itu metode *Behaviour* mempunyai TP yang lebih sedikit dibandingkan metode *Static-Dynamic* dengan parameter umum.

FP adalah *False Positive*, dimana jumlah tersebut adalah jumlah *non-trojan* menurut *DataSet* namun terdeteksi sebagai *trojan* oleh aplikasi. *Behaviour* mempunyai rata-rata FP 865 sedangkan *Static-Dynamic* sebanyak 1045. Karena parameter metode *Static-Dynamic* mempunyai ruang lingkup yang umum, jumlah FPnya lebih banyak dibandingkan dengan metode *Behaviour.*

TN adalah *True Negative*, dimana jumlah tersebut adalah jumlah *non-trojan* menurut *DataSet* dan memang terdeteksi sebagai *non-trojan* oleh aplikasi. *Behaviour* mempunyai rata-rata TN 2841 sedangkan *Static-Dynamic* sebanyak 2661. Metode *Static-Dynamic* dan metode *Behaviour* semakin tinggi *threshold* maka akan semakin besar TNnya.

FN adalah *False Negative*, dimana jumlah tersebut adalah jumlah *trojan* namun terdeteksi sebagai *non-trojan* oleh aplikasi. *Behaviour* mempunyai rata-rata FN 6312 sedangkan *Static-Dynamic* sebanyak 6268. Jika *threshold* semakin tinggi, maka semakin *strict* parameternya. Maka dari itu FN dari metode *Behaviour* lebih besar daripada metode *Static-Dynamic.*

Dari empat parameter diatas, menghasilkan *output* dari presisi, *recall* dan akurasi. Presisi adalah ketepatan jumlah *trojan* yang ditangkap oleh aplikasi dibandingkan dengan jumlah *trojan* sebenarnya. Rata-rata presisi dari metode *Static-Dynamic* adalah 52,91%. Sedangkan untuk metode *Behaviour,* rata-rata presisinya adalah 52,61%.

Recall adalah ketepatan trojan yang ditangkap oleh aplikasi. Rata-rata *recall* dari metode *Static-Dynamic* adalah 12,5%. Untuk metode *Behaviour* adalah 11,83%.

Akurasi adalah hasil pengukuran nilai sesungguhnya. Rata-rata akurasi dari metode *Behaviour* adalah 32,54%. Untuk metode *Static-Dynamic* adalah 32,70%.

5. Kesimpulan

Dari hasil pengujian diatas bisa ditarik kesimpulan, penerapan metode *Static-Dynamic* manual dengan empat sampel akan menghasilkan *rules* yang nantinya akan digunakan dalam pembuatan aplikasi otomatisasi *Static-Dynamic* dan dalam aplikasi metode *Behaviour.* Semakin tinggi sebuah *threshold,* maka akan semakin kecil *recall*nya dan FN akan semakin besar karena sampel yang *trojan* namun dianggap *non-trojan* karena parameter yang terlalu *strict.*

Presisi dari kedua metode akan semakin kecil jika *threshold* semakin besar, maka TP akan semakin kecil. Karena parameter yang terlalu *strict* membuat sebuah *trojan* yang memang terdata sebagai *trojan* di *DataSet* akan terdeteksi sebagai *non-trojan* pada aplikasi. Akurasi dilihat dari jumlah kemungkinan benar yaitu TP dan TN berbanding dengan jumlah data. Jika *threshold* semakin naik, jumlah TP akan semakin turun sedangkan jumlah TN akan semakin naik. Oleh karena itu tingkat akurasi kedua metode tersebut akan menurun hingga pertengahan *threshold* dan akan kembali naik ketika *threshold* semakin tinggi. Menentukan tinggi *threshold* dalam pendeteksian akan sangat penting untuk menentukan TP, TN, FN, FP, akurasi, presisi, dan *recall* dari sebuah metode.

Saran yang diberikan untuk penelitian selanjutnya, sampel yang digunakan tidak hanya *trojan* namun seluruh jenis *malware* dengan versi terbaru agar cakupannya luas. Sebaiknya juga dilakukan penelitian dengan sistem operasi yang berbeda selain Windows.