LAPORAN AKHIR
PENELITIAN

PENGEMBANGAN APLIKASI PENGAMATAN PERILAKU *MALWARE* DAN PEMBANGKITAN *SCRIPT* PENANGANAN SECARA OTOMATIS

PENELITIAN LABORATORIUM/LAPANGAN

Oleh:

Kusworo Anindito, S.T., M.T.

Fakultas Teknologi Industri

Universitas Atma Jaya Yogyakarta

2009

LEMBAR PENGESAHAN

1.	**a. Judul Penelitian**	:	PENGEMBANGAN APLIKASI PENGAMATAN PERILAKU *MALWARE* DAN PEMBANGKITAN *SCRIPT* PENANGANAN SECARA OTOMATIS
	b. Macam Penelitian	:	Laboratorium
2.	**Peneliti**		
	a. Nama	:	Kusworo Anindito, S.T., M.T.
	b. Jenis Kelamin	:	Laki-laki
	c. Usia saat pengajuan proposal	:	35 tahun 6 bulan
	d. Jabatan Akademik/Gol	:	Lektor / IIIc
	e. Fakultas / Program Studi	:	Teknologi Industri / Teknik Informatika
3.	**Jumlah Peneliti**	:	1 (satu) orang
4.	**Lokasi Penelitian**	:	Yogyakarta
5.	**Jangka Waktu Penelitian**	:	6 (enam) bulan
6.	**Biaya yang disetujui**	:	2.850.000,- (Dua juta delapan ratus lima puluh ribu rupiah)

Yogyakarta, 19 Januari 2009
Ketua Peneliti,

Kusworo Anindito, S.T., M.T.

Mengetahui,

Konsultan

Prof. Ir. F. Soesianto, B.Sc.E, Ph.D

Kepala Lab Jaringan Komputer

Th. Devi Indriasari, S.T., M.Sc.

Dekan FTI UAJY,

Paulus Mudjihartono, S.T, M.T.

Ketua LPPM UAJY,

14 FEB 2009

Ir. B. Kristyanto M.Eng., Ph.D.

DAFTAR ISI

INTISARI

Malware **merupakan ancaman besar dalam dunia komputer saat ini. Berbagai jenis** ***malware,*** **seperti virus,** ***worm, trojan horse, rootkit,*** **terus berkembang. Terhubungnya komputer-komputer pada jaringan lokal dan Internet juga membuat penyebaran** ***malware*** **semakin cepat dan luas. Kemunculan** ***malware*** **baru ini membuat pendeteksian dan penanganannya menjadi semakin sulit dilakukan. Antivirus yang terinstal sering kali belum mampu mendeteksi dan menangani** ***malware*** **terbaru sampai** ***virus definition/malware signature*****-nya terbarukan.**

Penelitian ini mengamati perubahan-perubahan yang dilakukan oleh ***malware*** **dan membangkitkan** ***script*** **untuk menangani** ***malware*** **tersebut dengan mematikan proses** ***malware,*** **menghapus** ***file*** **induk** ***malware*** **dan memperbaiki** ***windows registry*****.**

BAB 1. PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi jaringan, khususnya Internet, memberikan banyak kemudahan dan kegunaan, tetapi juga telah memunculkan suatu permasalahan baru, yaitu membantu penyebaran *malware*/virus. Serangan *malware*/virus baik yang melalui jaringan lokal maupun Internet serta melalui media penyimpanan sekunder seperti flashdisk mengalami peningkatan yang signifikan. Hal ini tentu saja akan membawa kerugian bagi pengguna komputer, terlebih bagi perusahaan yang menjalankan proses bisnisnya dengan dibantu oleh komputer (Yurnalis, 2008). Kondisi ini menyebabkan beberapa pengembang aplikasi berusaha untuk menyediakan aplikasi antivirus baik yang sifatnya freeware maupun shareware.

Virus pertama kali dikenal pada tahun 1980-an. Pada tahun 1990 diperkirakan ada 500 program virus. Tahun 1996 diperkirakan meningkat menjadi 10000 (Munro, 2002). Saat ini telah ada lebih dari 100000 virus (2008a). Perkembangan virus dan *malware* lain yang begitu cepat inilah yang menyebabkan para pengguna antivirus menunggu virus definition/*malware* signature terbaru dari pengembang antivirus yang mereka gunakan. Namun seringkali virus yang berkembang di suatu wilayah, misalnya di Indonesia, agak lambat penanganannya karena keterlambatan pengembang dalam memperoleh contoh virusnya. Keterlambatan ini dapat diatasi dengan membuat program Fix khusus untuk virus tersebut dengan mengirimkan contoh virus ke server yang di dalamnya terdapat proses yang mampu secara otomatis membangkitkan *script* untuk menangani virus tersebut.

1.2 Perumusan Masalah:

Rumusan masalah yang akan dijawab melalui penelitian yang akan dilakukan adalah sebagai berikut:

1. Bagaimana cara mengamati perilaku suatu *malware* dan perubahan yang dilakukan pada windows *registry*, *file* system dan memory?
2. Bagaimana mengembangkan prototipe aplikasi penanganan *malware* yang dapat secara otomatis membangkitkan *script* untuk menangani *malware* tertentu?

1.3 Tujuan Penelitian:

Penelitian yang akan dilakukan bertujuan untuk:

1. Membangun aplikasi yang mampu mendeteksi tingkah laku dan perubahan yang dilakukan oleh suatu *malware*.
2. Membangun suatu prototipe aplikasi yang mampu menangani *malware* yang dapat secara otomatis membangkitkan *script* untuk menangani *malware* yang ditemukan

1.4 Metodologi:

Penelitian ini dilakukan dengan melakukan sejumlah aktivitas yang berkaitan, antara lain:

1. Studi Pustaka

   Studi pustaka dilakukan dengan mengumpulkan informasi dari buku-buku, artikel, maupun jurnal ilmiah yang membahas mengenai hal-hal yang terkait dengan *malware* dan cara penanganannya.
2. Analisis Kebutuhan Sistem.

   Analisis kebutuhan sistem dilakukan dengan menggali kebutuhan fungsional dari layanan aplikasi yang akan dikembangkan dan menentukan sejauh mana kebutuhan-kebutuhan tersebut akan diakomodasi dalam layanan aplikasi yang akan dibangun.
3. Perancangan Sistem.

   Perancangan sistem dilakukan untuk mendapatkan deskripsi mengenai arsitektural aplikasi dan deskripsi data.
4. Implementasi Sistem.

   Implementasi sistem dilakukan dengan menterjemahkan deskripsi perancangan yang telah dibuat ke dalam kode-kode program sesuai dengan tools yang digunakan untuk membangun aplikasi.
5. Pengujian Sistem

   Pengujian sistem dilakukan untuk menguji fungsionalitas aplikasi yang akan dibangun apakah sudah sesuai dengan spesifikasi yang telah ditentukan(menangani beberapa jenis *malware*).
6. Penulisan Laporan dan Dokumentasi

   Tahap ini dilakukan dengan membuat dokumentasi terhadap seluruh aktivitas penelitian dengan harapan dapat dipergunakan untuk penelitian lainnya.

1.5 Manfaat Hasil Penelitian:

Manfaat yang didapat dari penelitian yang akan dilakukan adalah sebagai berikut:

1. Pemahaman mengenai sistem kerja *malware* dan cara penanganannya.
2. Aplikasi yang dihasilkan dapat digunakan untuk menangani *malware* tertentu.

BAB 2. LANDASAN TEORI

2.1 *Malware*

Istilah *computer virus* pertama kali digunakan oleh Fred Cohen dalam papernya yang berjudul ”*Computer Viruses* – Theory and Experiments” pada tahun 1984. Berikut kutipan definisi yang diberikan oleh Fred Cohen dalam paper tersebut:

”*We define a computer ’virus’ as a program that can ’infect’ other programs by modifiying them to include the possibly evolved copy of itself. With the infection property, a virus can spread throughout a computer system or network using authorizations of every user using it to infect their programs. Every program that gets infected may also act as a virus ang thus the infection grows*”.

Malware, yang merupakan kependekan dari malicious *software*, merujuk pada program yang dibuat dengan tujuan membahayakan atau menyerang sebuah sistem komputer. Saat ini telah berkembang berbagai jenis *malware*, seperti virus, *trojan horse*, *worm*, *joke program*, dan *rootkit*.

2.2 Tinjauan Pustaka Mengenai *Malware*

Malware adalah sepotong kode yang mengubah kelakuan kernel sistem operasi atau beberapa aplikasi yang rawan keamanan, tanpa persetujuan pengguna dan berjalan sedemikian rupa sehingga tidak mungkin mendeteksi perubahannya menggunakan fitur terdokumentasi dari sistem operasi atau aplikasi (misalnya API).(Rutkowska, 2006). Saat ini telah berkembang berbagai jenis *malware*, seperti virus, *trojan horse*, *worm*, dan *rootkit*. Virus sendiri ada berbagai kategori berdasarkan fungsionalitas dan metodologi, yaitu *boot virus*, *parasitic virus*, *bomb*, *macro virus*, *encrypted virus*, *oligomorphic virus*, *polymorphic virus* dan *stealth virus*. Perangkat lunak anti virus sendiri terus berkembang seiring dengan perkembangan *malware*. Ada berbagai macam metodologi yang digunakan oleh perangkat lunak anti virus, yaitu *signature detection* atau *pattern matching*, *x-raying*, *emulation*, *frequency analysis*, dan *heuristics* (Srinivasan, 2007).

Microsoft Windows merupakan salah satu sistem operasi yang populer saat ini serta paling sering diserang. *Malicious software* yang berjalan pada suatu *host* sering kali digunakan untuk melakukan serangan ini. Ada dua pertahanan yang diterapkan secara luas, *virus scanner* dan *security patch*. *Virus scanner* berusaha untuk mendeteksi *malicious software* pada *host*, dan *security patch* merupakan perbaikan sistem operasi untuk menutup

lubang yang dieksploitasi *malicious software*. Kedua metode memiliki kelemahan yang sama. Keduanya efektif melawan serangan yang telah dikenalinya, tetapi tidak dapat mendeteksi dan mencegah serangan tipe baru (Stolfo, 2008a).

Hampir semua *virus scanner* berbasis *signature* menggunakan sederet *byte* atau *embedded string* pada perangkat lunak untuk mengidentifikasikan program tertentu sebagai *malicious software*. Jika suatu basisdata *signature* dari *virus scanner* tidak mengandung *signature* dari *malicious program* tertentu, *virus scanner* tidak mampu mendeteksi atau memproteksi terhadap program tersebut. Secara umum, *virus scanner* membutuhkan pembaharuan *signature* secara teratur, jika tidak maka menjadi tidak berguna. Begitu juga dengan *security patch* yang hanya mampu memproteksi sistem jika telah dibuat, didistribusikan dan diterapkan pada sistem *host*.

Seringkali pembaharuan *virus signature* dan *security patch* tidak dilakukan secara berkala, sehingga menyebabkan sistem menjadi lemah terhadap serangan yang dilakukan oleh *malicious software*. Meskipun pembaharuan sering dilakukan, sistem tersebut masih rawan serangan selama waktu *malicious software* baru diciptakan sampai pengembang perangkat lunak membuat *signature* atau *patch* baru, mendistribusikannya dan diterapkan pada sistem yang terkena serangan. Penelitian ini dilakukan untuk membantu perbaikan sistem terhadap suatu serangan *malware* tertentu tanpa harus menunggu *signature* atau *patch* dari pengembang perangkat lunak anti virus atau sistem operasi yang digunakan.

BAB 3. ANALISA DAN PERANCANGAN SISTEM

Bab ini berisi spesikasi kebutuhan serta perancangan aplikasi *console* untuk mengamati perilaku *malware* dan menciptakan *script* penanganan untuk *malware* tersebut. Sistem penganalisa perilaku *malware* ini selanjutnya akan disebut sebagai SenaPrima. Aplikasi ini dibuat dengan antarmuka yang sangat sederhana untuk mempermudah pemakaian dan mengurangi beban komputasi. Secara umum sistem ini memiliki cara kerja sebagai berikut:

1. melakukan penyimpanan kondisi awal (*snapshot*) dan menyimpan ke basisdata sementara
2. menyalin dan menjalankan *malware*
3. menyimpan kondisi akhir (*snapshot*) dan menyimpan ke basisdata sementara
4. membandingkan kedua *snapshot*
5. membuat laporan berupa dokumen HTML
6. menciptakan *script* penanganan berupa INF

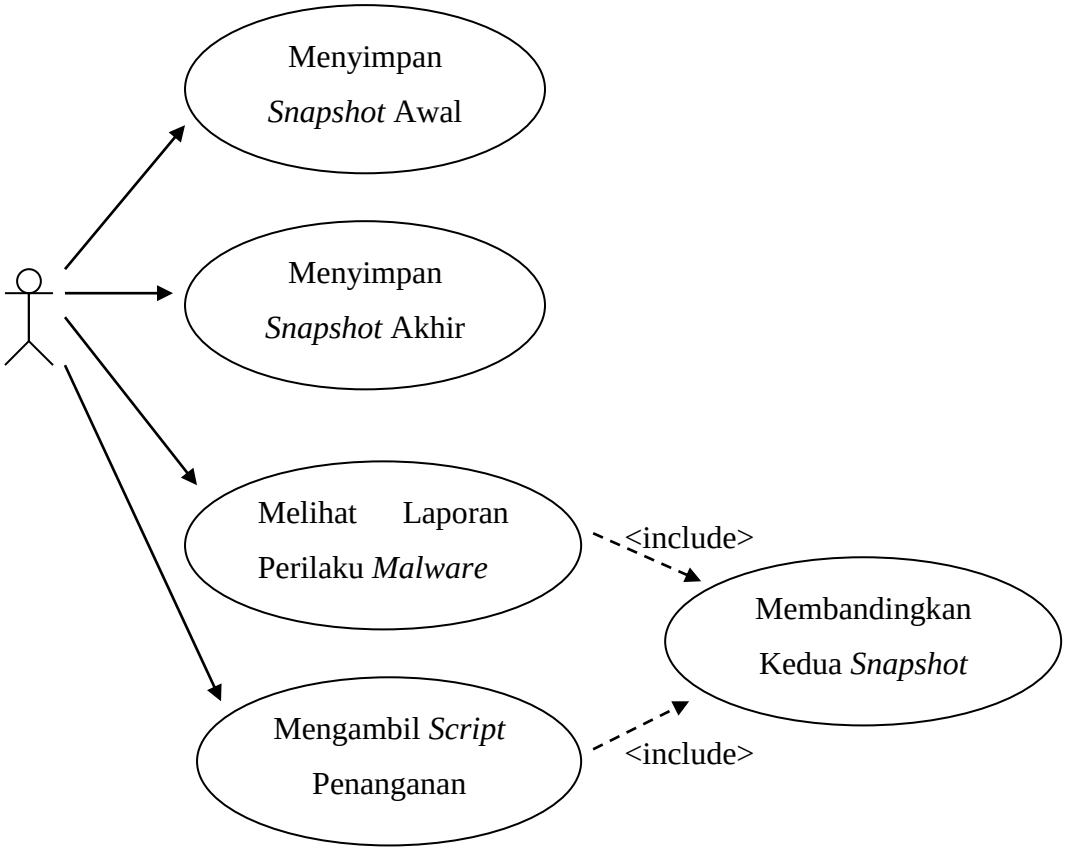
3.1 Spesifikasi Kebutuhan Pengguna

Pengguna SenaPrima ini adalah orang yang ingin mengamati perilaku sebuah *malware* atau orang yang bermasalah dengan sebuah *malware* dan ingin menanganinya. Pengguna bisa melihat perilaku *malware* dengan melihat laporan yang dihasilkan, berupa halaman HTML. Selain laporan tersebut, aplikasi ni juga menghasilkan *script* yang dapat digunakan pengguna untuk menangani *malware*.

3.2 Spesifikasi Kebutuhan Fungsional

Berdasarkan analisa, kebutuhan fungsionalitas dari SenaPrima ini ditunjukkan dengan diagram *use case* pada gambar 3.1 yang rinciannya akan dijelaskan berikut ini:

Use case Menyimpan *Snapshot* Awal memungkinkan pengguna meminta aplikasi untuk membaca seluruh isi (*file* dan direktori) dari media penyimpan, proses yang berjalan, dan *registry* windows, serta menyimpan data tersebut dalam basisdata. *Use case* ini dijalankan sebelum pengguna menjalankan *malware* yang ingin diamati/ditangani.



Gambar 3.1. Diagram *Use Case* SenaPrima

Use case Menyimpan *Snapshot* Akhir memungkinkan pengguna meminta aplikasi untuk membaca seluruh isi (*file* dan direktori) dari media penyimpan, proses yang berjalan, dan *registry* windows, serta menyimpan data tersebut dalam basisdata. *Use case* ini dijalankan setelah pengguna menjalankan *malware* yang ingin diamati/ditangani.

Use case Membandingkan Kedua *Snapshot* berfungsi untuk mencari perubahan yang terjadi antara kondisi sebelum dan setelah *malware* dijalankan. Perubahan ini dapat terjadi jika ada proses baru diciptakan, proses dimatikan, *file*/direktori baru, *file*/direktori dihapus, *registry key* baru, *registry key* dihapus,atau nilai *registry* diubah.

Use case Melihat Laporan Perilaku *Malware* memungkinkan pengguna untuk melihat perubahan yang terjadi dalam sistem komputer. *Use case* ini memanfaatkan hasil pembandingan yang dilakukan *use case* Membandingkan Kedua *Snapshot* untuk menciptakan halaman HTML sebagai laporan bagi pengguna. Laporan kepada pengguna berisi informasi mengenai proses baru, proses dimatikan, *file*/direktori baru, *file*/direktori dihapus, *registry key* baru, *registry key* dihapus,dan nilai *registry* yang diubah.

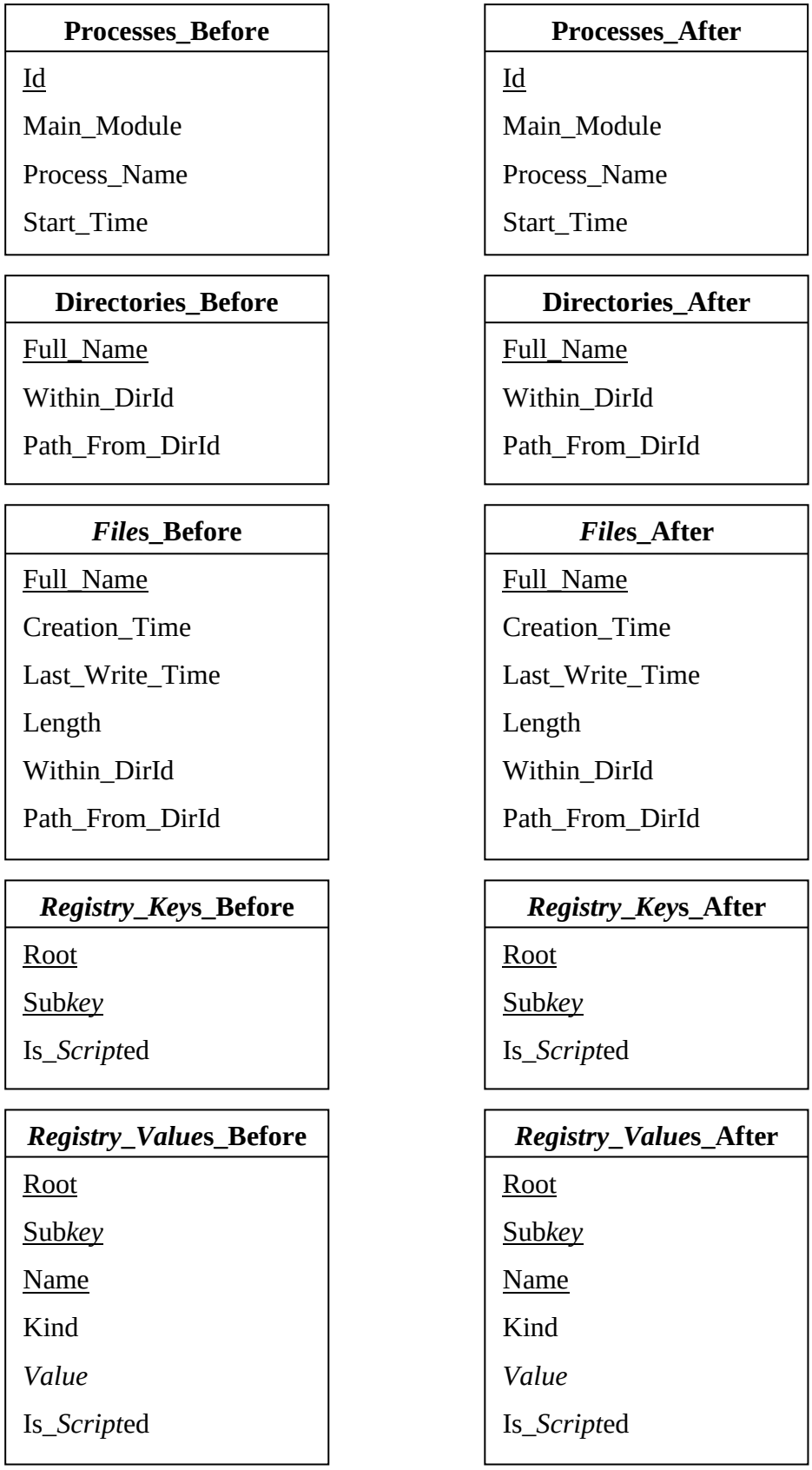
Use case Mengambil *Script* Penangan memungkinkan pengguna untuk memperoleh *script* INF yang dapat digunakannya untuk menangani *malware* yang diamati. *Use case* ini memanfaatkan hasil pembandingan yang dilakukan *use case* Membandingkan Kedua *Snapshot* untuk menciptakan *file script*. *Script* memiliki kemampuan untuk mematikan proses baru, menghapus *file*/direktori baru, menghapus *registry key* baru, serta mengembalikan nilai *registry* yang diubah *malware*.

3.3 Spesifikasi kebutuhan Data

Dari analisa, dibutuhkan basisdata untuk menyimpan data *snapshot* dari sistem komputer, baik sebelum maupun setelah *malware* dijalankan. Basisdata ini bersifat sementara, artinya basisdata hanya ada selama proses pengamatan dilakukan. Jika aplikasinya ditutup, maka basisdatanya akan terhapus.

Data yang dibutuhkan untuk disimpan sementara, yang nantinya digunakan untuk pembandingan konsidi sebelum dan setelah *malware* dijalankan terlihat pada Gambar 3.2. Entitas data yang dibutuhkan yaitu:

- **Entitas *Processes_Before*** yaitu entitas yang merepresentasikan data proses-proses yang berjalan pada sistem komputer tersebut pada saat diambil *snapshot*-nya sebelum *malware* dijalankan.
- **Entitas *Processes_After*** yaitu entitas yang merepresentasikan data proses-proses yang berjalan pada sistem komputer tersebut pada saat diambil *snapshot*-nya setelah *malware* dijalankan.
- **Entitas *Directories_Before*** yaitu entitas yang merepresentasikan data direktori-direktori yang ada pada *file* system komputer tersebut pada saat diambil *snapshot*-nya sebelum *malware* dijalankan.
- **Entitas *Directories_After*** yaitu entitas yang merepresentasikan data direktori-direktori yang ada pada *file* system komputer tersebut pada saat diambil *snapshot*-nya setelah *malware* dijalankan.
- **Entitas *Files_Before*** yaitu entitas yang merepresentasikan data *file-file* yang ada pada *file* system komputer tersebut pada saat diambil *snapshot*-nya sebelum *malware* dijalankan.
- **Entitas *Files_After*** yaitu entitas yang merepresentasikan data *file-file* yang ada pada *file* system komputer tersebut pada saat diambil *snapshot*-nya setelah *malware* dijalankan.

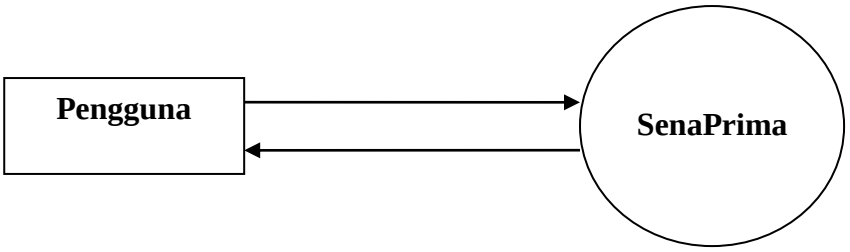


Gambar 3.2. *Entity Relationship Diagram (ERD)* dari SenaPrima

- **Entitas *Registry_Keys_Before*** yaitu entitas yang merepresentasikan data semua *registry key* yang tersimpan pada sistem komputer tersebut pada saat diambil *snapshot*-nya sebelum *malware* dijalankan.
- **Entitas *Registry_Keys_After*** yaitu entitas yang merepresentasikan data semua *registry key* yang tersimpan pada sistem komputer tersebut pada saat diambil *snapshot*-nya setelah *malware* dijalankan.
- **Entitas *Registry_Values_Before*** yaitu entitas yang merepresentasikan data semua *registry value* yang tersimpan pada sistem komputer tersebut pada saat diambil *snapshot*-nya sebelum *malware* dijalankan.
- **Entitas *Registry_Values_After*** yaitu entitas yang merepresentasikan data semua *registry value* yang tersimpan pada sistem komputer tersebut pada saat diambil *snapshot*-nya setelah *malware* dijalankan.

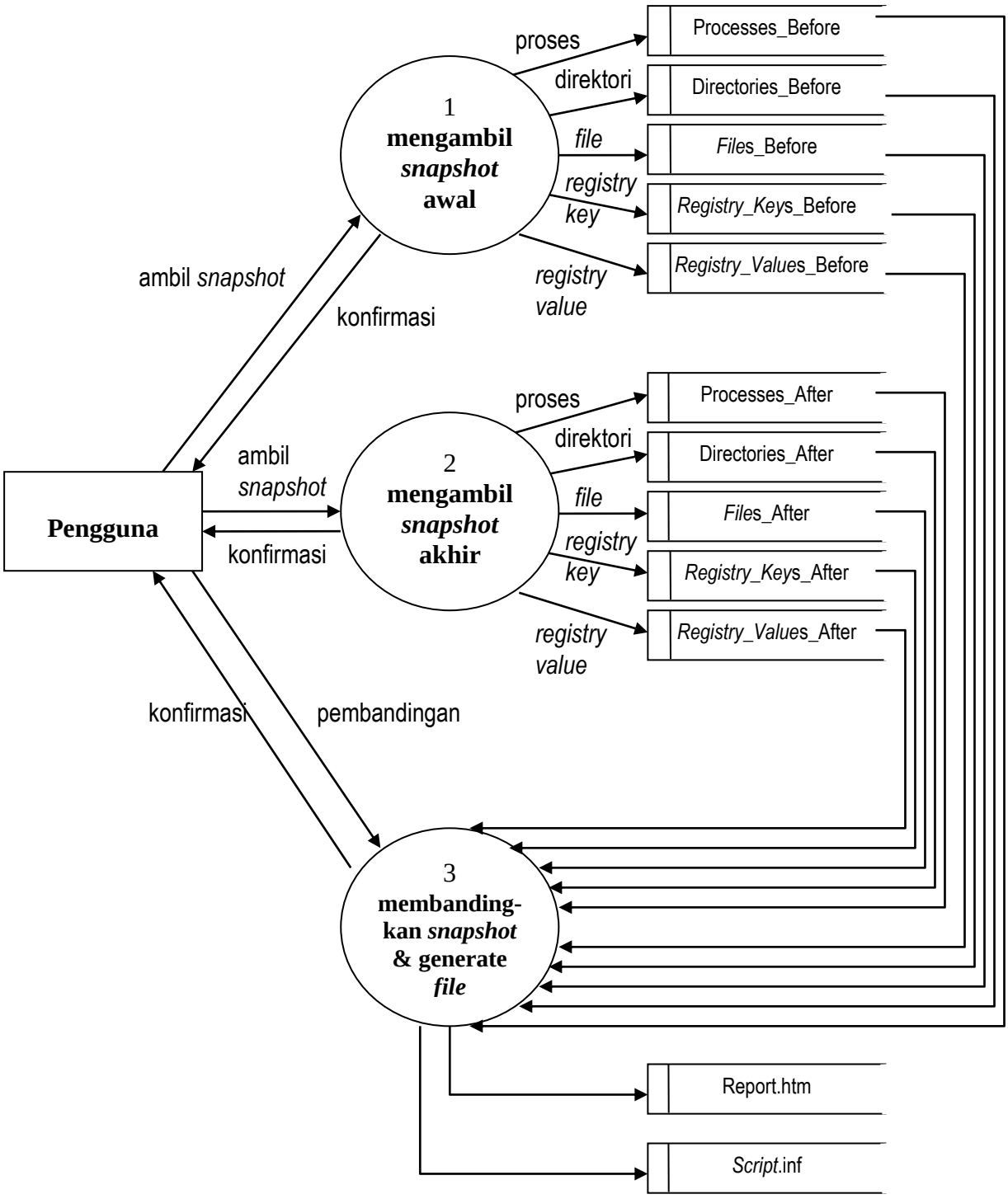
3.4 Perancangan Fungsional

Fungsionalitas-fungsionalitas sistem penganalisa perilaku *malware* ini yang dideskripsikan dalam bentuk *use case* pada bab sebelumnya, selanjutnya direalisasi dalam bentuk Diagram Alir Data (Data Flow Diagram, DFD).



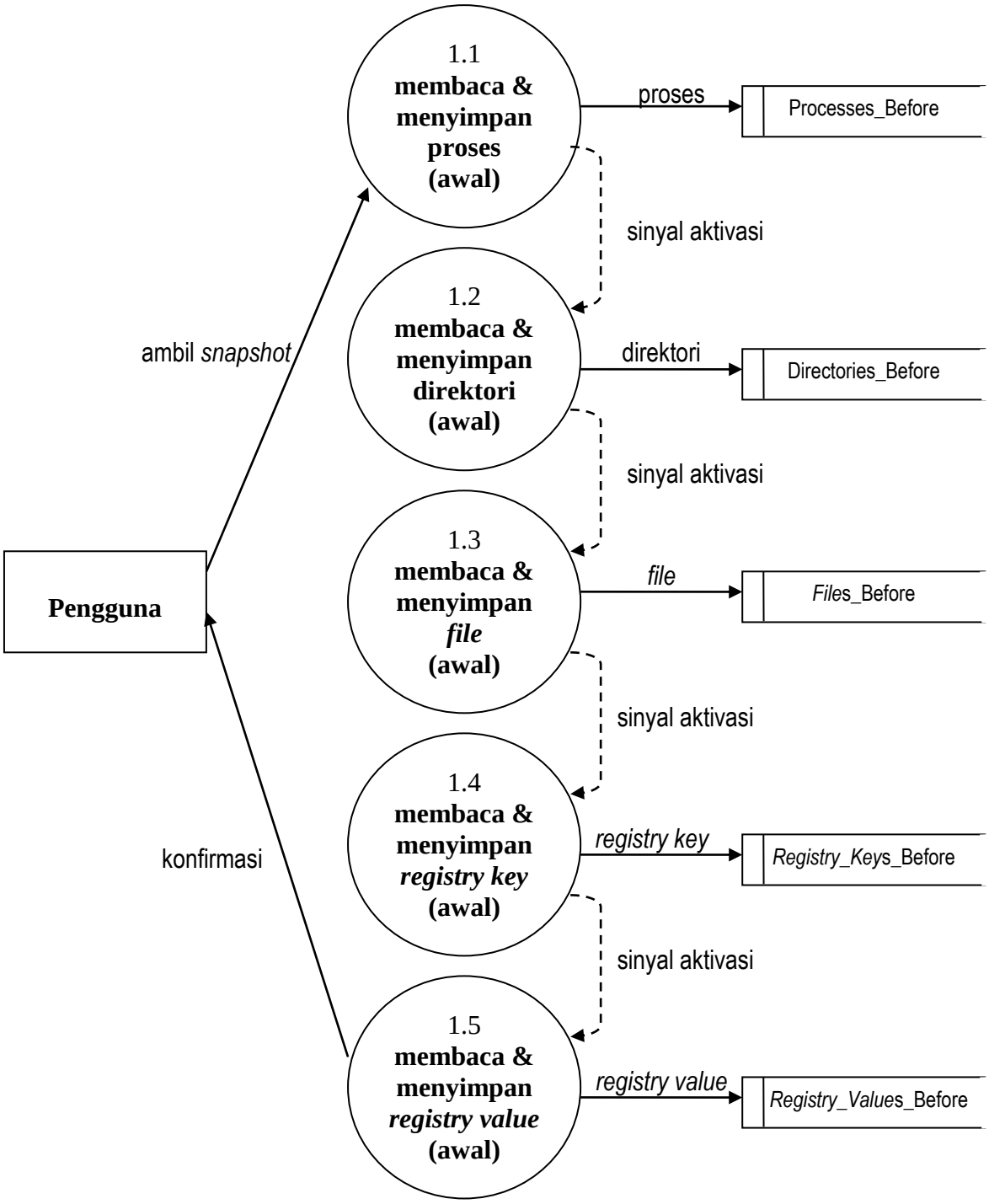
Gambar 3.3. Diagram Context

Gambar 3.3 menunjukkan diagram context dari sistem penganalisa perilaku *malware* (Sena Prima) yang dikembangkan. Hanya ada satu tipe pengguna, yaitu orang yang memanfaatkan aplikasi ini untuk mengamati perilaku *malware*.



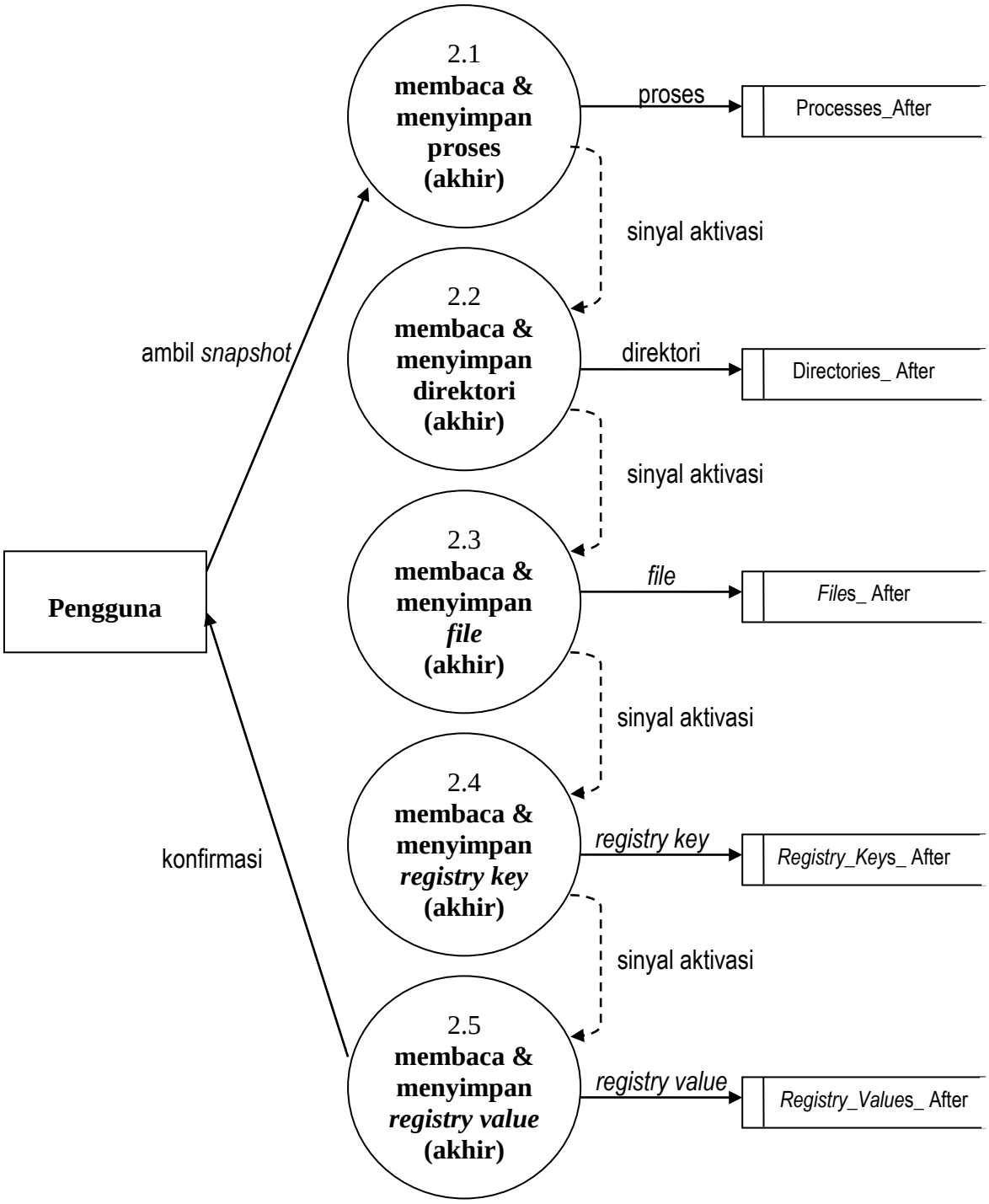
Gambar 3.4. DFD Level 1

Gambar 3.4 menunjukkan data flow diagram level 1 dari sistem penganalisa perilaku *malware*. Disini terdapat tiga proses utama, yaitu pengambilan *snapshot* kondisi sistem sebelum dan setelah *malware* dijalankan dan pembandingan keduanya untuk mengetahui perubahan yang terjadi. Data yang disimpan di basisdata berupa semua proses, direktori, *file*, *registry key*, dan *registry value*.



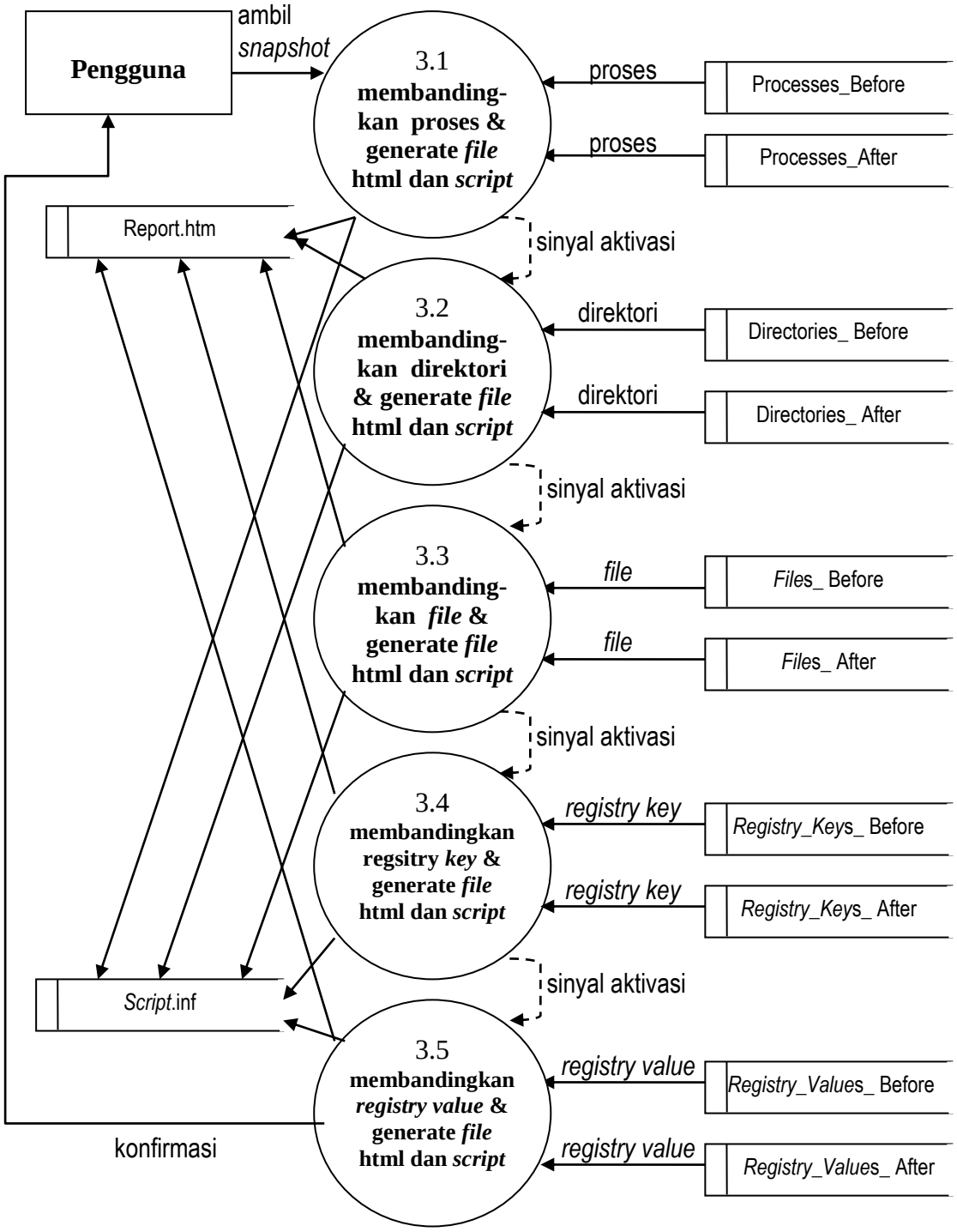
Gambar 3.5. DFD Level 2 dari Proses 1

Gambar 3.5 menunjukkan detail dari proses mengambil *snapshot* awal. Pengguna hanya berinteraksi dengan proses diawal dan diakhir. Semua data yang terkait dengan proses, direktori, *file*, *registry key*, dan *registry value* dibaca dari sistem sebelum *malware* dijalankan dan disimpan di basisdata.



Gambar 3.6. DFD Level 2 dari Proses 2

Gambar 3.6 menunjukkan detail dari proses mengambil *snapshot* akhir. Pengguna hanya berinteraksi dengan proses diawal dan diakhir. Semua data yang terkait dengan proses, direktori, *file*, *registry key*, dan *registry value* dibaca dari sistem setelah *malware* dijalankan dan disimpan di basisdata.



Gambar 3.7. DFD Level 2 dari Proses 3

Gambar 3.7 menunjukkan detail dari proses membandingkan dan generate *file*. Pembandingan dilakukan dengan memanfaatkan data kondisi sebelum dan setelah *malware* dijalankan. Dari proses ini akan dihasilkan dua *file*, yaitu sebuah halaman html yang berisi informasi perubahan dan *file* inf yang berisi *script* untuk menangani *malware* tersebut.

3.5 Perancangan Data

Selanjutnya berdasarkan analisa kebutuhan data yang tertuang dalam bentuk *ERD (Entity Relationship Diagram)* pada gambar 3.2, maka dapat disusun rancangan basis data yang tertuang dalam bentuk Relasi antar tabel. Karena tidak ada relasi antar tabel-tabel yang dibuat, maka gambar *physical model* dari basisdata yang dibuat sama dengan gambar ERD (Gambar 3.2). Berikut akan dijabarkan mengenai kamus data dari tabel-tabel yang ada.

Tabel Processes_Before

Nama Elemen	Tipe Data	Range Nilai	Keterangan
ID	Integer	0-99999999	PRIMARY *KEY*
Main_Module	Text	[a-z,A-Z,0-9]	NOT NULL
Process_Name	Text	[a-z,A-Z,0-9]	NOT NULL
Start_Time	Integer	0-99999999	NOT NULL

Tabel Processes_After

Nama Elemen	Tipe Data	Range Nilai	Keterangan
ID	Integer	0-99999999	PRIMARY *KEY*
Main_Module	Text	[a-z,A-Z,0-9]	NOT NULL
Process_Name	Text	[a-z,A-Z,0-9]	NOT NULL
Start_Time	Integer	0-99999999	NOT NULL

Tabel Directories_Before

Nama Elemen	Tipe Data	Range Nilai	Keterangan
Full_Name	Text	[a-z,A-Z,0-9]	PRIMARY *KEY*
Within_DirId	Integer	0-99999999	NOT NULL
Path_From_DirId	Text	[a-z,A-Z,0-9]	NOT NULL

Tabel Directories_After

Nama Elemen	Tipe Data	Range Nilai	Keterangan
Full_Name	Text	[a-z,A-Z,0-9]	PRIMARY *KEY*
Within_DirId	Integer	0-99999999	NOT NULL
Path_From_DirId	Text	[a-z,A-Z,0-9]	NOT NULL

Tabel *File*s_Before

Nama Elemen	Tipe Data	Range Nilai	Keterangan
Full_Name	Text	[a-z,A-Z,0-9]	PRIMARY *KEY*
Creation_Time	Integer	0-99999999	NOT NULL
Last_Write_Time	Integer	0-99999999	NOT NULL
Length	Integer	0-99999999	NOT NULL
Within_DirId	Integer	0-99999999	NOT NULL
Path_From_DirId	Text	[a-z,A-Z,0-9]	NOT NULL

Tabel *File*s_After

Nama Elemen	Tipe Data	Range Nilai	Keterangan
Full_Name	Text	[a-z,A-Z,0-9]	PRIMARY *KEY*
Creation_Time	Integer	0-99999999	NOT NULL
Last_Write_Time	Integer	0-99999999	NOT NULL
Length	Integer	0-99999999	NOT NULL
Within_DirId	Integer	0-99999999	NOT NULL
Path_From_DirId	Text	[a-z,A-Z,0-9]	NOT NULL

Tabel *Registry_Key*s_Before

Nama Elemen	Tipe Data	Range Nilai	Keterangan
Root	Integer	0-99999999	PRIMARY *KEY*
Sub*key*	Text	[a-z,A-Z,0-9]	PRIMARY *KEY*
Is_*Script*ed	Integer	0-99999999	NOT NULL

Tabel *Registry_Key*s_After

Nama Elemen	Tipe Data	Range Nilai	Keterangan
Root	Integer	0-99999999	PRIMARY *KEY*
Sub*key*	Text	[a-z,A-Z,0-9]	PRIMARY *KEY*
Is_*Script*ed	Integer	0-99999999	NOT NULL

Tabel *Registry_Value*s_Before

Nama Elemen	Tipe Data	Range Nilai	Keterangan
Root	Integer	0-99999999	PRIMARY *KEY*
Sub*key*	Text	[a-z,A-Z,0-9]	PRIMARY *KEY*
Name	Text	[a-z,A-Z,0-9]	PRIMARY *KEY*
Kind	Integer	0-99999999	NOT NULL
Value	Blob		NOT NULL
Is_*Script*ed	Integer	0-99999999	NOT NULL

Tabel *Registry_Value*s_After

Nama Elemen	Tipe Data	Range Nilai	Keterangan
Root	Integer	0-99999999	PRIMARY *KEY*
Sub*key*	Text	[a-z,A-Z,0-9]	PRIMARY *KEY*
Name	Text	[a-z,A-Z,0-9]	PRIMARY *KEY*
Kind	Integer	0-99999999	NOT NULL
Value	Blob		NOT NULL
Is_*Script*ed	Integer	0-99999999	NOT NULL

3.6 Antar Muka Pengguna

Desain antar muka aplikasi ini dirancang sesederhana mungkin untuk meminimalkan beban komputasi dan untuk kemudahan penggunaan. Aplikasi ini berjalan pada *console* (*Command Line User Interface*). Dalam menjalankan aplikasi ini pengguna hanya diminta untuk menekan tombol ENTER untuk memulai pengambilan *snapshot* atau pembandingan. Berikut adalah rancangan tampilan untuk aplikasi ini.

```
************************************************************
* Tekan ENTER untuk menyimpan snapshot 'awal' dari sistem  *
************************************************************

Penyimpanan snapshot dalam progres...
```

Gambar 3.8. Rancangan Tampilan Pengambilan *Snapshot* Awal

```
************************************************************
* Tekan ENTER untuk menyimpan snapshot 'akhir' dari sistem *
************************************************************

Penyimpanan snapshot dalam progres...
```

Gambar 3.9. Rancangan Tampilan Pengambilan *Snapshot* Akhir

```
*************************************************************************
* Tekan ENTER untuk membandingkan snapshot serta membuat report and script *
*************************************************************************

Pembandingan snapshot serta pembuatan report dan script dalam progres...
```

Gambar 3.10. Rancangan Tampilan Pembandingan *Snapshot*

Gambar 3.8 merupakan rancangan tampilan aplikasi pada *console* yang meminta pengguna menekan ENTER untuk memulai menyimpan *snapshot*. Aplikasi juga memberikan pesan selama proses tersebut masih berlangsung. Gambar 3.9 mirip dengan gambar 3.8, hanya pengambilan *snapshot* dilakukan setelah *malware* dijalankan. Setelah kedua *snapshot* tersimpan, maka pengguna dapat memulai meminta aplikasi untuk mencari perubahan yang terjadi dan menciptkan *file* html untuk laporan perubahan dan *script* yang nantinya digunakan untuk menghapus proses, direktori, *file* atau registy *key* baru yang dibuat *malware*, serta mengembalikan *registry value* yang nilainya diubah *malware*.

```
INFORMASI UMUM
Laporan diciptakan pada : .....................
Sistem Operasi          : .....................

DAFTAR PERUBAHAN PROSES
(nama proses yang baru diciptakan malware)
(nama proses yang dimatikan malware)

DAFTAR PERUBAHAN FILE
(nama direktori dan file yang baru diciptakan malware)
(nama file yang diubah malware)
(nama direktori dan file yang dihapus malware)

DAFTAR PERUBAHAN REGISTRY
(nama registry key baru yang diciptakan)
(nama registry key yang dihapus malware)
(nama registry key dan value yang diubah oleh malware)
```

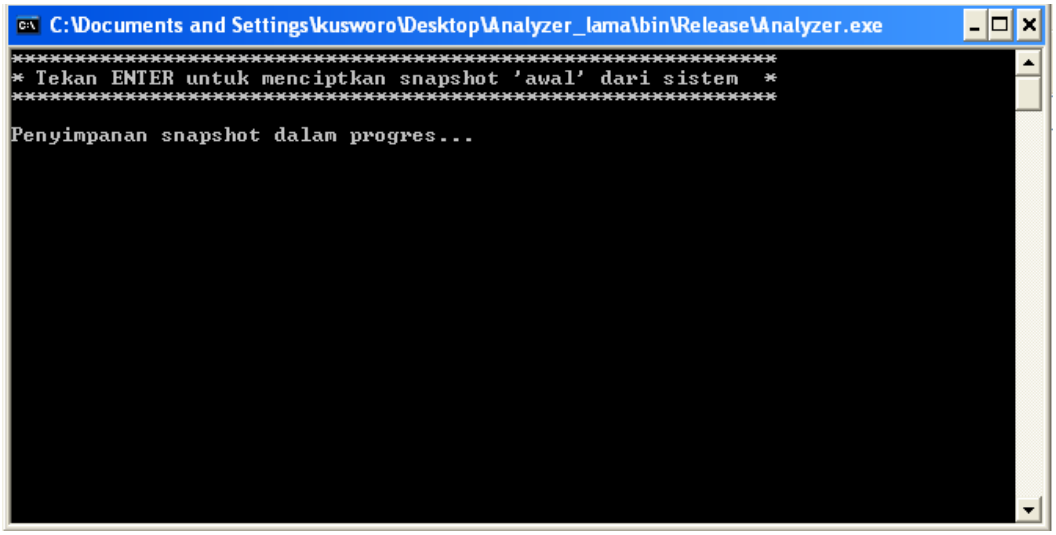
Gambar 3.11. Rancangan Tampilan Laporan Perubahan (HTML)

Gambar 3.11 menunjukkan rancangan tampilan dari halaman web yang berisi informasi mengenai perubahan yang terjadi dalam sistem komputer sebelum dan setelah *malware* dijalankan.

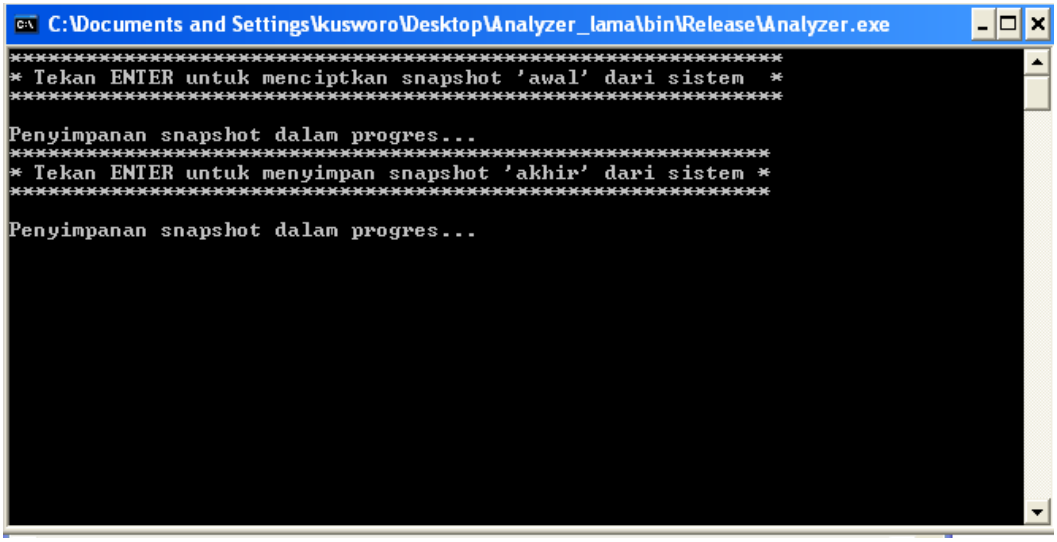
BAB 4. IMPLEMENTASI DAN PEMBAHASAN

4.1 Hasil Implementasi

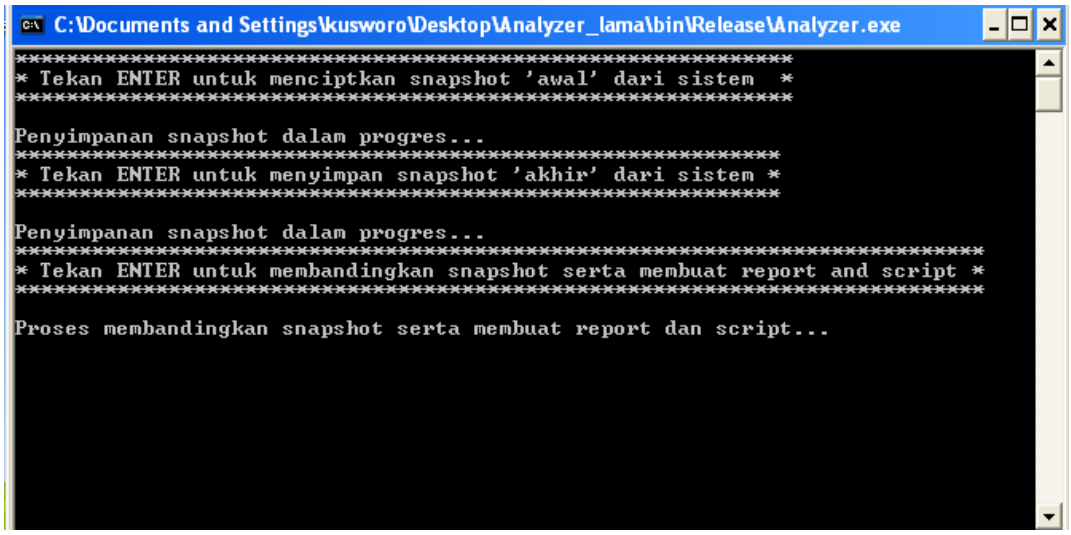
Bab ini menjelaskan hasil implementasi dari sistem penganalisa perilaku *malware*. Program aplikasi dibangun dengan menggunakan framework .NET dengan bahasa pemrograman C#. Aplikasi ini berjalan pada *console*. Berikut ini akan ditampilkan beberapa gambar selama proses berlangsung dan cuplikan isi *file script* dan laporan dalam bentuk HTML.



Gambar 4.1. Pengambilan *Snapshot* Awal



Gambar 4.2. Pengambilan *Snapshot* Akhir



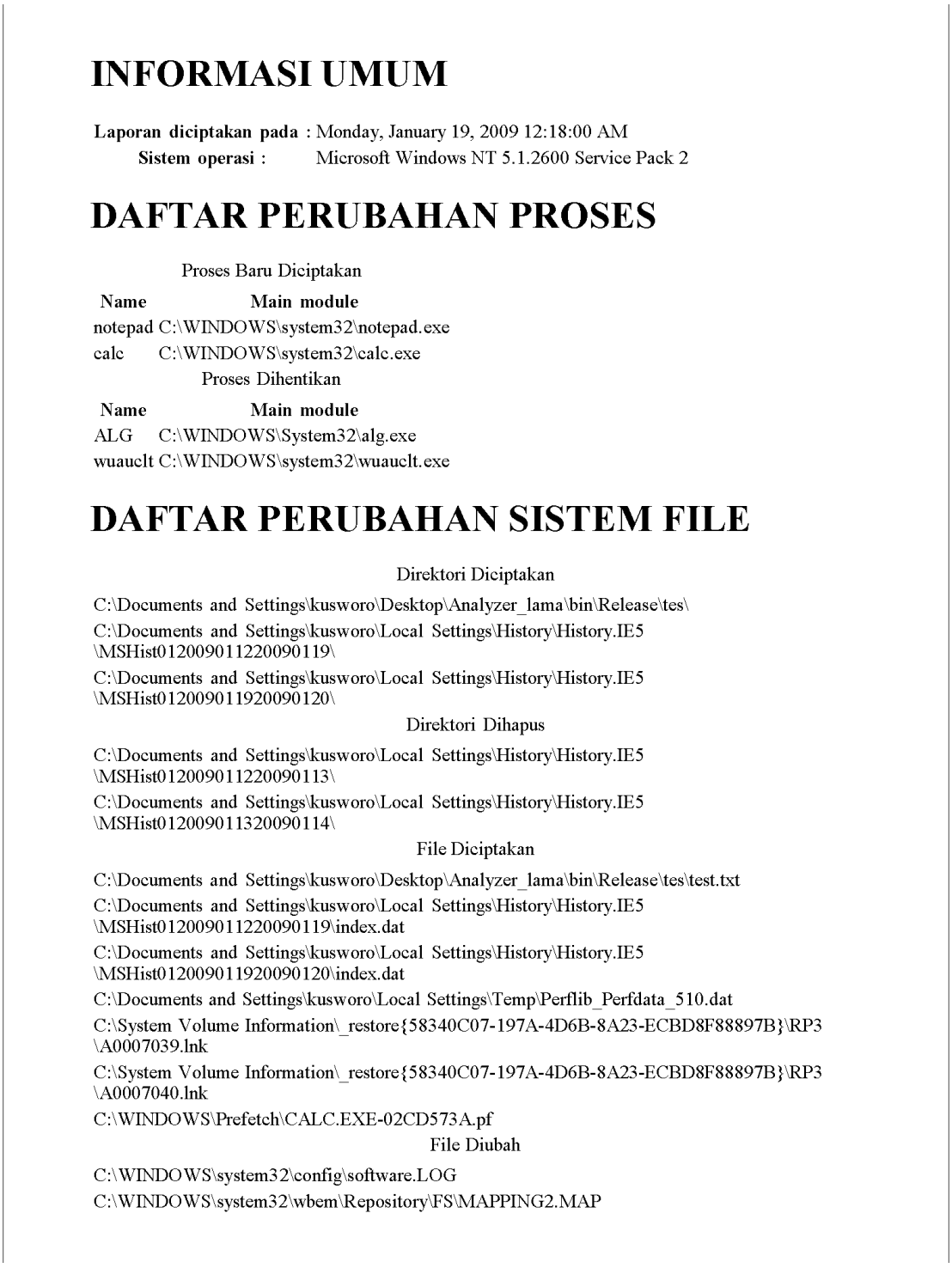
Gambar 4.3. Proses Pembandingan serta Pembuatan *File* HTML dan *Script*

Gambar 4.1 memperlihatkan tampilan pada saat pengguna telah menekan tombol ENTER untuk memulai penyimpanan *snapshot* awal. Data mengenai semua proses yang sedang berjalan, direktori dan *file* yang ada di sistem *file*, serta *registry* pada saat itu akan tersimpan di basisdata pada tabel yang sesuai.

Pada saat aplikasi memberi pesan untuk menyimpan *snapshot* akhir, *malware* yang ingin diamati dijalankan. Setelah menunggu beberapa saat, pengguna dapat menyimpan *snapshot* akhir dengan menekan tombol ENTER. Data mengenai semua proses yang sedang berjalan, direktori dan *file* yang ada di sistem *file*, serta *registry* pada saat itu akan tersimpan di basisdata pada tabel yang sesuai. Tampilan saat pemrosesan dapat dilihat pada gambar 4.2.

Setelah pengambilan *snapshot* akhir selesai, pengguna dapat langsung menuruh aplikasi untuk mencari perubahan yang terjadi. Aplikasi akan membandingkan data yang telah tersimpan di basisdata. Kemudian menampilkan informasi perubahannya dalam bentuk halaman web dan membuat *file script* untuk menangani perubahan tersebut. Gambar 4.3 memperlihatkan tampilan saat pemrosesan perbandingan dan pembuatan *file*.

Aplikasi akan menutup secara otomatis pada saat pembandingan dan pembuatan *file* telah selesai. *File* Report.htm dan *Script*.inf akan tersimpan di direktori yang sama dengan program aplikasi.



INFORMASI UMUM

Laporan diciptakan pada : Monday, January 19, 2009 12:18:00 AM
Sistem operasi : Microsoft Windows NT 5.1.2600 Service Pack 2

DAFTAR PERUBAHAN PROSES

Proses Baru Diciptakan

Name	Main module
notepad	C:\WINDOWS\system32\notepad.exe
calc	C:\WINDOWS\system32\calc.exe

Proses Dihentikan

Name	Main module
ALG	C:\WINDOWS\System32\alg.exe
wuauclt	C:\WINDOWS\system32\wuauclt.exe

DAFTAR PERUBAHAN SISTEM FILE

Direktori Diciptakan

C:\Documents and Settings\kusworo\Desktop\Analyzer_lama\bin\Release\tes\
C:\Documents and Settings\kusworo\Local Settings\History\History.IE5 \MSHist012009011220090119\
C:\Documents and Settings\kusworo\Local Settings\History\History.IE5 \MSHist012009011920090120\

Direktori Dihapus

C:\Documents and Settings\kusworo\Local Settings\History\History.IE5 \MSHist012009011220090113\
C:\Documents and Settings\kusworo\Local Settings\History\History.IE5 \MSHist012009011320090114\

File Diciptakan

C:\Documents and Settings\kusworo\Desktop\Analyzer_lama\bin\Release\tes\test.txt
C:\Documents and Settings\kusworo\Local Settings\History\History.IE5 \MSHist012009011220090119\index.dat
C:\Documents and Settings\kusworo\Local Settings\History\History.IE5 \MSHist012009011920090120\index.dat
C:\Documents and Settings\kusworo\Local Settings\Temp\Perflib_Perfdata_510.dat
C:\System Volume Information_restore{58340C07-197A-4D6B-8A23-ECBD8F88897B}\RP3 \A0007039.lnk
C:\System Volume Information_restore{58340C07-197A-4D6B-8A23-ECBD8F88897B}\RP3 \A0007040.lnk
C:\WINDOWS\Prefetch\CALC.EXE-02CD573A.pf

File Diubah

C:\WINDOWS\system32\config\software.LOG
C:\WINDOWS\system32\wbem\Repository\FS\MAPPING2.MAP

Gambar 4.4. Informasi Perubahan pada Halaman Web

Gambar 4.4 memperlihatkan tampilan report mengenai perubahan yang terjadi pada sistem komputer setelah *malware* dijalankan. Daftar perubahan yang terjadi bisa cukup panjang dan ada kemungkinan perubhan yang terjadi tidak membahayakan. Melalui halaman web ini pengguna mendapatkan informasi mengenai perilaku *malware* yang diamati.

```
[Version]
Signature=$CHICAGO$
AdvancedINF=2.5

[DefaultInstall]
RequiredEngine=SETUPAPI
CustomDestination=CustomDestination
DelFiles=DelFiles_16384
DelFiles=DelFiles_16418
DelFiles=DelFiles_16420
DelFiles=DelFiles_16424
DelFiles=DelFiles_49000
DelDirs=DelDirs
AddReg=AddReg
DelReg=DelReg

[CustomDestination]
49000=CustomDestination_49000,4
49001=CustomDestination_49001,4
49002=CustomDestination_49002,5
49003=CustomDestination_49003,5

[CustomDestination_49000]
,,,,"C:\"

[CustomDestination_49001]
,,,,"E:\"

[CustomDestination_49002]
,,,,"%16418%\History.IE5\MSHist012009011220090113\"

[CustomDestination_49003]
,,,,"%16418%\History.IE5\MSHist012009011320090114\"

[DestinationDirs]
DelFiles_16384=16384
DelFiles_16418=16418
DelFiles_16420=16420
DelFiles_16424=16424
DelFiles_49000=49000

[DelFiles_16384]
"Analyzer_lama\bin\Release\tes\test.txt",,,1

[DelFiles_16418]
"History.IE5\MSHist012009011220090119\index.dat",,,1
"History.IE5\MSHist012009011920090120\index.dat",,,1

[DelFiles_16420]
"Prefetch\CALC.EXE-02CD573A.pf",,,1

[DelFiles_16424]
"Local  Settings\Temp\Perflib_Perfdata_510.dat",,,1

[DelFiles_49000]
"System  Volume  Information\_restore{58340C07-197A-4D6B-8A23-
ECBD8F88897B}\RP3\A0007039.lnk",,,1
"System  Volume  Information\_restore{58340C07-197A-4D6B-8A23-
ECBD8F88897B}\RP3\A0007040.lnk",,,1

[DelDirs]
"%16418%\History.IE5\MSHist012009011920090120\"
```

Gambar 4.5. *Script* yang Tercipta untuk Menangani *Malware*

Selain menciptakan halaman web untuk memberikan informasi perilaku *malware* yang diamati, aplikasi ini juga menciptakan *file script* INF yang berisi *script* untuk mengmbalikan kondisi yang diubah *malware*. *Script* dijalankan dengan melakukan klik kanan dan memilih menu Install.

4.2 Pembahasan

Secara umum sistem penganalisa perilaku *malware* ini dapat berjalan dengan baik. Disarankan untuk melakukan pengamatan di Virtual Machine untuk menghindari kemungkinan terjadinya kerusakan yang dilakukan *malware* pada sistem. Pengguna yang ingin mengamati perilaku sebuah *malware* dapat menggunakan aplikasi ini. Pertama simpan data semua proses, direktori, *file*, *registry key* dan *registry value* ke dalam basisdata. Kemudian pengguna menjalankan *malware* yang ingin diamati. Setelah menunggu beberapa saat, pengguna menyimpan *snapshot* lagi. Perubahan yang dilakukan *malware* dapat diperoleh dengan membandingkan kedua *snapshot* yang tersimpan pada basisdata. Informasi yang diperoleh tersebut kemudian ditampilkan dalam bentuk halam web serta digunakan untuk menciptakan *file script* INF yang dapat dipergunakan untuk menangani *malware* yang diamati.

Aplikasi ini mampu menangani berbagai perubahan pada sistem yang dilakukan oleh *malware*, antara lain:

1. Menghentikan proses yang dijalankan *malware*
2. Menghapus direktori yang diciptakan *malware*
3. Menciptakan direktori yang dihapus *malware*
4. Menghapus *file* yang diciptakan *malware*
5. Menghapus *registry key* yang diciptakan *malware*
6. Menciptakan *registry key* yang dihapus *malware*
7. Menghapus *registry value* yang diciptakan *malware*
8. Menciptakan *registry value* yang dihapus *malware*
9. Mengembalikan *registry value* yang diubah *malware*

Meskipun memiliki banyak kemampuan yang dapat membantu pengguna yang terkena serangan suatu *malware*, tetapi aplikasi ini masih memiliki kekurangan:

1. tidak dapat mengembalikan *file* yang dihapus *malware*
2. tidak dapat memulihkan *file* yang diubah *malware*

Program aplikasi ini juga tidak menjalankan kembali proses yang dihentikan oleh *malware*, dengan asumsi pengguna dapat menjalankannya secara manual. Perubahan *registry key* yang dilakukan *malware* dapat ditangani dengan menghapus *registry key* yang berubah tersebut dan menciptakan yang baru sesuai *registry key* yang lama.

Selain kelemahan yang telah disebutkan di atas, program aplikasi ini juga belum melakukan pemilahan perubahan mana saja yang benar-benar dilakukan oleh *malware* dan perubahan yang memang terjadi secara wajar. Misalnya pada saat menunggu *malware* bekerja pengguna menjalankan aplikasi atau mengubah current directory di explorer, maka aplikasi ini juga mencatat perubahan proses yang berjalan serta perubahan pada *registry*. Perubahan ini kemudian ditampilkan di halaman web dan diciptakan *script* untuk menanganinya. Meskipun pada saat *script* dijalankan tidak akan mengganggu sistem, tetapi pengguna kemungkinan akan kesulitan mengetahui perilaku sebenarnya dari *malware*.

Isu-isu yang belum diekslplorasi dalam penelitian ini diharapkan dapat menjadi bahan studi lanjutan. Mengingat bidang rekayasa *malware* ini terus berkembang sehingga masih banyak kemungkinan studi dan fitur-fitur baru yang dapat dimunculkan.

BAB 5. KESIMPULAN

Prototipe sistem penganalisa perilaku *malware* ini telah berhasil dikembangkan dengan baik, meskipun masih memiliki beberapa kekurangan. Perilaku *malware* diamati dengan mencari perubahan yang terjadi pada proses, sistem *file* dan *registry*. Perubahan yang dilakukan *malware* dapat diperoleh dengan membandingkan *snapshot* sebelum *malware* dijalankan dengan *snapshot* setelah *malware* dijalankan. Agar mudah diakses dan dibaca pengguna, maka informasi perubahan ini ditampilkan dalam bentuk web.

Perubahan yang dilakukan oleh *malware* dapat diperbaiki dengan *script* inf. informasi yang diperoleh dari pembandingan kedua *snapshot* tersebut digunakan untuk membangkitkan *script* yang sesuai. *script* yang dihasilkan mampu menghentikan proses yang diciptakan *malware*, menghapus direktori dan *file* yang diciptakan *malware*, serta mengembalikan *registry* yang diubah *malware*. Tetapi, aplikasi ini belum memiliki kemampuan untuk mengembalikan *file* yang dihapus *malware* serta tidak dapat memulihkan *file* yang diubah *malware*.

Dengan sistem penganalisa perilaku *malware* ini diharapkan pengguna sementara dapat menangani kerusakan pada sistem komputernya sembari menunggu *virus definition/malware signature* terbaru dari pengembang antivirus yang mereka gunakan. Bagi pengembang antivirus sendiri, aplikasi ini dapat dimanfaatkan untuk mengamati perilaku *malware*, sehingga dapat memperbaiki program antivirus mereka agar dapat menanganinya dengan sempurna.

DAFTAR PUSTAKA

Booth, D., et.all, 2004, "Web services Architecture". W3C Working Group Note, dari situs http://www.w3.org/TR/2004/NOTE-ws-arch-20040211, diakses 08 Januari 2008.

Filiol, Eric, 2005, "*Computer* Vir*use*s: from Theory to Applications". France. Springer-Verlag France.

Yurnalis, Widia, 2007, "Tren Web Services Dunia", SDA Asia Online: Indonesia, dari situs http://www.sda-indo.com/sda/features/psecom,id,1654,nodeid,4,_language,Indonesia.html, diakses 12 Januari 2008.

Yurnalis, Widia, 2008, "Tren *Malware*, Spam, dan Virus di 2008", SDA Asia Online: Indonesia, dari situs http://www.sda-indo.com/sda/features/psecom,id,1692,nodeid,4,_language,Indonesia.html, diakses 12 Januari 2008.

Zadel, Mark, et.all, 2004, "Web services for Music Information Retrieval". Proceedings of the 5th International Conference on Music Information Retrieval, dari situs http://ismir2004.ismir.net/proceedings/p087-page-478-paper243.pdf, diakses 08 Januari 2008.

------, [2008a], Amazon Web Services, http://www.amazon.com/AWS-home-page-Money/b/ref=sc_fe_l_1/002-9560265-8122436?ie=UTF8&node=3435361&no=201590011&me=A36L942TSJ2AJA, diakses 08 Januari 2008.