APLIKASI PENGEKSTRAK POLA DAN PENDETEKSI WORMS DENGAN PENDEKATAN PELACAKAN ENTRY POINT

Hendra, S.T., M.T.

STMIK IBBI, Jl. Sei Deli No. 18 Medan
Email : hendra.soewarno@gmail.com

Abstrak

Perkembangan worms lokal telah menjadi salah satu gangguan pemanfaatan teknologi komputer di tanah air, karena fasilitas untuk pendeteksian dan penghapusan worms lokal tidak tersedia dengan cepat pada antivirus komersial. Makalah ini menawarkan suatu metode pengekstrak pola dari suatu file atau process worms dengan pendekatan pelacakan entry point. Pola yang dihasilkan kemudian digunakan untuk mendeteksi keberadaan worms tersebut dimemori maupun media penyimpanan. Metode pengekstrak pola ini akan diaplikasikan pada suatu removal tools yang memungkinkan pemakai untuk menambah data definisi worms secara mandiri.

Kata Kunci : Worms, entry point, removal tools

1. Pendahuluan

Pada jaman komputasi bergerak saat ini, pertukaran data dapat dilakukan secara praktis melalui jaringan maupun menggunakan media seperti *flashdisk*. Pertukaran data yang tidak dilakukan secara hati-hati menjadi target dari infeksi *malware*. Pada akhir-akhir ini perkembangan *worms* lokal telah menunjukan peningkatan yang nyata, dan salah satu worms yang dikenal dengan nama Brontok maupun varian-nya telah berkembang menjadi suatu permasalahan bagi pelaku teknologi informasi ditanah air

Pada umumnya, worms lokal melakukan penyerangan secara massif dengan mereplikasi dirinya pada media penyimpanan, dan mengaktifkan dirinya dengan menggunakan fasilitas autorun. Berbagai pendekatan metode pendeteksian telah dikembangkan baik secara pola maupun pendekatan heuristic. Metode pendeteksian secara pola membutuhkan tersedia pola untuk mengenali worms tersebut, dalam hal ini berarti bahwa munculnya suatu worms baru belum tentu dapat dideteksi dan dinetralkan oleh perangkat anti malware yang tersedia, karena anti malware tersebut belum memiliki pola yang dapat digunakan untuk mendeteksi worms tersebut, sedangkan pendekatan heuristic sendirinya sering menjadi masalah tersendiri karena menyebabkan *false alarm* yang berakibat tidak berfungsinya sistim aplikasi yang dicurigai sebagai *malware*.

Masalah pada penelitian ini adalah bagaimana rancangan metoda ekstraksi pola dari suatu process maupun file worms yang dapat diaplikasikan pada suatu *removal tools* yang memungkinkan pemakaian untuk menambahkan pola worms secara mandiri.

Secara umum tujuan penelitian ini adalah mengembangkan metode pengekstrak pola worms, dan secara khusus tujuan penelitian ini adalah membuat prototipe *removal tools* memungkinkan pemakaian untuk menambah data definisi worms secara mandiri.

Penelitian ini akan dibatasi pada worms yang memiliki format Portable executable (PE) pada platform system operasi Windows, serta tidak memiliki kemampuan polymorphis pada bagian *entry point*. Pada penelitian ini menggunakan asumsi bahwa pemakai dari aplikasi removal dapat mengenali file maupun process worms dan memasukannya kedalam removal tools untuk ekstraksi polanya.

2. Pembahasan

Worms komputer adalah program komputer yang melakukan replikasi dirinya, menyerupai suatu virus komputer. Suatu virus memasukan dirikan ke suatu program executable dan menjadi bagian dari executable tersebut; tetapi suatu worms adalah berdiri sendiri dan tidak perlu menjadi bagian dari program lain untuk menyebarkan dirinya.

Worms sering kali dirancang untuk melakukan ekploitasi terhadap kemampuan transmisi file yang biasanya tersedia pada banyak sistem komputer. Perbedaan utama antara virus dengan worms adalah, virus tidak menyebarkan dirinya, sedangkan worms melakukan hal tersebut.

Worms memanfaatkan jaringan untuk mengirim dirinya ke sistem lain dengan tanpa intervensi pemakai. Umumnya worms mengganggu jaringan dan menghabiskan bandwidth, sedangkan virus melakukan infeksi dan merusak file pada komputer target.

2.1. Pola Pendeteksian

Pendeteksian adakah suatu proses dari pencarian atau penemuan sumber sebenarnya atau masalah dari skenario yang mungkin, dan pola adalah suatu cara regular dimana suatu skenario tertentu terjadi. Pola pendeteksian adalah penting didalam melakukan investigasi untuk mendapatkan bukti-bukti yang terjadi pada suatu kejadian kejahatan.

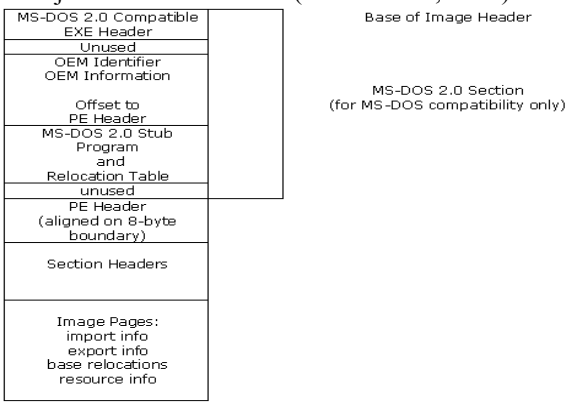
Pada dunia komputer pola pendeteksian berupa data digital karena pada dasarnya komputer adalah mesin yang melakukan proses terhadap data dari masukan dan mengeluarkan data sebagai keluaran dari hasil proses. Worms komputer sendirinya adalah merupakan sekumpulan dari instruksi yang dikenal oleh komputer, instruksi-instruksi tersebut tersimpan sebagai sebagai suatu file yang dapat dieksekusi oleh sistim operasi.

Pendeteksian terhadap Worms membutuhkan pola yang merupakan ekstraksi dari data Worms itu sendirinya, sehingga dapat digunakan sebagai pola untuk mendeteksi worms tersebut dengan metode seperti pelacakan string, *wildcards* atau *regular expressions*, *bookmarks* atau *check bytes*, pelacakan awal dan akhir, pelacakan *entry-point* dan *fixed-point*

2.2. Struktur File PE

Untuk menjelaskan bagaimana teknik pengambilan pola yang dikembangkan oleh penulis, maka berikut ini akan dilakukan tinjauan terhadap struktur file program worms.

Sebagaimana program executable lainnya, struktur file worms pada juga menggunakan format Portable executable (PE). PE adalah format dari program-program binary (exe, dll, sys, scr) untuk MS windows NT, windows 95 dan Win32s. Suatu PE file terdiri dari beberapa bagian informasi yang ditunjukan oleh Gambar 1. (Matt Pietrek, 2002)



Gambar 1. Struktur File PE

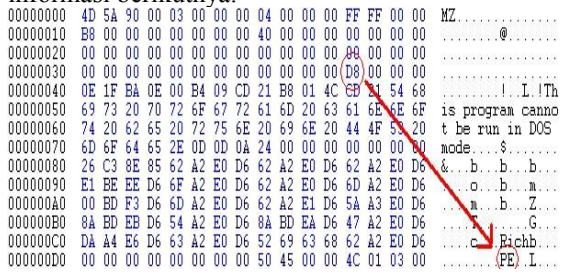
2.2.1 MS – DOS Header

Untuk menjaga kompatibilitas, suatu PE file diawali dengan DOS header, dimana kalau suatu program Win32 PE dieksekusi pada lingkungan 16-bit DOS akan segera dihentikan dengan suatu pesan "This program cannot run in DOS mode"

Gambar 2. Daftar Hexa dari suatu MS-DOS header (Sumber Matt Pietrek, 2002)

2.2.2 PE Header

Pada offset ke 60 dari posisi awal Dos header terdapat sebuah pointer (e_lfanew) ke Portable Executable (PE) File header. Jika suatu PE file dijalankan pada lingkungan DOS-16 akan mencetak pesan error dan berhenti, sedangkan Windows akan mengikuti pointer ini untuk menuju ke bagian informasi berikutnya.



Gambar 3. Daftar Hexa dari suatu PE signature (Sumber Matt Pietrek, 2002)

PE header hanya terdiri dari suatu signatur File ID signature, yang memiliki nilai "PE\0\0" dimana masing-masing '\0' karakter adalah suatu karakter ASCII NULL. Informasi selanjutnya setelah PE signature adalah COFF header.

2.2.3 COFF Header

Berikut ini adalah Common Object File Format (COFF) header yang dinyatakan sebagai struktur data C :

```
typedef struct _IMAGE_FILE_HEADER {
    USHORT  Machine;
    USHORT  NumberOfSections;
    ULONG   TimeDateStamp;
    ULONG   PointerToSymbolTable;
    ULONG   NumberOfSymbols;
    USHORT  SizeOfOptionalHeader;
    USHORT  Characteristics;
}IMAGE_FILE_HEADER,
*PIMAGE_FILE_HEADER;
#define              IMAGE_SIZEOF_FILE_HEADER
20
```

Machine, field ini menunjukan bahwa file tersebut dikompilasi untuk mesin tertentu berdasarkan nilainya : (**14C**h) untuk Intel 80386 processor, (14Dh) untuk Intel 80486 processor,(14Eh) untuk Intel Pentium processor atau sesudahnya.
SizeOfOptionalHeader, field ini menunjukan berapa panjang "PE Optional Header" yang mengikuti COFF header.
Characteristics , field ini merupakan bit flag yang menunjukan karakteristik dari file.

2.2.4 PE Optional Header

"PE Optional Header" tidak selamanya "optional", karena dibutuhkan dalam file Executable files, tetapi tidak dalam object file. PE Optional Header berada setelah COFF header, yang kalau dinyatakan dalam struktur data C.

```
typedef struct _IMAGE_OPTIONAL_HEADER {
  //
  // Standard fields.
  //
  USHORT  Magic;
  UCHAR   MajorLinkerVersion;
  UCHAR   MinorLinkerVersion;
  ULONG   SizeOfCode;
  ULONG   SizeOfInitializedData;
  ULONG   SizeOfUninitializedData;
  ULONG   AddressOfEntryPoint;
  ULONG   BaseOfCode;
  ULONG   BaseOfData;
  //
  // NT additional fields.
  //
  ULONG   ImageBase;
  ULONG   SectionAlignment;
  ULONG   FileAlignment;
  USHORT  MajorOperatingSystemVersion;
  USHORT  MinorOperatingSystemVersion;
  USHORT  MajorImageVersion;
  USHORT  MinorImageVersion;
  USHORT  MajorSubsystemVersion;
  USHORT  MinorSubsystemVersion;
  ULONG   Reserved1;
  ULONG   SizeOfImage;
  ULONG   SizeOfHeaders;
  ULONG   CheckSum;
  USHORT  Subsystem;
  USHORT  DllCharacteristics;
  ULONG   SizeOfStackReserve;
  ULONG   SizeOfStackCommit;
  ULONG   SizeOfHeapReserve;
  ULONG   SizeOfHeapCommit;
  ULONG   LoaderFlags;
  ULONG   NumberOfRvaAndSizes;
  IMAGE_DATA_DIRECTORY
DataDirectory[IMAGE_NUMBEROF_DIRECTORY
_ENTRIES];
} IMAGE_OPTIONAL_HEADER,
*PIMAGE_OPTIONAL_HEADER;
```

2.2.5 PE File Sections

Sections mengandung isi dari file, termasuk kode, data dan resource, dan informasi executable lainnya. Masing-masing section memiliki suatu header dan satu body (the raw data).

Section header berada pada posisi setelah PE Optional header. Masing-masing header memiliki struktur yang berukuran 40 byte, berikut ini adalah section header yang dinyatakan sebagai struktur data C.

```
#define IMAGE_SIZEOF_SHORT_NAME 8
typedef struct _IMAGE_SECTION_HEADER {
  UCHAR
Name[IMAGE_SIZEOF_SHORT_NAME];
  union {
    ULONG   PhysicalAddress;
    ULONG   VirtualSize;
  } Misc;
  ULONG   VirtualAddress;
  ULONG   SizeOfRawData;
  ULONG   PointerToRawData;
  ULONG   PointerToRelocations;
  ULONG   PointerToLinenumbers;
  USHORT  NumberOfRelocations;
  USHORT  NumberOfLinenumbers;
  ULONG   Characteristics;
} IMAGE_SECTION_HEADER,
*PIMAGE_SECTION_HEADER;
```

Untuk mengambil masing-masing bagian section header anda dapat menggunakan .text untuk executable code section, .bss, .rdata, .data untuk data section, dan .rsrc untuk resource section.

3. Perancangan

Penelitian ini dilakukan dengan :

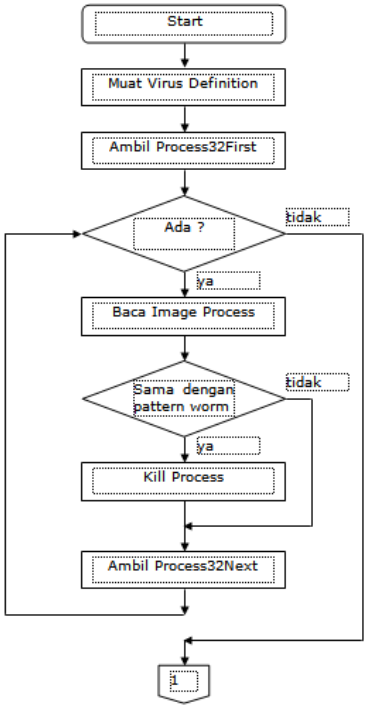
1. Pengamatan langsung terhadap struktur PE program worms yang berhasil penulis kumpulkan berdasarkan hasil deteksi dengan menggunakan program antivirus komersial sebagai Brontok.A, Klez, dan Netsky.
2. Pengamatan langsung terhadap struktur PE process worms yang merupakan hasil memori dump.
3. Melakukan studi pustaka dengan menggambil referensi dari beberapa situs yang relevan yang membahas tentang worms komputer, struktur format PE file, dan process program dimemori.
4. Mengembangkan algoritma untuk pengekstrak pola dari file maupun process worms dengan pendekatan *entry-point*.

5. Merancang algoritma removal tools, dan mengaplikasikannya menjadi prototipe perangkat lunak.
6. Melakukan pengujian terhadap prototipe perangkat lunak atas efektifitas hasil rancangan untuk suatu kesimpulan.

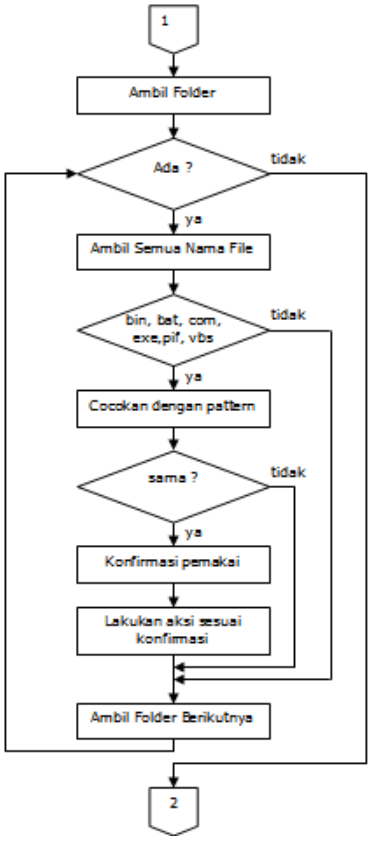
Adapun alasan peneliti menggunakan worms Brontok.A, Klez, Netsky menjadi worms pada penelitian ini karena worms ini merupakan worms yang sangat popular dan berhasil didalam penyebarannya.

Untuk mendukung proses penelitian, penulis menggunakan menggunakan beberapa tools pendukung seperti Hexa View, PE Explorer dan pengembangan aplikasi akan menggunakan bahasa pemrograman Delphi.

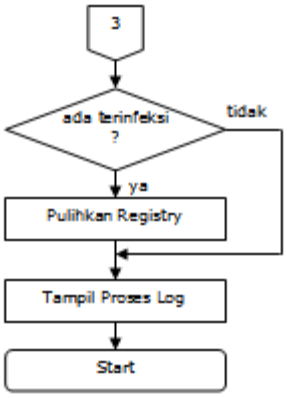
Adapun algoritma proses program aplikasi pendeteksi dan penghapus worms komputer ditunjukan secara flowchat pada Gambar 4, 5, dan 6



Gambar 4. Flowchart logika proses program penghapus Worms



Gambar 5. Flowchart logika proses program penghapus Worms (lanjutan)



Gambar 6. Flowchart logika proses program penghapus Worms (lanjutan)

3.1 PE Header

Pada *worm* yang kita teliti kita tidak dapat menggunakan struktur _IMAGE_DOS_HEADER untuk membaca e_lfanew yang merupakan pointer ke posisi PE header, untuk itu kita harus membaca awal executable file dan mencari pola "PE\0\0" untuk mendapatkan posisi PE header yang sebenarnya dengan algoritma sebagai berikut ini:

begin

```
    blockread(fhandle,Buf,512);      // membaca 512 byte pertama
    e_lfanew := Pos('PE'#0#0,Buf); // mencari pola PE\0\0
    if e_lfanew > 0 then
      begin
      Seek(fhandle, e_lfanew-1);      // pindah posisi ke e_lfanew
       blockread(fhandle,nt_header,sizeof(nt_header)); // baca ke PE Header
                                        AddrOfEP :=
nt_header.OptionalHeader.AddressOfEntryPoint;
                                    SectionAlign :=
nt_header.OptionalHeader.SectionAlignment;
                                    BaseOfCode :=
nt_header.OptionalHeader.BaseOfCode;
       blockread(fhandle,section_header,sizeof(section_header));
                                    CodeOffset :=
section_header.PointerToRawData ;
      result := true;
      end
    else
      result := false;
    close(fhandle);
 end;
```

3.2 Ekstraksi Pola dari *Worms*

Untuk pembuatan pola dengan pendekatan *entry point* tidak dapat dilakukan secara sembarang, tetapi perlu melakukan kalkulasi untuk mendapatkan posisi fisik dari *AddressOfEntryPoint* pada file executable dengan rumusan sebagai berikut:

fPos = AddrOfEp-BaseOfCode+CodeOffset

Setelah mendapatkan posisi fisik AddrOfEp, kita dapat membaca sejumlah byte awal executable, mengkonversikannya menjadi hexadesimal string dan disimpan sebagai pola pengenal bagi Worms tersebut dengan algoritma sebagai berikut:

```
    begin
      Seek(fhandle, fpos);
      blockread(fhandle,buf,plen);
      polaworms := datatohex(buf,plen);
      result := true;
      close(fhandle);
    end;
```

3.3 Pendeteksian *File Worms*

Pendeteksian worms pada media penyimpanan dapat dilakukan dengan membaca sejumlah byte yang sama pada saat pembuatan pola dari posisi fpos, sebagaimana ditunjukan pada algoritma berikut ini:

```
   begin
     result := false;
     fpos   := StrToInt('$'+Copy(pattern,31,8));
     plen   := StrToInt('$'+Copy(pattern,39,4));
     polaworms := Copy(pattern,43,64);
     seek(ftarget,fpos+relatif);
     fillchar(Buf,32,0);
     blockread(ftarget,Buf,plen,ret);
     result = (polaworms = (datatohex(Buf,plen));
   end;
```

3.4 Pendeteksian *Process Worms*

Untuk melakukan pemeriksaan pola worms terhadap process image dimemory, kita perlu mendapatkan ModBaseAddr kemudian ditambahkan dengan AddrOfEp dengan algoritma sebagai berikut:

```
   begin
     result := false;
     fpos := StrToInt('$'+Copy(pattern,31,8));
     plen := StrToInt('$'+Copy(pattern,39,4));
     polaworms:= Copy(pattern,43,64);
                                    retvalue :=
ReadProcessMemory(hProc,ptr(ModBaseAddr+fPos),@Buf,plen,ret);
     if retvalue then
       result := (polaworms = datatohex(Buf,plen));
   end;
```

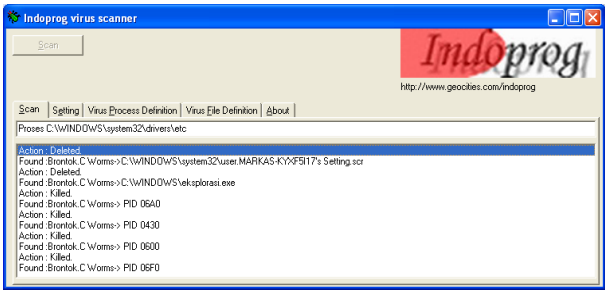
3.5 Pemulihan Registry System

Pemulihan registry system dilakukan dengan mengembalikan nilai registry ke nilai default Sistem Operasi dengan algoritma sebagai berikut:

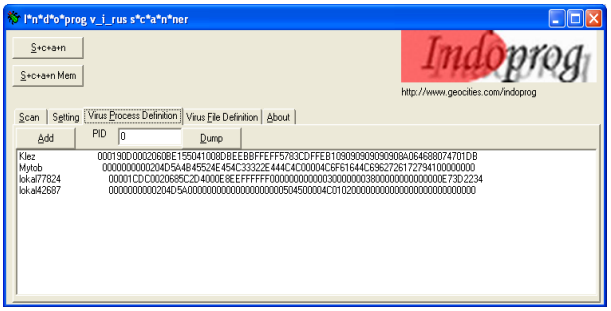
```
begin
{recover shell command}
Regwrite('HKCR','\exefile\shell\open\command','',
'"%1" %*','');
Regwrite('HKCR','\comfile\shell\open\command','',
'"%1" %*','');
Regwrite('HKCR','\piffile\shell\open\command','',
'"%1" %*','');
Regwrite('HKCR','\scrfile\shell\open\command','',
'"%1" %*','');
Regwrite('HKCR','\regfile\shell\open\command','',
'regedit.exe "%1"','');

{recover class}
Regwrite('HKLM','\SOFTWARE\Classes\exefile','','
Application','');

{recover run}
Regwrite('HKLM','\Software\Microsoft\Windows
NT\CurrentVersion\Winlogon','Shell',
'Explorer.exe','');
Regwrite('HKLM','\Software\Microsoft\Windows
NT\CurrentVersion\Winlogon','Userinit',
GetEnvironmentVariable('windir')
+'\system32\userinit.exe,','');
Regwrite('HKLM','\SYSTEM\CurrentControlSet\Con
trol\SafeBoot','AlternateShell', 'Cmd.exe','');
```

```
Regwrite('HKLM','\SYSTEM\ControlSet001\Control\
SafeBoot','AlternateShell', 'Cmd.exe','');
Regwrite('HKLM','\SYSTEM\ControlSet002\Control\
SafeBoot','AlternateShell', 'Cmd.exe','');
Regwrite('HKLM','\SYSTEM\ControlSet003\Control\
SafeBoot','AlternateShell', 'Cmd.exe','');
end
```

Adapun rancangan antar muka pemakai dapat dilihat pada gambar 7, 8, 9, dan 10.

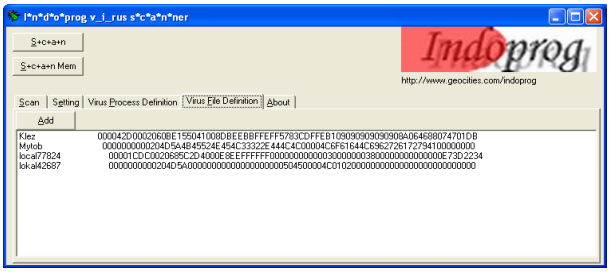


Gambar 7. Rancangan Layar Halaman Scan



Gambar 8. Rancangan layar untuk halaman Virus Process Definition

Gambar 9. Rancangan layar menambah pola Worms



Gambar 10. Daftar Virus Defininiton

4. Kesimpulan dan Saran

4.1 Kesimpulan

5 Sebagaimana file executable Windows biasanya, program Worms komputer memiliki struktur PE file, tetapi tidak tertutup kemungkinan terjadi reduksi pada DOS header.

6 Kunci utama pendeteksian worms komputer adalah ketersediaan signature unik yang digunakan sebagai pola pendeteksian.

7 Pengambilan beberapa byte awal executable worms pada posisi *entry point* dapat dijadikan sebagai signature untuk mendeteksi worms tersebut.

8 Aplikasi tidak perlu dijalankan secara safemode karena memiliki kemampuan deteksi process worms dimemori dan menghentikannya.

9 Aplikasi telah dilengkapi dengan kemampuan pemulihan registry ke nilai default.

10 Berdasarkan hasil evaluasi terhadap aplikasi removal tool yang dikembangkan oleh peneliti adalah efektif untuk mendeteksi dan menghapus program worms yang polanya telah digenerate dan dimasukan kedalam aplikasi.

4.2 Saran

Untuk pengembangan selanjutnya, metode yang sama dapat diterapkan pada realtime antiworms.

Daftar Rujukan

[1] Kaspersky Lab (2006), Viruslist.com – Network Worms, URL : http://www.viruslist.com/en/virusesdescribed?chapter=152540408,

[2] Szor Peter (2005), *The Art of Cumputer Virus Research and Defense*, Addison Wesley Professional.

[3] Matt Pietrek (2002), *Inside Windows : An In-Depth Look into the Win32 Portable Executable File Format*, URL : http://msdn.microsoft.com/msdnmag/issues/02/02/PE.

[4] Help and Support (2005), *Description of Microsoft Windows Registry*, URL : http://support.microsoft.com/default.aspx?scid=kb;EN-US;256986

[5] Yassin Ahmed(2004) , *The Code Project Win32 APIs for Process Retrieval*, URL : http://www.codeproject.com/threads/Tasks.asp

[6] Help and Support (2004), *How to Terminate an Application "Cleanly" in Win32*, URL : http://support.microsoft.com/?kbid=178893

[7] Bezroukov Nikolai(2005), *Fighting Network Worms*, URL : http://www.softpanorama.org/Malware/Malware_defense_history/fighting_network_worms.shtml