# LAPORAN AKHIR

## PENELITIAN

## PEMBANGUNAN LAYANAN APLIKASI ANTIVIRUS MELALUI JARINGAN INTERNET BERBASIS *WEB SERVICES*

**PENELITIAN LABORATORIUM/LAPANGAN**



**Oleh:**

**Yohanes Sigit Purnomo W.P., S.T., M.Kom.**

**Fakultas Teknologi Industri**

**Universitas Atma Jaya Yogyakarta**

**2008**

# LEMBAR PENGESAHAN

| | | | |
|---|---|---|---|
| 1. | a. Judul Penelitian | : | PEMBANGUNAN LAYANAN APLIKASI ANTIVIRUS MELALUI JARINGAN INTERNET BERBASIS *WEB SERVICES* |
| | b. Macam Penelitian | : | Laboratorium |
| 2. | Peneliti | | |
| | a. Nama | : | Y. Sigit Purnomo W.P., S.T., M.Kom. |
| | b. Jenis Kelamin | : | Laki-laki |
| | c. Usia saat pengajuan proposal | : | 29 tahun 9 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 diajukan | : | 2.950.000,-<br>(Dua juta sembilan ratus lima puluh ribu rupiah) |

Yogyakarta, 03 No vember 2008
Ketua Peneliti,

Y. Sigit Purnomo W.P., ST, M.Kom.

Mengetahui,

Konsultan

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

Ketua P.S. Teknik Informatika UAJY

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

Dekan FTI UAJY,

Paulus Mudjihartono, S.T, M.T.

Ketua LPPM UAJY,

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

# DAFTAR ISI

# INTISARI

Perkembangan teknologi Internet telah memunculkan berbagai macam layanan aplikasi yang dibangun di atas jaringan Internet. Layanan-layanan aplikasi tersebut ada yang dibangun dengan memanfaatkan *Web services*, misal Amazon *Web services,* Google API *(Application Programming Interface)*, Yahoo!, dan eBay. Perkembangan teknologi Internet di satu sisi 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.

*Web services* juga dapat digunakan untuk membangun suatu layanan aplikasi antivirus yang dapat diakses melalui jaringan Internet. Hal ini dilakukan dengan mengekspos fungsi-fungsi yang berkaitan dengan aplikasi antivirus ke dalam *method-method Web services*. Jika fungsi-fungsi tersebut telah diekspos, maka akan memungkinkan pihak lain, khususnya pengembang aplikasi antivirus untuk mengakses layanan tersebut dan membuat aplikasi antivirus sesuai dengan kebutuhannya ataupun sesuai dengan spesifikasi komputer yang dimiliki.

Layanan aplikasi antivirus melalui jaringan Internet berbasis *Web services* telah berhasil dibangun sehingga pihak lain dapat menggunakannya untuk mengembangkan aplikasi antivirus sesuai dengan kebutuhan atau spesifikasi komputer yang dimiliki. Layanan ini juga telah berhasil diakses melalui prototipe aplikasi antivirus berbasis Desktop yang telah dibangun sehingga dapat digunakan untuk melakukan proses *scanning malware*/virus, baik *scanning memory* maupun *scanning files*.

Kata kunci: antivirus, Internet, *malware, signature,* virus, *Web services*

# BAB 1. PENDAHULUAN

## 1.1 Pendahuluan

Perkembangan teknologi Internet telah memunculkan berbagai macam layanan aplikasi yang dibangun di atas jaringan Internet. Layanan-layanan aplikasi tersebut ada yang dibangun dengan memanfaatkan *Web services*, misal Amazon *Web services,* Google API *(Application Programming Interface)*, Yahoo!, dan eBay. Pihak lain, khususnya pengembang aplikasi dapat mengembangkan sistem yang mengakses layanan aplikasi yang telah disediakan tersebut sesuai dengan kebutuhannya. Hal ini dikarenakan *Web services* dibangun berdasarkan standar terbuka sehingga mampu mendukung interoperabilitas interaksi antar mesin yang ada dalam jaringan Internet.

Perkembangan teknologi Internet di satu sisi 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*.

Aplikasi antivirus yang ada saat ini hampir semuanya ter-*bundle* dalam satu paket sehingga pengguna mungkin saja tidak dapat menggunakannya pada komputer yang dimiliki karena spesifikasinya tidak mendukung. Misal, sebuah aplikasi antivirus dibangun di atas .NET Framework, jika pengguna hanya memiliki komputer yang menggunakan sistem operasi LINUX, maka pengguna tersebut tidak dapat menggunakan aplikasi antivirus tersebut. Hal inilah yang melatarbelakangi untuk mengembangkan sebuah layanan aplikasi antivirus melalui jaringan Internet berbasis *Web services* yang memungkinkan pengguna untuk mengembangkan sendiri aplikasinya sesuai dengan kebutuhannya atau dengan spesifikasi mesin yang dimiliki.

## 1.2 Perumusan Masalah

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

1. Bagaimana membangun layanan aplikasi antivirus melalui jaringan Internet berbasis *Web services* sehingga pihak lain dapat menggunakannya untuk mengembangkan aplikasi antivirus sesuai dengan kebutuhan atau spesifikasi komputer yang dimiliki?
2. Bagaimana membangun prototipe aplikasi antivirus dengan memanfaatkan *Web services* yang telah dibangun?

## 1.3 Tujuan Penelitian

Penelitian yang akan dilakukan bertujuan untuk:

1. Membangun suatu layanan aplikasi antivirus melalui jaringan Internet berbasis *Web services*.
2. Membangun suatu prototipe aplikasi antivirus dengan memanfaatkan *Web services* yang telah dibangun .

## 1.4 Manfaat Hasil Penelitian

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

1. Pemahaman tentang proses pengembangan layanan aplikasi antivirus melalui jaringan Internet berbasis *Web services*.
2. Pemahaman tentang pembangunan prototipe aplikasi antivirus berbasis *Web services*.

## 1.5 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*/virus dan penggunaan *Web services* untuk membangun sebuah layanan aplikasi Internet.

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 layanan 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 layanan aplikasi.

5. Pengujian Sistem

   Pengujian sistem dilakukan untuk menguji fungsionalitas layanan aplikasi yang akan dibangun apakah sudah sesuai dengan spesifikasi yang telah ditentukan.

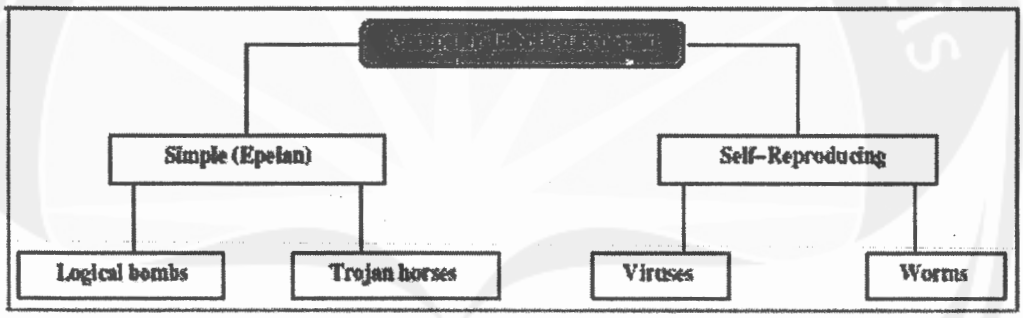
6. Penulisan Laporan dan Dokumentasi

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

# BAB 2. LANDASAN TEORI

## 2.1 *Malware*

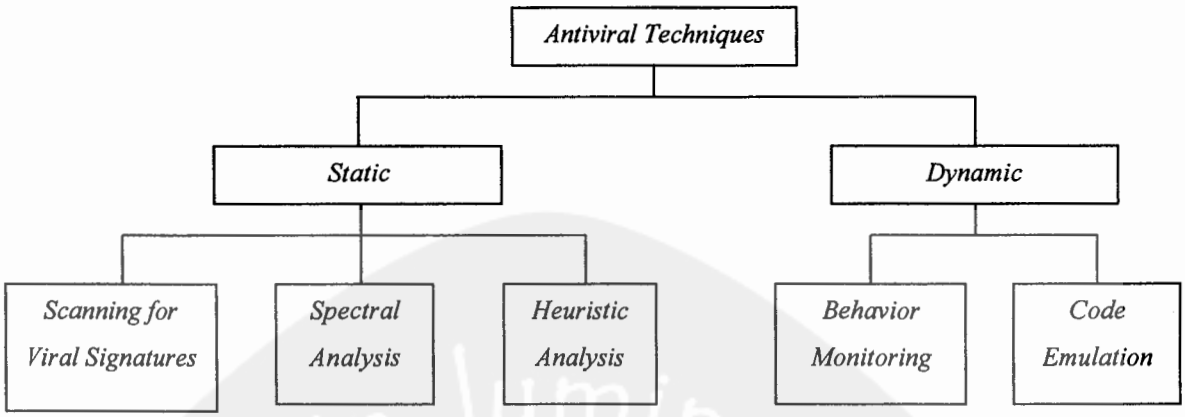
*Computer infection program* atau *malware* didefinisikan sebagai sebuah program sederhana atau program yang dapat mereplikasi dirinya sendiri (*self-replicating*), yang meng-*install* dirinya sendiri secara diskret ke suatu sistem pengolahan data tanpa sepengetahuan pengguna dengan maksud untuk mengancam kerahasiaan data, integritas data dan ketersediaan sistem atau memastikan bahwa pengguna dijebak untuk kejahatan komputer (Filiol, 2005). *Malware* dapat dikategorikan menjadi *malware* sederhana (*malware* yang tidak dapat mereplikasi dirinya sendiri atau *epeian*) dan *malware* yang dapat mereplikasi dirinya sendiri (Gambar 1.2.). *Malware* sederhana dapat digolongan menjadi *logical bomb* dan *trojan horse*, sedangkan *malware* yang dapat mereplikasi dirinya sendiri dapat digolongkan menjadi virus dan *worm*.




Gambar 1.2. Taksonomi *Malware*

(Sumber: Filiol, 2005)

Penanganan *malware* dapat dilakukan dengan menggunakan teknik *antiviral* statis maupun dinamis (Filiol, 2005). Metode statis dapat dilakukan dengan menggunakan teknik *scanning* virus atau *malware signature*, analisis spektral, dan analisis *heuristic*. Sedangkan metode dinamis dapat dilakukan dengan monitoring perilaku atau aktivitas sistem dan emulasi kode program.




Gambar 1.3. *Antiviral Techniques*
(Sumber: Filiol, 2005)

## 2.2 *Web Services*

*Web services* adalah suatu sistem perangkat lunak yang dirancang untuk mendukung interoperabilitas interaksi antar mesin melalui jaringan. *Web services* memiliki suatu antarmuka yang dideskripsikan dalam suatu format yang dapat diproses oleh mesin sehingga sistem yang lain dapat berinteraksi dengan *Web services* menggunakan sebuah mekanisme yang telah ditentukan pada WSDL (*Web service Description Language*) menggunakan pesan SOAP (*Simple Object Access Protocol*), yang biasanya disampaikan menggunakan HTTP (*Hyper-text Transfer Protocol*) dan serialisasi XML serta standar web lainnya yang terkait (Booth, et.all, 2004).

Penggunaan *Web services* menjadi semakin populer pada lingkungan bisnis sehingga mendorong banyak bisnis untuk menambahkan antarmuka *Web services* publik ke basis data mereka dan memungkinkan untuk melakukan akses yang terprogram secara langsung ke antarmuka *Web services* publik tersebut (Zadel, et.all, 2004). Saat ini ada beberapa *Web services* yang dibuka untuk publik, seperti ebay, amazon.com, Google, dan Yahoo! (Yurnalis, 2007). Amazon *Web services* saat ini telah menyediakan berbagai macam *Web services* yang memungkinkan pengembang sistem untuk meningkatkan penggunaan data dan infrastruktur yang dimiliki oleh Amazon dengan mudah dan murah sehingga pengembang sistem dan bisnis eksternal dapat membangun aplikasi webnya dengan handal, mudah diperluas kapasitas layanannya (*scalable*), dan efektif pembiayaannya (2008a, 2008). Layanan-layanan *Web services* yang disediakan oleh Amazon antara lain *Amazon Associates Web Service (*sebelumnya *Amazon ECS), Amazon*

*Elastic Compute Cloud (Beta), Amazon Flexible Payments Service (Beta), Amazon Mechanical Turk (Beta), Amazon SimpleDB (Beta), Amazon Simple Queue Service, Amazon Simple Storage Service, Alexa Site Thumbnail, Alexa Top Sites, Alexa Web Information Service,* dan *Alexa Web Search. Web services* juga telah digunakan untuk menghasilkan kelompok-kelompok artis musik dari basisdata Google dan Amazon berdasarkan metadata kultural (Zadel, et.all, 2004).

# BAB 3. ANALISIS DAN PERANCANGAN SISTEM

Bab ini berisi spesikasi kebutuhan serta perancangan layanan aplikasi antivirus melalui jaringan Internet berbasis *Web services* (selanjutnya disebut wsAV) dan prototipe aplikasinya (selanjutnya disebut wsAVapp). Berdasarkan analisis kebutuhan, wsAV akan menyediakan beberapa fungsionalitas yang dapat diakses oleh pihak lain melalui *Web services*. Fungsionalitas tersebut adalah:

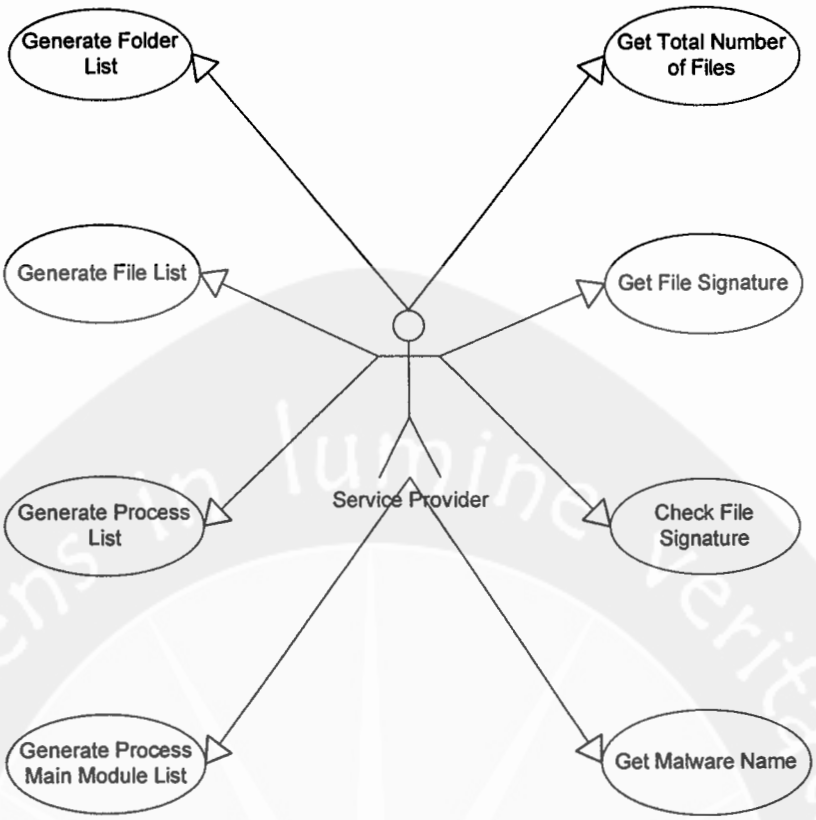
1. Fungsionalitas untuk men-*generate folder list* dari suatu lokasi path.
2. Fungsionalitas untuk men-*generate file list* dari suatu lokasi path.
3. Fungsionalitas untuk menghitung jumlah total file dari suatu lokasi path.
4. Fungsionalitas untuk men-*generate process list* yang aktif di memori dari suatu komputer.
5. Fungsionalitas untuk men-*generate process main module list* dari proses yang aktif di memori dari suatu komputer.
6. Fungsionalitas untuk memperoleh *signature* dari suatu file.
7. Fungsionalitas untuk mengecek apakah file merupakan *malware*/virus atau bukan berdasarkan file *signature*.
8. Fungsionalitas untuk memperoleh nama *malware*/virus berdasarkan *signature*.

Sedangkan wsAVapp, berdasarkan analisis kebutuhan, akan memiliki fungsionalitas untuk melakukan *scanning* terhadap proses yang sedang aktif di memori dan *scanning* terhadap file yang ada dalam suatu lokasi path. Proses *scanning* akan dilakukan dengan memanggil layanan-layanan *Web services* yang telah disediakan dalam wsAV.

Rincian spesifikasi kebutuhan serta rancangan layanan aplikasi antivirus melalui jaringan Internet berbasis *Web services* (wsAV) dan prototipe aplikasinya (wsAVapp) akan dibahas pada subbab-subbab berikut ini.

## 3.1 Spesifikasi Kebutuhan Fungsional

Berdasarkan analisis, kebutuhan fungsionalitas dari wsAV ditunjukkan dengan diagram *use case* pada gambar 3.1., dan deskripsi rincinya ditunjukkan dengan *use case glossary* pada tabel 3.1.



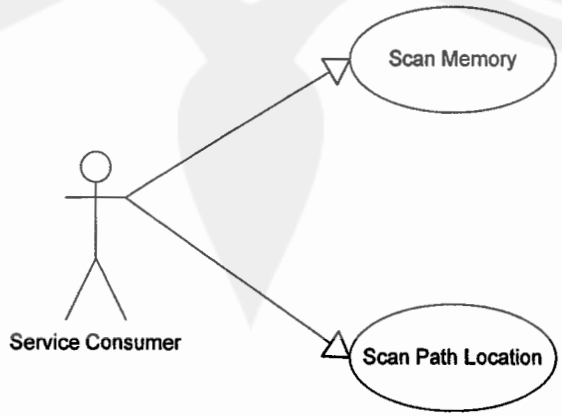

**Gambar 3.1. Diagram *Use Case* wsAV**

**Tabel 3.1. *Use Case* Glossary wsAV**

| *Use case* Name | *Use case* Description |
|---|---|
| Generate Folder List | *Use case* ini menyediakan fungsionalitas bagi *Service Consumer* untuk men-*generate folder list* dari suatu lokasi path. |
| Generate File List | *Use case* ini menyediakan fungsionalitas bagi *Service Consumer* untuk men-*generate file list* dari suatu lokasi path. |
| Generate Process List | *Use case* ini menyediakan fungsionalitas bagi *Service Consumer* untuk men-*generate process list* yang sedang aktif di memori dari suatu komputer. |
| Generate Process Main Module List | *Use case* ini menyediakan fungsionalitas bagi *Service Consumer* untuk men-*generate process main module list* dari proses yang |

| | |
|---|---|
| | sedang aktif di memori pada suatu komputer. |
| Get Total Number of Files | *Use case* ini menyediakan fungsionalitas bagi *Service Consumer* untuk menghitung jumlah total file yang ada pada suatu lokasi path. |
| Get File Signature | *Use case* ini menyediakan fungsionalitas bagi *Service Consumer* untuk mendapatkan *signature* dari suatu file dengan menggunakan metode MD5. |
| Check File Signature | *Use case* ini menyediakan fungsionalitas bagi *Service Consumer* untuk membandingkan apakah *signature* yang diperoleh dari suatu file merupakan *malware* atau bukan. |
| Get Malware Name | *Use case* ini menyediakan fungsionalitas bagi *Service Consumer* untuk mendapatkan nama *malware* dari suatu *malware signature*. |

Berdasarkan analisis, kebutuhan fungsionalitas dari wsAVapp ditunjukkan dengan diagram *use case* pada gambar 3.2., dan deskripsi rincinya ditunjukkan dengan *use case glossary* pada tabel 3.2.




**Gambar 3.2. Diagram *Use case* wsAVapp**

**Tabel 3.2. *Use case* Glossary wsAVapp**

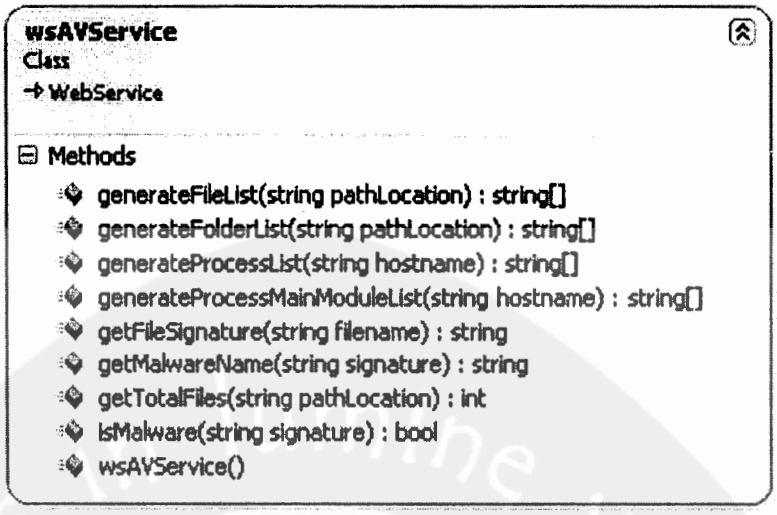
| *Use case* Name | *Use case* Description |
|---|---|
| Scan Memory | *Use case* ini menyediakan fungsionalitas untuk melakukan proses *scanning memory* untuk mendeteksi apakah ada proses aktif di memori yang diaktifkan oleh suatu virus/*malware*. *Use case* ini melibatkan *use case* Generate Process List, Generate Process Main Module List, Get File Signature, Check File Signature, dan Get Malware Name yang disediakan oleh *Service Provider*. |
| Scan Path Location | *Use case* ini menyediakan fungsionalitas untuk melakukan proses *scanning* untuk mendeteksi apakah ada file di suatu lokasi path yang merupakan *malware*. *Use case* ini melibatkan *use case* Generate Folder List, Generate File List, Get Total Number of Files, Get File Signature, Check File Signature, dan Get Malware Name yang disediakan oleh *Service Provider*. |

## 3.2 Spesifikasi Kebutuhan Data

Dari analisis, data yang dibutuhkan untuk disimpan sebagai data *persistent* adalah data entitas *malware* yang terdiri dari atribut *malwareName* dan *malwareSignature* yang keduanya bertipe teks.

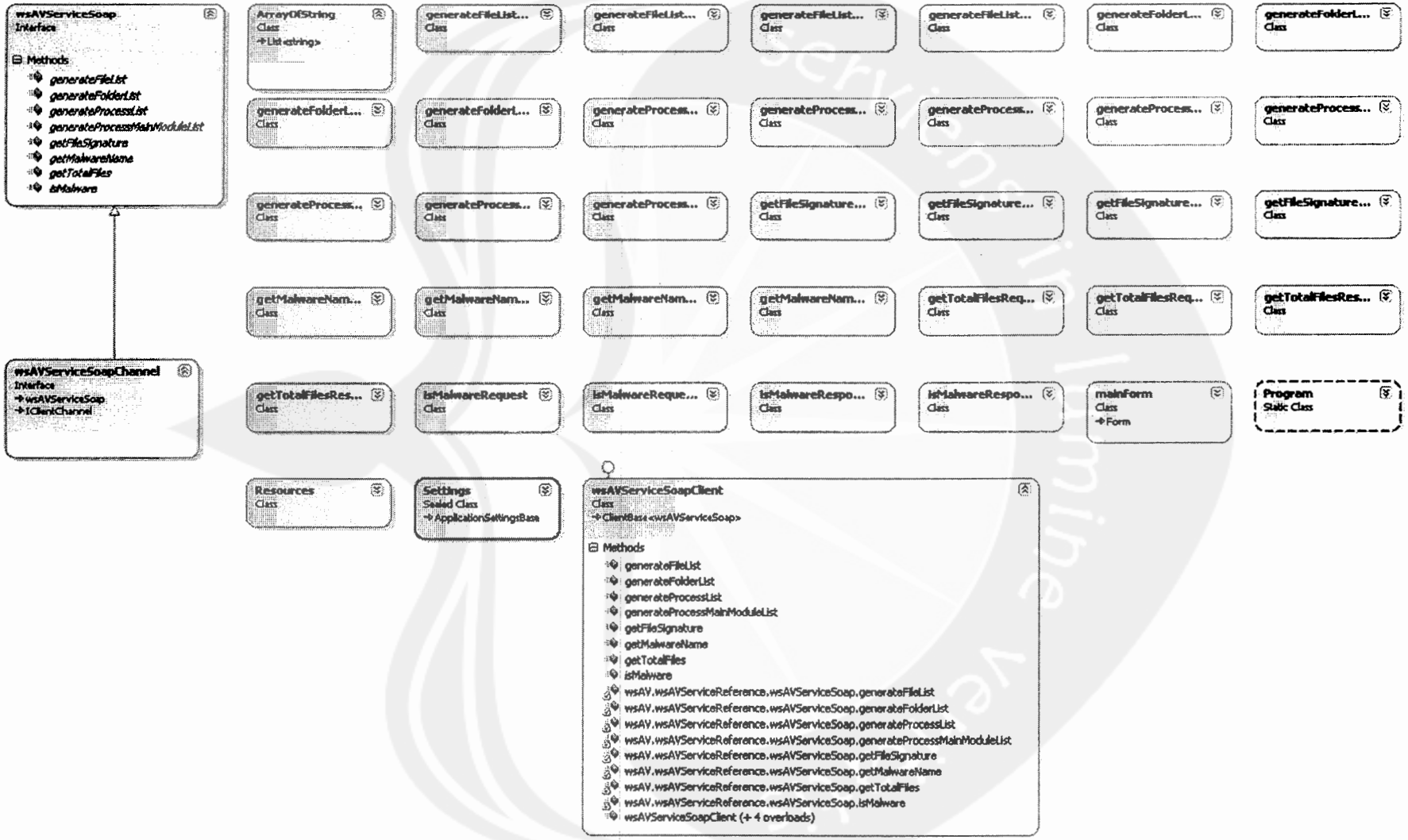
## 3.3 Perancangan Fungsional

Fungsionalitas-fungsionalitas yang dideskripsikan dalam bentuk *use case* pada bab sebelumnya, selanjutnya direalisasi dalam bentuk kelas-kelas yang mengimplementasikan fungsionalitas tersebut. Secara lengkap diagram kelas dari rancangan fungsional wsAV terlihat pada gambar 3.3.




**Gambar 3.3. Diagram Kelas wsAV**

Kelas wsAVService (gambar 3.3.) memiliki sejumlah *method* yang menggambarkan fungsionalitas (*use case*) yang telah dijelaskan pada tabel 3.1. Sedangkan untuk diagram kelas wsAVapp yang merupakan prototipe dari aplikasi antivirus yang menggunakan *Web services* ditunjukkan pada gambar 3.4.




**Gambar 3.4. Diagram Kelas wsAVapp**

## 3.4 Perancangan Data

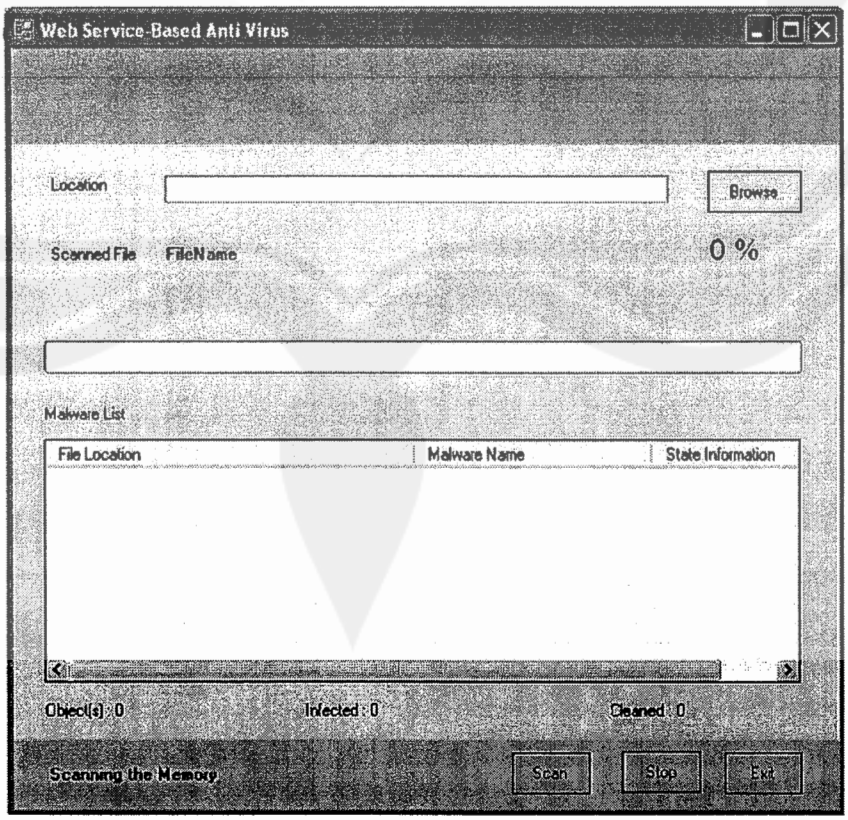
Selanjutnya, berdasarkan analisis kebutuhan data yang telah dilakukan sebelumnya, maka dilakukan perancangan tabel data sebagai berikut:

**Tabel *Malware***

| Nama Elemen | Tipe Data | Range Nilai |
|---|---|---|
| *malware*Name | nvarchar(255) | [a-z,A-Z,0-9,.,-] |
| *malware*Siganture | nvarchar(255) | [a-z,A-Z,0-9] |

## 3.5 Perancangan Arsitektur Navigasi Antar Muka Pengguna

Berikutnya akan dibahas perancangan arsitektur navigasi antar muka pengguna aplikasi wsAVapp. Untuk wsAV tidak ada arsitektur navigasi antar muka pengguna karena yang disediakan hanya *Web services* saja dan tidak memiliki antar muka pengguna. Arsitektur navigasi antar muka pengguna aplikasi wsAVapp hanya terdiri dari satu form utama saja, dimana di dalam form utama tersebut terdapat fungsi untuk memilih lokasi path yang akan discan, fungsi untuk melakukan scan, fungsi untuk menghentikan proses scan, dan fungsi untuk keluar dari aplikasi. Rancangan antar muka pengguna aplikasi wsAVapp adalah sebagai berikut:

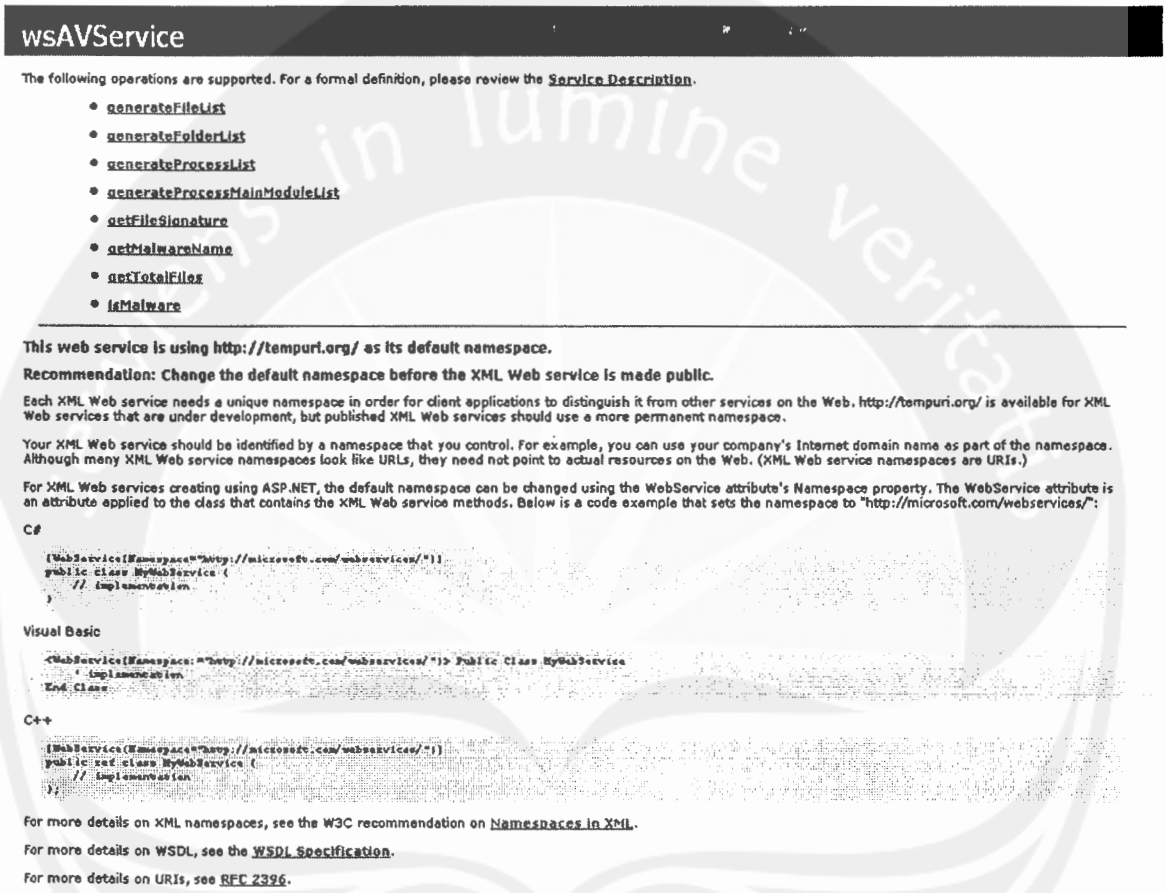



**Gambar 3.5. Rancangan Antar Muka Pengguna Aplikasi wsAVapp**

# BAB 4. IMPLEMENTASI SISTEM

## 4.1 Implementasi *Web Services* wsAV

Bab ini menjelaskan implementasi dari layanan aplikasi wsAV. Masing-masing fungsionalitas yang diimplementasikan ke dalam bentuk *Web services* akan diuji terlebih dahulu sebelum diakses melalui prototipe aplikasi wsAVapp.



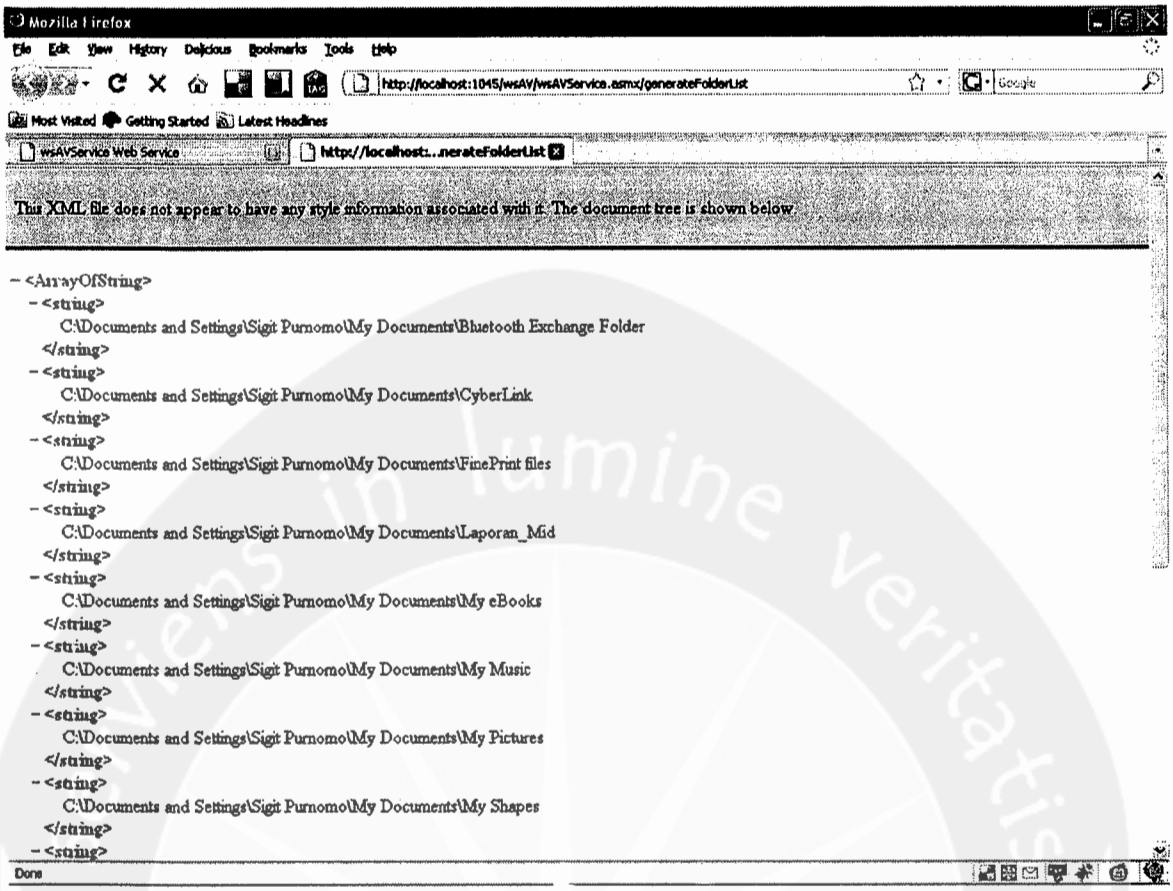

**Gambar 4.1. Halaman Utama Pengujian Web Services wsAV**

Gambar 4.1 menunjukkan halaman utama yang berisi informasi mengenai *method-method* apa saja yang disediakan oleh *Web services* wsAV. Halaman ini berfungsi untuk melakukan pengujian terhadap setiap *method* dari *Web services* apakah berfungsi dengan benar atau tidak. Untuk melakukan pengujian terhadap setiap *method* dapat dilakukan dengan mengklik nama *method* yang ada.

### 4.1.1 Pengujian *Method* generateFolderList

*Method* generateFolderList menyediakan fungsionalitas bagi *Service Consumer* untuk men-*generate folder list* dari suatu lokasi path. *Method* ini menerima parameter masukan berupa lokasi path dan kemudian akan menghasilkan array string yang berisi *folder list* dari lokasi path tersebut. Pengujian dan hasilnya dapat dilihat pada Gambar 4.2. dan Gambar 4.3.


**Gambar 4.2. Halaman Pengujian *Method* generateFolderList**

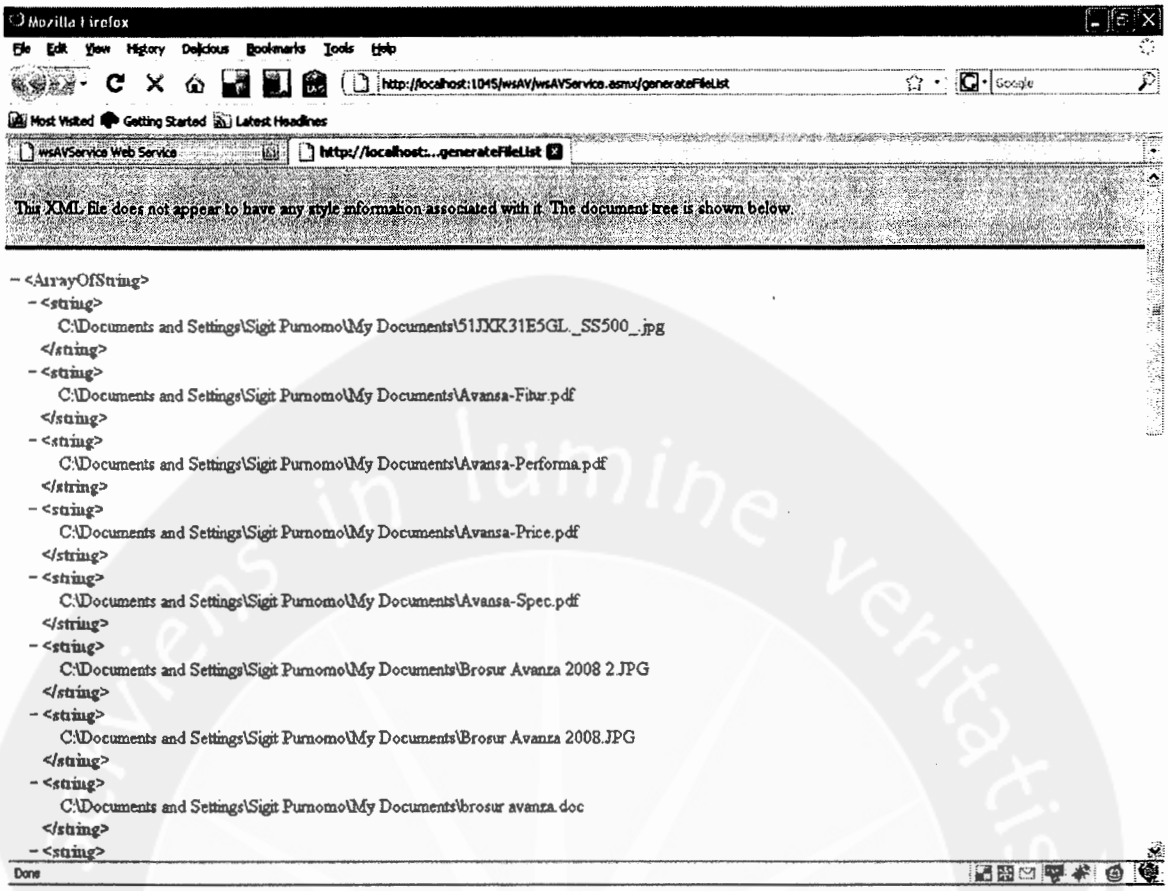



**Gambar 4.3. Hasil Pengujian *Method* generateFolderList**

### 4.1.2 Pengujian *Method* generateFileList

*Method* generateFileList menyediakan fungsionalitas bagi *Service Consumer* untuk men-*generate file list* dari suatu lokasi path. *Method* ini menerima parameter masukan berupa lokasi path dan kemudian akan menghasilkan array string yang berisi *file list* dari lokasi path tersebut. Pengujian dan hasilnya dapat dilihat pada Gambar 4.4. dan Gambar 4.5.


**Gambar 4.4. Halaman Pengujian *Method* generateFileList**

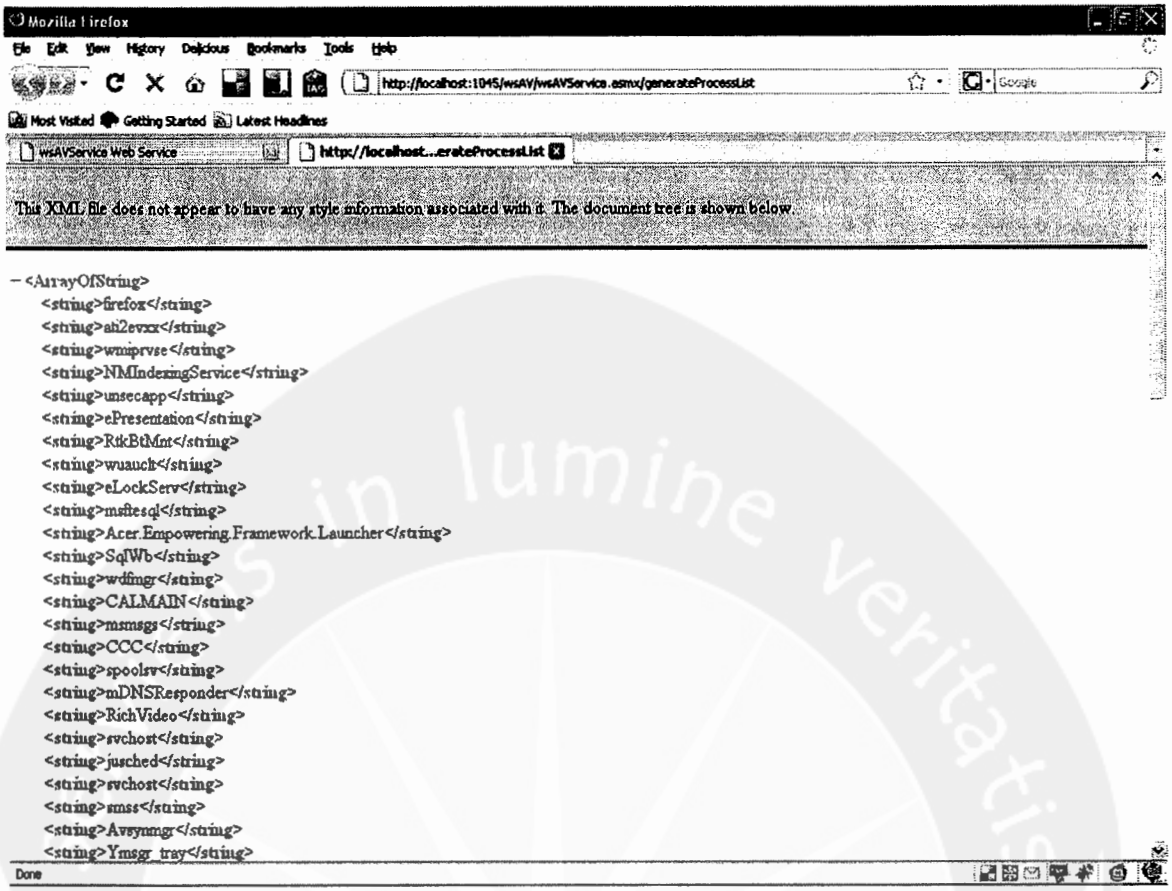



**Gambar 4.5. Hasil Pengujian *Method* generateFileList**

### 4.1.3 Pengujian *Method* generateProcessList

*Method* generateProcessList menyediakan fungsionalitas bagi *Service Consumer* untuk men-*generate process list* yang sedang aktif di memori dari suatu komputer. *Method* ini menerima parameter masukan berupa nama komputer dan kemudian akan menghasilkan array string yang berisi *process list* yang sedang aktif di memori dari komputer tersebut. Pengujian dan hasilnya dapat dilihat pada Gambar 4.6. dan Gambar 4.7.


**Gambar 4.6. Halaman Pengujian *Method* generateProcessList**




**Gambar 4.7. Hasil Pengujian *Method* generateProcessList**

### 4.1.4 Pengujian *Method* generateProcessMainModuleList

*Method* generateProcessMainModuleList menyediakan fungsionalitas bagi *Service Consumer* untuk men-*generate process main module list* dari proses yang sedang aktif di memori pada suatu komputer. *Method* ini menerima parameter masukan berupa nama komputer dan kemudian akan menghasilkan array string yang berisi *process main module list* dari proses yang sedang aktif di memori komputer tersebut. Pengujian dan hasilnya dapat dilihat pada Gambar 4.8. dan Gambar 4.9.

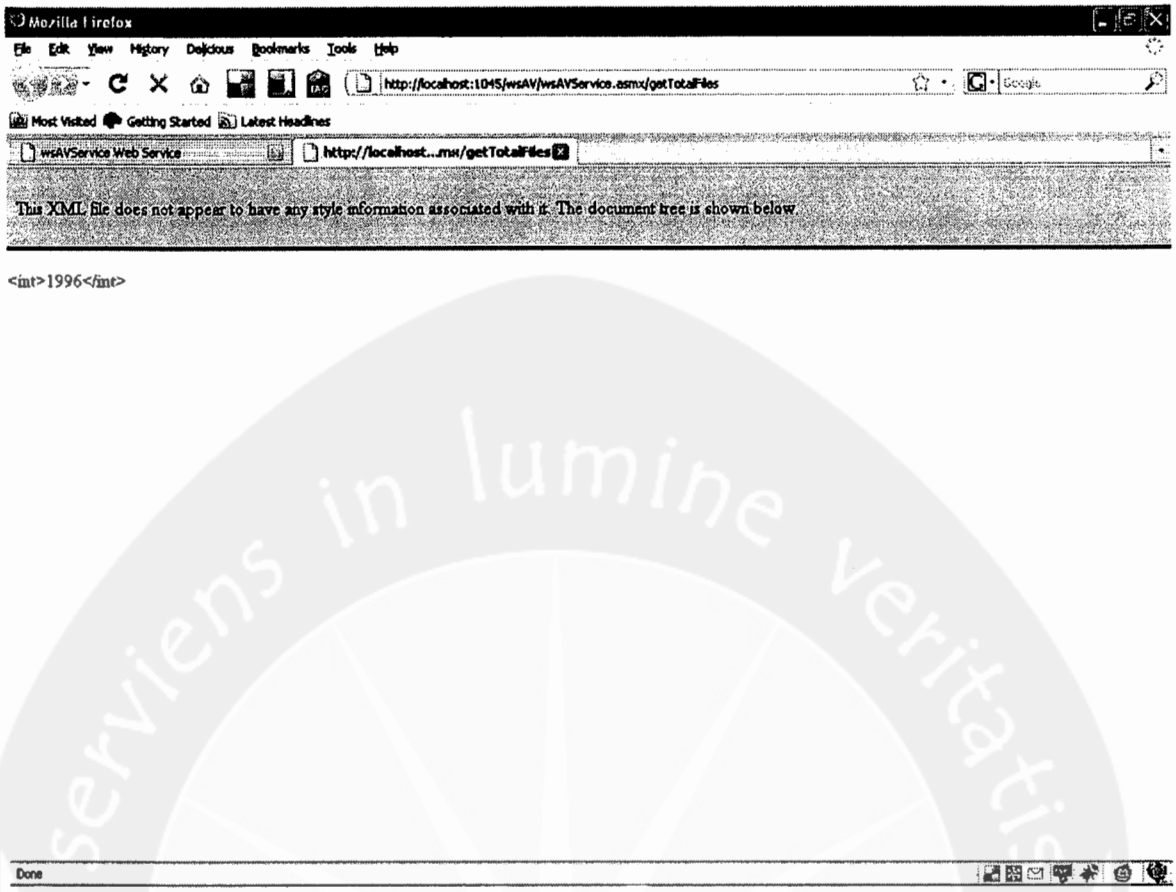

**Gambar 4.8. Halaman Pengujian *Method* generateProcessMainModuleList**


**Gambar 4.9. Hasil Pengujian *Method* generateProcessMainModuleList**

### 4.1.5 Pengujian *Method* getTotalFiles

*Method* getTotalFiles menyediakan fungsionalitas bagi *Service Consumer* untuk menndapatkan jumlah total file dari suatu lokasi path. *Method* ini menerima parameter masukan berupa lokasi path dan kemudian akan menghasilkan suatu nilai integer yang merupakan jumlah total file dari lokasi path tersebut. Pengujian dan hasilnya dapat dilihat pada Gambar 4.10. dan Gambar 4.11.

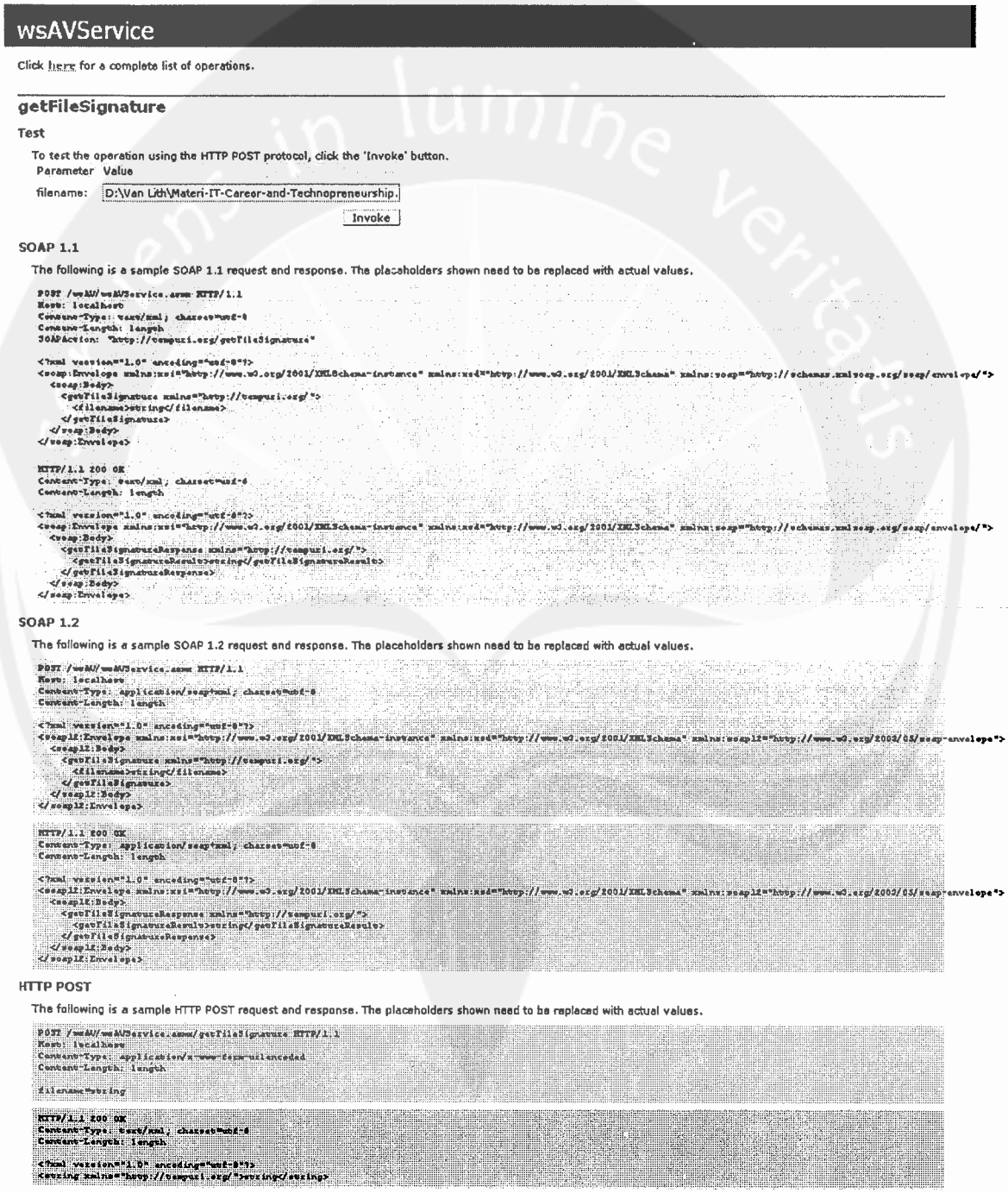

**Gambar 4.10. Halaman Pengujian *Method* getTotalFiles**



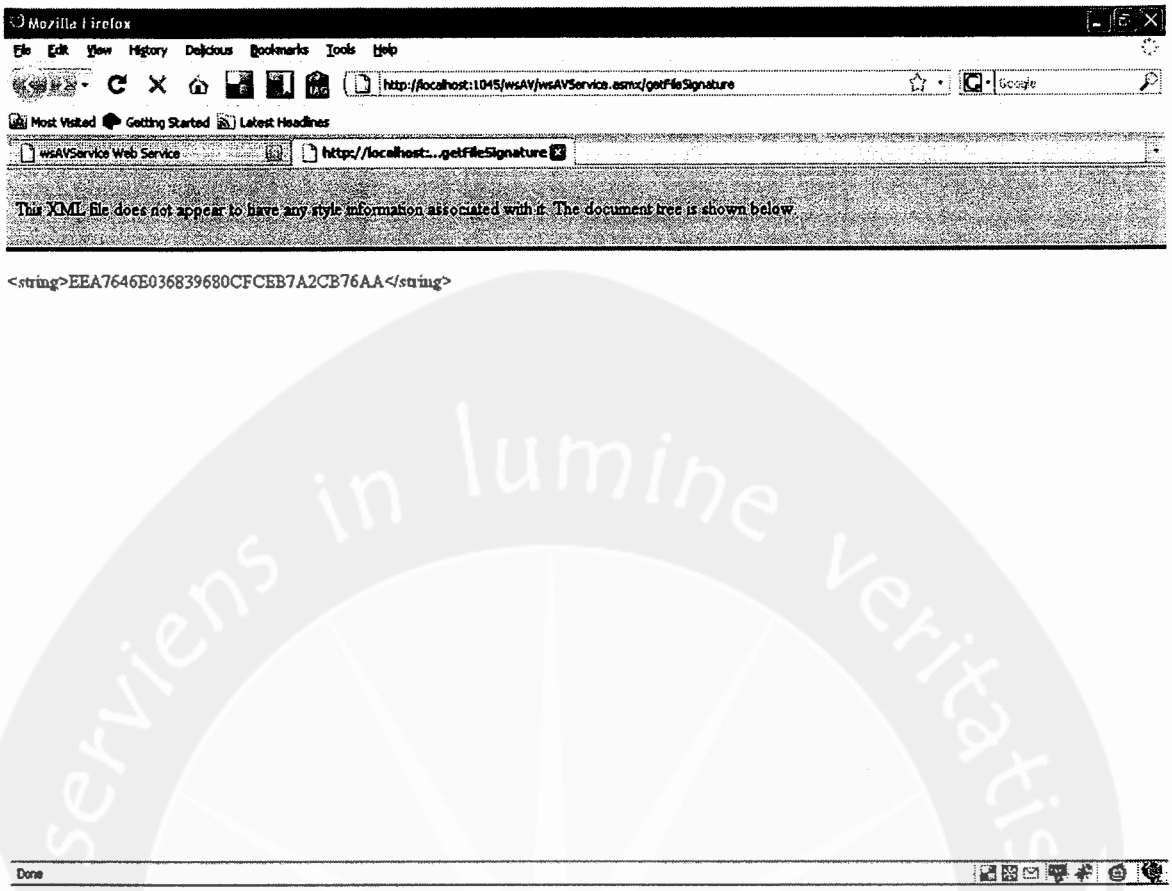

**Gambar 4.11. Hasil Pengujian *Method* getTotalFiles**

### 4.1.6 Pengujian *Method* getFileSignature

*Method* getFileSignature menyediakan fungsionalitas bagi *Service Consumer* untuk men-*generate signature* dari suatu file dengan menggunakan algoritma MD5. *Method* ini menerima parameter masukan berupa nama file dan kemudian akan menghasilkan string yang berisi *signature* dari file tersebut. Pengujian dan hasilnya dapat dilihat pada Gambar 4.12. dan Gambar 4.13.




**Gambar 4.12. Halaman Pengujian *Method* getFileSignature**

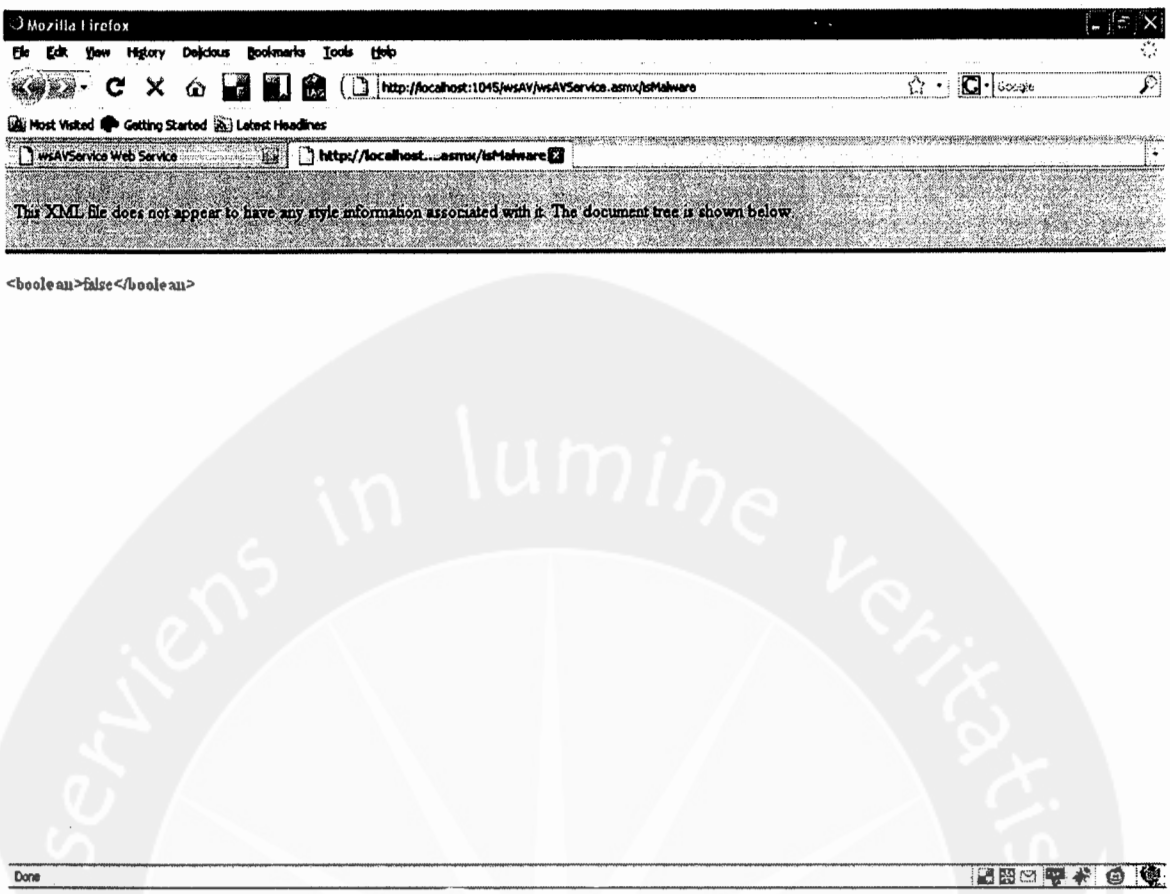



**Gambar 4.13. Hasil Pengujian *Method* getFileSignature**

### 4.1.7 Pengujian *Method* isMalware

*Method* isMalware menyediakan fungsionalitas bagi *Service Consumer* untuk mendeteksi apakah *signature* dari suatu file merupakan *malware* atau bukan. *Method* ini menerima parameter masukan berupa *signature* dari suatu file dan kemudian akan menghasilkan nilai *true* jika *signature* merupakan *malware* dan *false* jika bukan merupakan *malware*. Pengujian dan hasilnya dapat dilihat pada Gambar 4.14. dan Gambar 4.15.


**Gambar 4.14. Halaman Pengujian *Method* isMalware**

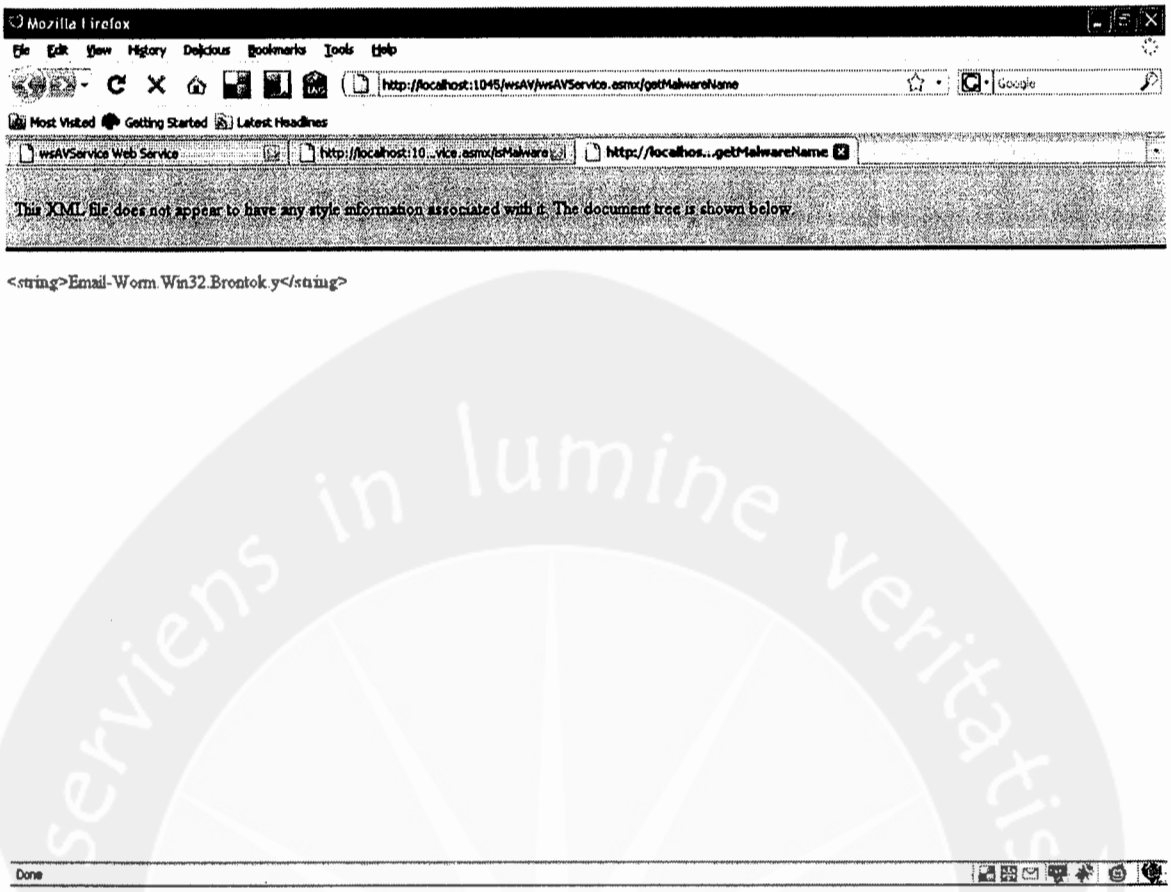



**Gambar 4.15. Hasil Pengujian *Method* isMalware**

### 4.1.8 Pengujian *Method* getMalwareName

*Method* getMalwareName menyediakan fungsionalitas bagi *Service Consumer* untuk memperoleh nama *malware* berdasarkan *signature*-nya. *Method* ini menerima parameter masukan berupa *signature malware* dan kemudian akan menghasilkan string yang berisi nama *malware* dari *signature* tersebut. Pengujian dan hasilnya dapat dilihat pada Gambar 4.16. dan Gambar 4.17.


**Gambar 4.16. Halaman Pengujian *Method* getMalwareName**




**Gambar 4.17. Hasil Pengujian *Method* getMalwareName**

## 4.2 Implementasi Prototipe Aplikasi wsAVapp

Bab ini menjelaskan implementasi dari prototipe aplikasi wsAVapp yang memiliki fungsionalitas untuk melakukan *scanning memory* dan *scanning path location* dengan memanfaatkan *method-method* yang disediakan oleh *Web services* wsAV.

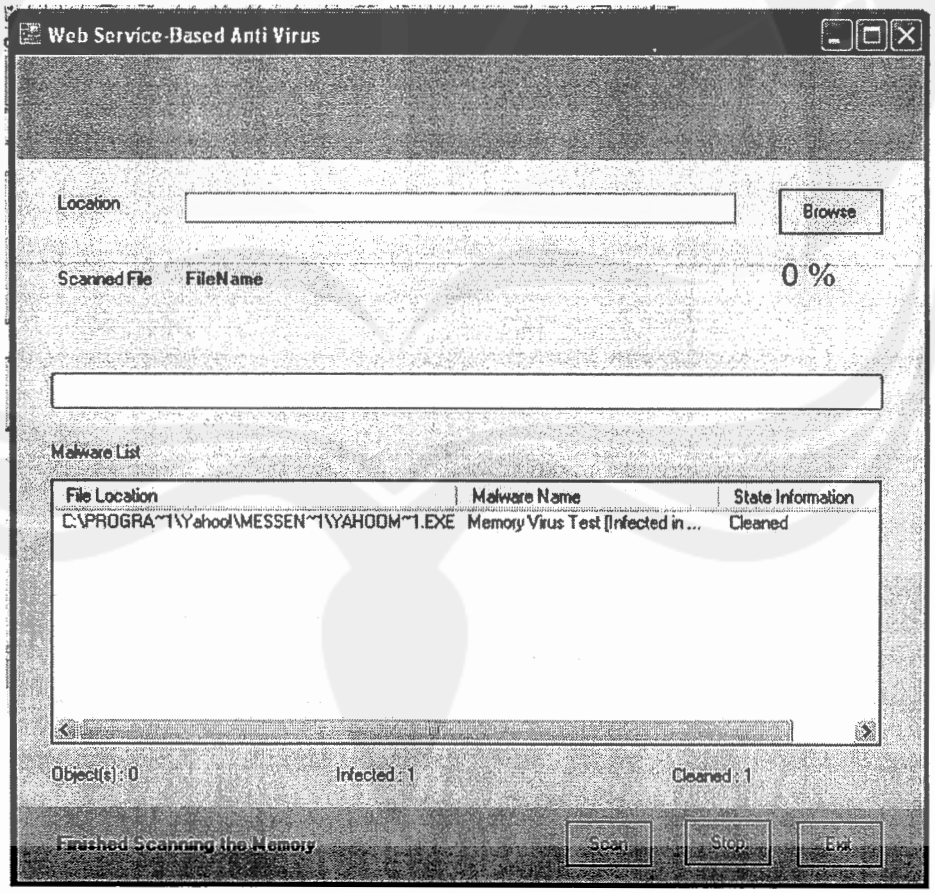
### 4.2.1 Pengujian Fungsi *Scanning Memory*

*Scanning memory* dilakukan untuk mendeteksi apakah ada proses di memori yang diaktifkan oleh suatu *malware*. Pada pengujian ini, akan digunakan aplikasi Yahoo! Messenger yang *signature process main module*-nya telah disimpan ke dalam tabel *malware* sehingga jika proses ini aktif di memori maka akan dideteksi sebagai *malware*. Hasil pengujian dapat dilihat pada Gambar 4.18.

Proses *scanning memory* akan dipanggil pada saat aplikasi wsAVapp dijalankan. Algoritma proses *scanning memory* adalah sebagai berikut:

1. Membaca informasi nama komputer.

2. Memanggil fungsi *Web services* generateProcessList dan generateProcessMainModuleList dengan parameter nama komputer untuk memperoleh daftar proses yang sedang aktif di memori beserta *main module* dari setiap proses.
3. Untuk setiap proses yang ada di dalam daftar, dilakukan hal sebagai berikut:
    a. mencari *signature main modul* proses dengan pemanggilan fungsi *Web services* getFileSignature dengan parameter *main module* dari setiap proses,
    b. membandingkan *signature* yang diperoleh dengan basisdata *malware*, apakah *signature* tersebut merupakan *malware* melalui pemanggilan fungsi *Web services* isMalware dengan parameter *signature* yang diperoleh,
    c. jika *signature* teridentifikasi sebagai *malware*, maka langkah selanjutnya adalah menon-aktifkan proses tersebut dari memori, kemudian menghapus *main module* proses dari media penyimpan.



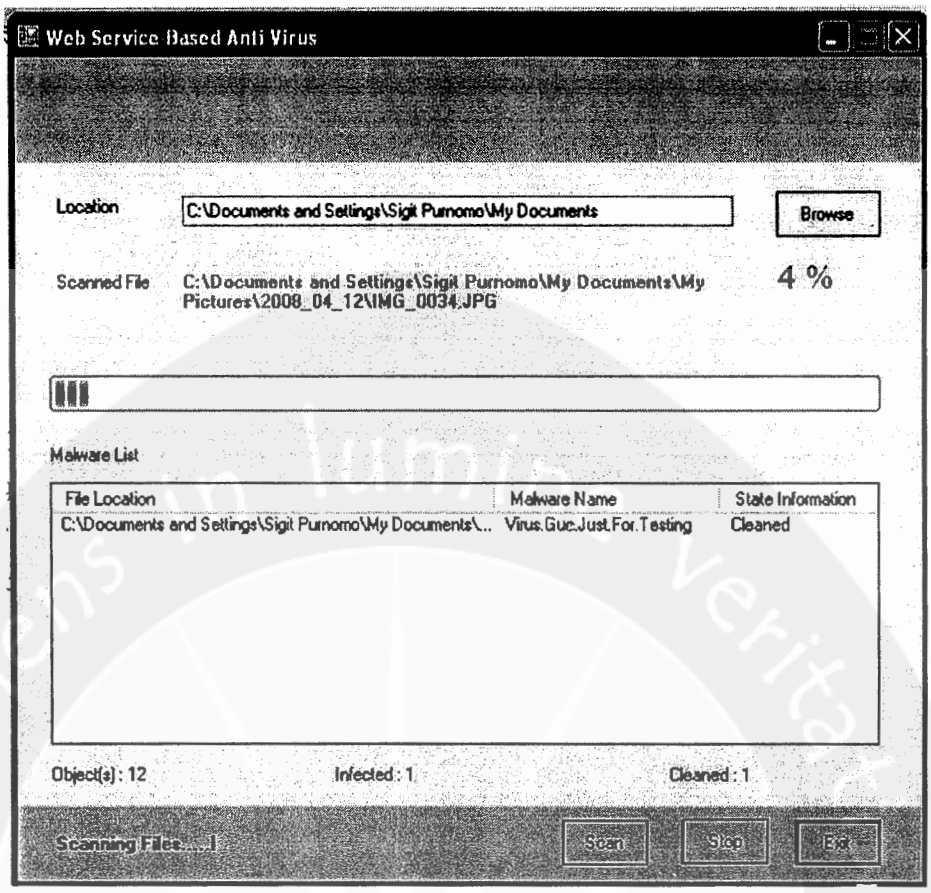

**Gambar 4.18. Hasil Pengujian Fungsi *Scanning Memory***

### 4.2.2 Pengujian Fungsi *Scanning Path Location*

*Scanning path location* dilakukan untuk mendeteksi apakah ada file di lokasi path yang merupakan *malware*. Pada pengujian ini, akan digunakan file Materi-IT-Career-and-Technopreneurship.ppt yang disimpan di folder My Documents yang *signature*-nya telah disimpan ke dalam tabel *malware* sebagai *malware* Virus.Gue.Just.For.Testing. Hasil pengujian dapat dilihat pada Gambar 4.19.

Proses *scanning path location* akan dipanggil pada saat pengguna aplikasi wsAVapp menekan tombol Scan. Sebelum melakukan proses ini, pengguna harus memilih terlebih dahulu lokasi path yang akan di-*scan* dengan menekan tombol Browse. Algoritma proses *scanning path location* adalah sebagai berikut:

1. Memanggil fungsi *Web services* generateFolderList dan generateFileList dengan parameter lokasi path untuk memperoleh daftar folder dan file yang ada pada lokasi path tersebut.
2. Untuk setiap file yang ada di dalam daftar, dilakukan hal sebagai berikut:
    a. mencari *signature* file dengan pemanggilan fungsi *Web services* getFileSignature dengan parameter nama file,
    b. membandingkan *signature* yang diperoleh dengan basisdata *malware*, apakah *signature* tersebut merupakan *malware* melalui pemanggilan fungsi *Web services* isMalware dengan parameter *signature* yang diperoleh,
    c. jika *signature* teridentifikasi sebagai *malware*, maka langkah selanjutnya adalah menghapus file dari media penyimpan.
3. Langkah 1 dan 2 akan dijalankan juga secara rekursif pada setiap folder yang ada dalam daftar folder.




**Gambar 4.18. Hasil Pengujian Fungsi *Scanning Path Location***

## 4.3 Analisis Kelebihan dan Kekurangan Sistem

Pemanfaatan *Web services* untuk membangun suatu layanan aplikasi antivirus melalui jaringan Internet memberikan berbagai macam kelebihan. Kelebihan tersebut, di antaranya yaitu, *Web services* yang memiliki fungsi-fungsi yang berhubungan dengan aplikasi antivirus dapat digunakan oleh pengembang aplikasi untuk membangun aplikasi antivirus sendiri baik berbasis Desktop, Web maupun Mobile. Khusus untuk penelitian ini, prototipe aplikasi antivirus yang dibuat adalah berbasis Desktop. Kelebihan ini juga memungkinkan pengembang aplikasi antivirus untuk mengembangkan aplikasi antivirus sesuai dengan spesifikasi komputer yang dimiliki.

Selain kelebihan di atas, layanan aplikasi antivirus berbasis *Web services* juga akan membuat aplikasi yang dibuat oleh pengembang aplikasi selalu memiliki basisdata *malware*/virus *signature* yang mutakhir. Hal ini disebabkan karena aplikasi yang dibuat akan selalu mengakses basis data *malware* yang dikelola oleh penyedia layanan dan tersimpan di server. Kondisi ini membuat pengguna aplikasi antivirus tidak perlu

melakukan pemutakhiran basisdata *malware*/virus *signature* seperti pada aplikasi antivirus lainnya.

Selain memiliki beberapa kelebihan, pemanfaatan *Web services* untuk membangun suatu layanan aplikasi antivirus juga memiliki kekurangan, yaitu, memaksa pengguna untuk memiliki konektivitas ke Internet. Hal ini disebabkan karena aplikasi yang digunakan/dibangun akan selalu melakukan akses ke *Web services* yang telah di-*publish* di Web Server. Kekurangan ini, di masa depan diharapkan tidak akan menjadi masalah karena diasumsikan konektivitas Internet dapat dengan mudah dimiliki oleh pengguna/pengembang aplikasi yang memanfaatkan *Web services* ini.

Kekurangan lainnya dari pemanfaatkan *Web services* untuk membangun suatu layanan aplikasi antivirus adalah kinerja proses *scanning* dapat menjadi lambat. Hal ini dikarenakan pada saat melakukan proses *scanning*, aplikasi harus melakukan akses ke Web Server melalui jaringan Internet, sehingga lamanya waktu akses sulit diperkirakan. Jika akses Internetnya cepat, maka dapat membantu untuk meningkatkan kecepatan proses *scanning*. Jika akses Internetnya lambat, maka proses *scanning* juga akan menjadi relatif lebih lambat.

# BAB 5. KESIMPULAN

Layanan aplikasi antivirus melalui jaringan Internet berbasis *Web services* telah berhasil dibangun sehingga pihak lain dapat menggunakannya untuk mengembangkan aplikasi antivirus sesuai dengan kebutuhan atau spesifikasi komputer yang dimiliki. Layanan ini juga telah berhasil diakses melalui prototipe aplikasi antivirus berbasis Desktop yang telah dibangun sehingga dapat digunakan untuk melakukan proses *scanning* virus/*malware*, baik *scanning memory* maupun *scanning path location.*

Pengujian pada layanan aplikasi antivirus berbasis *Web services* dan prototipe aplikasi antivirus yang dibuat baru sebatas *alpha testing*, sehingga untuk menguji *feasibilitas* dan skalabilitas yang sesungguhnya, dibutuhkan pengujian lanjutan (*beta testing*) langsung dari pengguna Internet. Hal ini tidak dimungkinkan untuk durasi dan batasan penelitian ini (6 bulan). Penelitian-penelitian selanjutnya dapat difokuskan untuk mempublikasi layanan ini pada Internet dan mendapatkan umpan balik untuk memperbaiki kesalahan dan penambahan fitur-fitur baru yang dikehendaki oleh pengguna.

# 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 Viruses: 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.**

# LEMBAR PENGESAHAN

| | | | |
|---|---|---|---|
| 1. | a. Judul Penelitian | : | PEMBANGUNAN LAYANAN APLIKASI ANTIVIRUS MELALUI JARINGAN INTERNET BERBASIS *WEB SERVICES* |
| | b. Macam Penelitian | : | Laboratorium |
| 2. | Peneliti | | |
| | a. Nama | : | Y. Sigit Purnomo W.P., S.T., M.Kom. |
| | b. Jenis Kelamin | : | Laki-laki |
| | c. Usia saat pengajuan proposal | : | 29 tahun 9 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 diajukan | : | 2.950.000,-<br>(Dua juta sembilan ratus lima puluh ribu rupiah) |

Yogyakarta, 03 November 2008
Ketua Peneliti,

Y. Sigit Purnomo W.P., ST, M.Kom.

Mengetahui,

| | |
|---|---|
| Konsultan | Pj. Kepala Lab Jaringan Komputer |
| Prof. Ir. F. Soesianto, B.Sc.E, Ph.D | Kusworo Anindito, S.T., M.T. |
| Dekan FTI UAJY, | Ketua LPPM UAJY, |
| Paulus Mudjihartono, S.T, M.T. | Ir. B. Kristyanto M.Eng., Ph.D. |