Aplikasi pengekstrak pola worms komputer.

Hendra*.

Intisari

Perkembangan worms lokal telah menjadi salah satu gangguan pemanfaatan teknologi komputer di tanah air, karena fasilitas pendeteksian dan penghapusan worms komputer lokal tidak tersedia dengan cepat pada antivirus komersial. Makalah ini menawarkan suatu model pengekstrak pola dari suatu file atau process worms komputer yang dapat kemudian dapat 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.

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. Brontok.Bro berhasil mencapai popularitas yang tinggi dikalangan teknologi informasi, hal ini dibuktikan dengan hasil pencarian kata Brontok dengan mesin pencari google yang mencapai 194.000 halaman.[1]

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 secare 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 aplikasi yang dicurigai sebagai malware.

2. Permasalahan

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

3. Tujuan

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

---

* Dosen Tetap STMIK IBBI.

[1] Dilakukan dengan mencari kata Brontok.Bro pada mesin pencari Google (http://www.google.com) pada tanggal 13 Juni 2006 pukul 13:36 WIB.

4. Pembatasan masalah

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*.

6. Asumsi

Pada penelitian ini menggunakan asumsi bahwa pemakai dari aplikasi removal dapat mengenali file worms dan memasukan kedalam removal tools untuk diekstraksi polanya.

7. Tinjauan teoritis

7.1. Worms komputer

"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."[1]

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. Virus umumnya tidak mempengaruhi unjuk kerja network, karena aktivitas penyebaran umumnya dilakukan pada komputer target itu sendirinya, dan melalui pertukaran file executable.

Selain kemampuan replikasi, suatu worms dapat dirancang untuk melakukan sejumlah hal seperti menghapus file tertentu pada suatu sistem, melakukan pengiriman dokumen lewat e-email, dan suatu payload.

Payload umumnya pada suatu worms adalah menginstalasi suatu backdoor (pembuka celah dengan melakukan listen pada port tertentu) pada komputer yang terinfeksi, sebagaimana yang dilakukan oleh Sobig dan Mydoom melakukan spam (pengiriman email massal) dengan mengirim email sampah ke alamat situs tertentu dengan tujuan melakukan ancaman dan pemerasan, serta melakukan serangan DoS.
Secara teoritis worms dapat bermanfaat sebagaimana yang sering menjadi pertanyaan pada bidang sains komputer dan kecerdasan buatan. Worms Nachi misalnya, mencoba mendownload dan menginstalasi patch dari situs web Microsoft untuk memperbaiki berbagai celah di komputer yang berhasil dimasukinya.

7.3. Pola Pendeteksian

Pendeteksian adakah suatu proses dari pencarian atau penemuan sumber sebenarnya atau masalah dari skenario yang mungkin dan pola adalah didefinisikan sebagais 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.

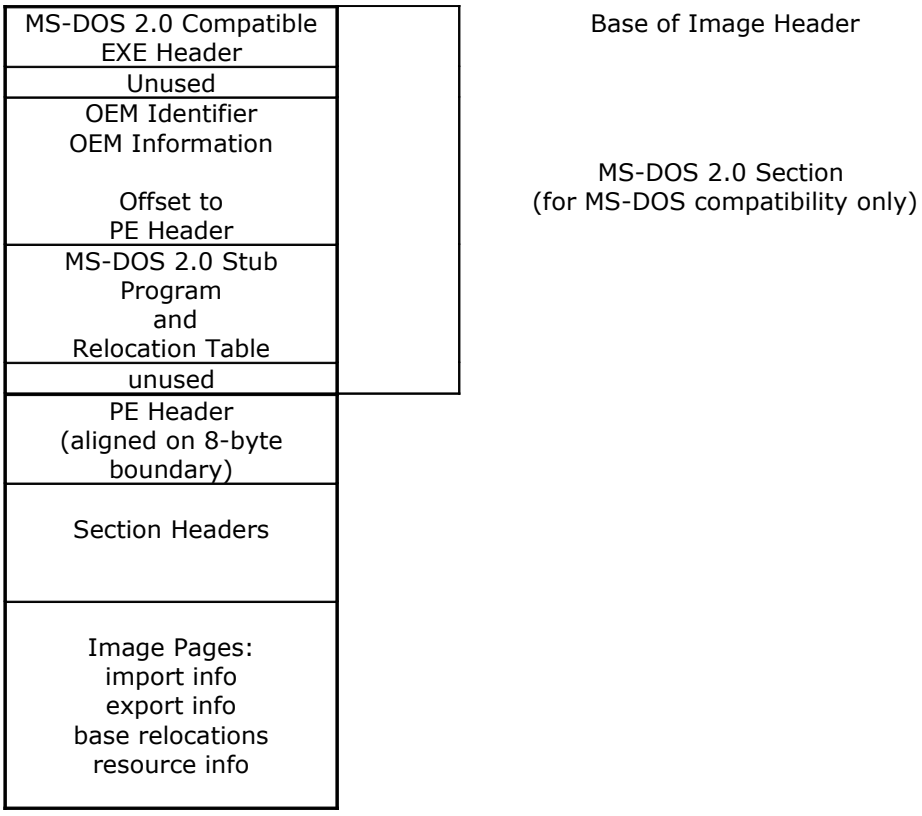
Pendeteksian terhadap Worms membutuhkan pola yang merupakan ekstraksi dari data Worms itu sendirinya, sehingga dapat digunakan sebagai pola pedeteksian worms tersebut secara pencocokan string.

7.4. 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 berikut:



Gambar 1. Struktur file PE
(sumber Matt Pietrek, 2002)

7.3.1. 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)

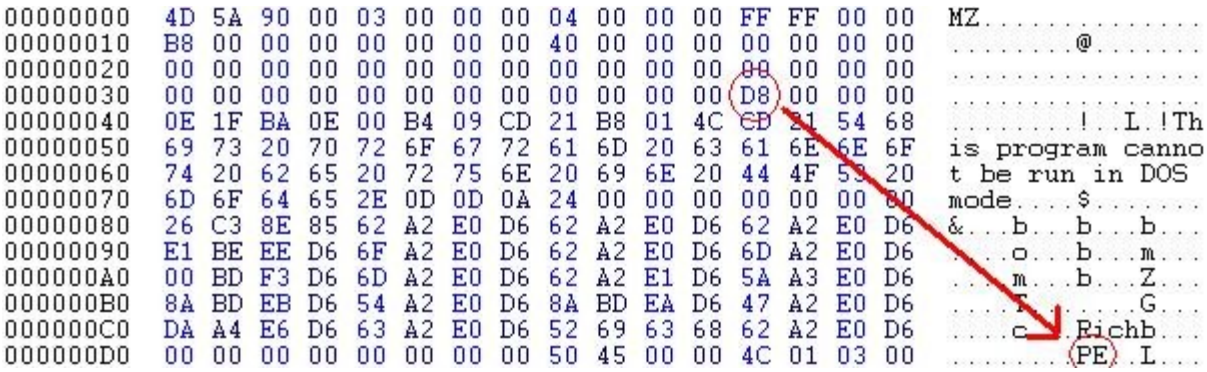
Berikut ini adalah DOS header yang dinyatakan dalam struktur data C :

WINNT.H

```
typedef struct _IMAGE_DOS_HEADER {  // DOS .EXE header
    USHORT e_magic;         // Magic number
    USHORT e_cblp;          // Bytes on last page of file
    USHORT e_cp;            // Pages in file
    USHORT e_crlc;          // Relocations
    USHORT e_cparhdr;       // Size of header in paragraphs
    USHORT e_minalloc;      // Minimum extra paragraphs needed
    USHORT e_maxalloc;      // Maximum extra paragraphs needed
    USHORT e_ss;            // Initial (relative) SS value
    USHORT e_sp;            // Initial SP value
    USHORT e_csum;          // Checksum
    USHORT e_ip;            // Initial IP value
    USHORT e_cs;            // Initial (relative) CS value
    USHORT e_lfarlc;        // File address of relocation table
    USHORT e_ovno;          // Overlay number
    USHORT e_res[4];        // Reserved words
    USHORT e_oemid;         // OEM identifier (for e_oeminfo)
    USHORT e_oeminfo;       // OEM information; e_oemid specific
    USHORT e_res2[10];      // Reserved words
    LONG   e_lfanew;        // File address of new exe header
  } IMAGE_DOS_HEADER, *PIMAGE_DOS_HEADER;
```

7.3.2. 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.

7.3.3. COFF Header

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

WINNT.H
```
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.

7.3.4. 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 adalah sebagai berikut :

WINNT.H
```
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;
```

7.3.5. 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 :

```
WINNT.H
#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.

7.4. Process Program

Process merupakan suatu instance yang sedang berjalan dari suatu program, termasuk semua variable dan status lainnya. Pada suatu system operasi multitasking, system operasi melakukan switch antara process-process untuk memungkinkan banyak process dijalankan secara simultan, walaupun sebenarnya hanya ada satu process yang dapat dijalankan pada suatu waktu thread CPU.

Sistem operasi mengatur masing-masing process secara terpisah dan mengalokasikan resource-resource yang mereka butuhkan sehingga tidak mengganggu satu sama lainnya

yang dapat menyebabkan kegagalan system (seperti deadlock). Sistem operasi juga menyediakan mekanisme untuk komunikasi antar process yang memungkinkan terjadinya interaksi antar process secara aman.

Pada umumnya, suatu process system komputer terdiri dari resource sebagai berikut:

- *Image*, yang merupakan kode executable komputer yang merupakan instance dari program yang dieksekusi.
- Memory, yang merupakan tempat bagi image dan data process tersebut.
- Pengenal dari system operasi terhadap resource yang dialokasi kepada process tersebut yang dikenal dengan istilah handle pada system operasi Windows.
- Atribut security, seperti process owner dan sejumlah setting permission terhadap process tersebut.
- Processor state, yang mana merupakan sisi dari register, pengalamatan memori fisik, dll. State ini secara khusus disimpan pada register komputer ketika process ini dijalankan, dan sebaliknya dimemori.

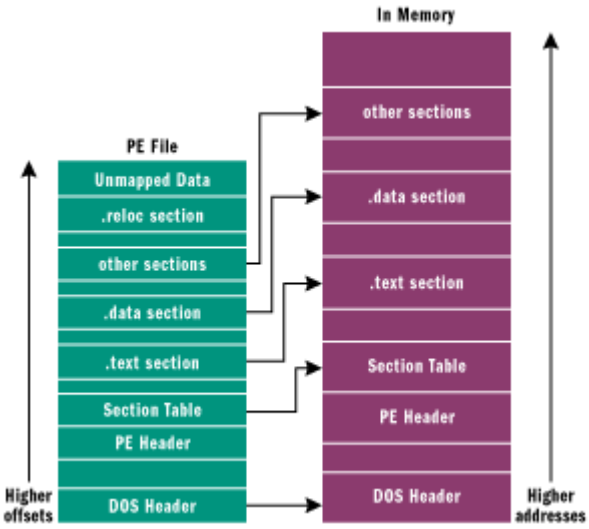
Pada system operasi Windows, masing-masing process memiliki Id pengenal yang dikenal sebagai istilah PID (Process Id), untuk melihat menggambil process yang berjalan kita dapat menggunakan fungsi WINAPI sebagai berikut :

```
function TFScanEngine.CheckAllProcess : Boolean;
var
  aSnapshotHandle: THandle;
  aProcessEntry32: TProcessEntry32;
  bContinue: BOOL;
  Ret : Boolean;
begin
  Result := False;
  aSnapshotHandle := CreateToolhelp32Snapshot(TH32CS_SNAPPROCESS, 0);
  aProcessEntry32.dwSize := SizeOf(aProcessEntry32);
  bContinue := Process32First(aSnapshotHandle, aProcessEntry32);
  while Integer(bContinue) <> 0 do
       begin
       Ret := CheckProcess(aProcessEntry32.th32ProcessID);
       Result := Result Or Ret;
       bContinue := Process32Next(aSnapshotHandle, aProcessEntry32);
       end;
  CloseHandle(aSnapshotHandle);
end;
```

Berikut ini adalah langkah-langkah suatu PE file dimuat ke memori :

1. Ketika suatu file PE dijalankan, suatu PE loader akan mencari lokasi PE header dari struktur DOS MZ header. Jika ditemukan, maka proses dilanjutkan ke struktur PE header.
2. PE loader memeriksa apakah PE header adalah valid, jika ya, akan dilanjutkan sampai ke akhir PE header.
3. Setelah PE header adalah section table. PE loader membaca informasi tentang masing-masing section dan melakukan pemetaan section-section ke memory dengan menggunakan file mapping. Juga memberikan atribut kemasing-masing section sebagaimana yang ditentukan dalam section table.
4. Setelah file PE dipetakan dalam memori, Pe loader akan berkonsentarasi pada import table untuk memuat dll yang dibutuhkan.

Sesuatu aspek dari PE file adalah struktur data pada disk adalah sama dengan struktur data yang digunakan dimemori.



Gambar 4. Mapping PE file terhadap image process.
(sumber Matt Pietrek, 2002)

Ketika suatu PE file dimuat ke memori melalui Windows loader yang dikenal sebagai istilah module. Alamat awal dimana file dimapping disebut sebagai HMODULE, anda dapat menggunakan fungsi API untuk mendapatkan alamat tersebut.

7.5. Relative Virtual Addresses

RVA singkatan dari relative virtual address yang secara relative menunjuk ke suatu lokasi pengalamatan virtual, sebagai contoh, jika suatu PE file ditempatkan pada alamat 400000h (ModBaseAddr) pada suatu lokasi pengalamatan virtual dan program tersebut mulai dieksekusi pada alamat 401000h, kita dapat menggatakan bahwa program mulai dijalankan pada RVA RVA 1000h. suatu RVA adalah relative terhadap VA dari module.

Mengapa suatu format PE file menggunakan RVA? Hal tersebut untuk membantu PE loader, sebab suatu module dapat ditempatkan dimana saja di lokasi pengalamatan virtual, dan adalah tidak mungkin bagi PE loader untuk melakukan relokasi terhadap semua pengalamatan dalam PE file. Dengan RVA PE loader tidak perlu melakukan perubahan apapun terhadap image PE, melainkan hanya perlu melakukan relokasi keseluruhan module ke VA yang baru.

7.6. Registry

Registry merupakan suatu database hirarki yang yang digunakan pada Microsoft Windows 9x, Windows CE, Windows NT, and Windows 2000/XP untuk meyimpan informasi yang penting untuk konfigurasi system untuk satu atau semua pemakai, aplikasi dan peralatan perangkat keras.

Registry pada Windows memiliki root sebagai berikut :

HKEY_CURRENT_USER
Merupakan root informasi konfigurasi user yang sedang logon ke system yang merupakan user profile untuk setting desktop, folder, dll. Key ini biasanya disingkat sebagai "HKCU."

HKEY_USERS
Menyimpan semua user profile pemakai yang terdaftar pada system komputer. HKEY_CURRENT_USER merupakan subkey dari HKEY_USERS. HKEY_USERS yang biasanya disingkat sebagai "HKU."

HKEY_LOCAL_MACHINE
Mengandung konfigurasi yang berlaku untuk semua pemakai, key ini biasanya disingkat sebagai "HKLM."

HKEY_CLASSES_ROOT
Merupakan subkey dari HKEY_LOCAL_MACHINE\Software. Informasi yang terdapat disini untuk menentukan program untuk membuka file yang dibuka pada Windows Explorer, key ini biasanya disingkat sebagai "HKCR."
Masing-masing key tersebut diatas dibagi menjadi subkey-subkey, yang mana terbagi lagi menjadi subkey-subkey lanjutaan, dan suatu key dapat memiliki nilai berikut ini :
Nilai String
Nilai Binary (0 dan 1)
Nilai DWORD (angka antara 0 dan 4,294,967,295 [$2^{32} - 1$])
Nilai Multi-String
Nilai String yang dapat diperluas

Karena registry merupakan pusat penyimpanan setting system Windows secara keseluruhan, sehingga banyak dieksploitasi oleh Worms untuk aktifitas pengaktifan dirinya setiap kali user melakukan booting system, maupun menyembunyikan dirinya dari system.

Berikut ini adalah Registry Key yang sering dieksploitasi oleh Worms :

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows\CurrentVersion\Run
```

Dan

```
HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run
```

Merupakan key yang berisi daftar executable yang akan dijalankan oleh system operasi setiap startup.

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\Winlogon\Shell
```

Merupakan key yang berisi program shell untuk proses logon ke system operasi.

8. Metodologi penelitian

Penelitian ini dilakukan dengan :

1. Pemgamatan langsung terhadap program worms yang berhasil penulis kumpulkan dan jika dideteksi dengan menggunakan program antivirus komersial sebagai Brontok.A untuk mencari suatu pola unik yang akan digunakan sebagai signature pendeteksian worms tersebut.
2. Melakukan studi pustaka dengan menggambil referensi dari beberapa situs yang relevan yang membahas tentang worms komputer, struktur format PE file, dan process program dimemori.
3. Melakukan pengujian terhadap prototipe perangkat lunak atas efektifitas hasil rancangan untuk suatu kesimpulan.

Adapun alasan peneliti menggunakan worms Brontok.A menjadi worms pada penelitian ini karena worms ini merupakan worms lokal 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.

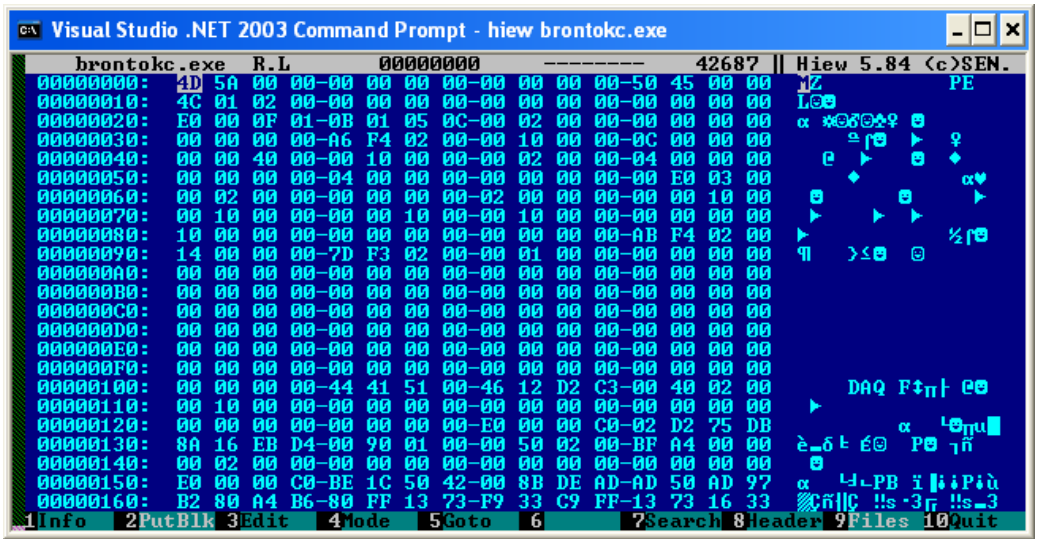
8. Analisa dan pengembangan

Proses analisa dan pengembangan yang dilakukan penulis dengan melakukan pengamatan terhadap masing-masing PE header worms dan kemudian mengembangkan algoritma untuk mengambil data yang diperlukan yaitu AddrOfEP, SectionAlign, BaseOfCode, CodeOffset.

Dari data AddrOfEP, SectionAlign, BaseOfCode, CodeOffset akan dilakukan kalkulasi untuk mendapatkan lokasi fisik AddrOfEP yang sesungguhnya, setelah mendapatkan lokasi fisik dari AddrOfEP, maka dapat dikembangkan algoritma yang membaca sejumlah byte awal executable worm, untuk selanjutnya dikonversi menjadi rangkaian bilangan hexadecimal dan kemudian dikombinasikan dengan nilai AddrOfEP sehingga menjadi suatu pola unik yang akan menjadi signature untuk mendeteksi worms tersebut.

8.1. PE Header pada beberapa Worms

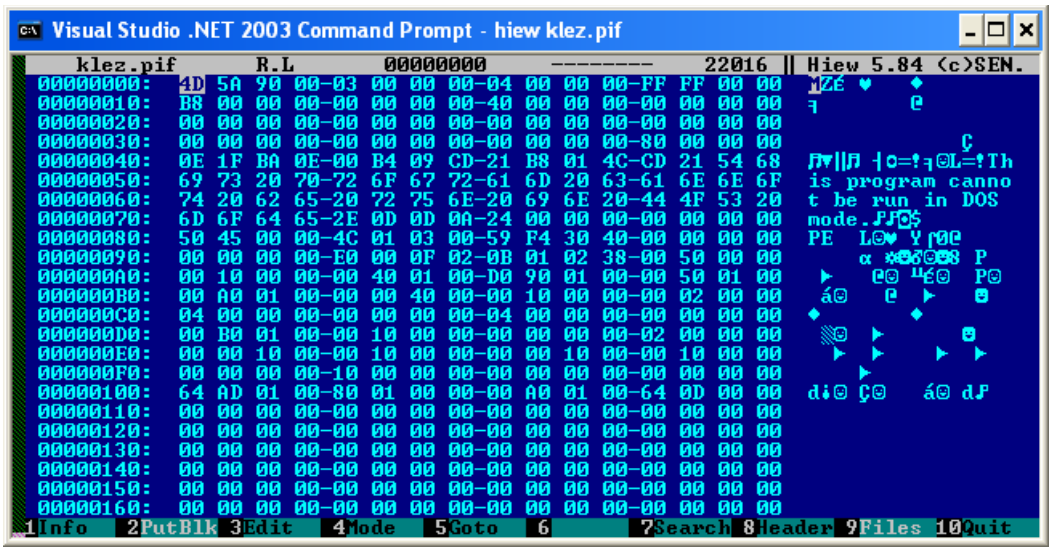
Berdasarkan pengamatan yang dilakukan oleh penulis, tidak semua worms memiliki DOS16-header yang normal, seperti Worms Brontok.A yang memiliki DOS16-Header sebagai berikut:



Gambar 5. PE Header poda worms Brontok.A.
(Sumber: Ekplorasi oleh peneliti)

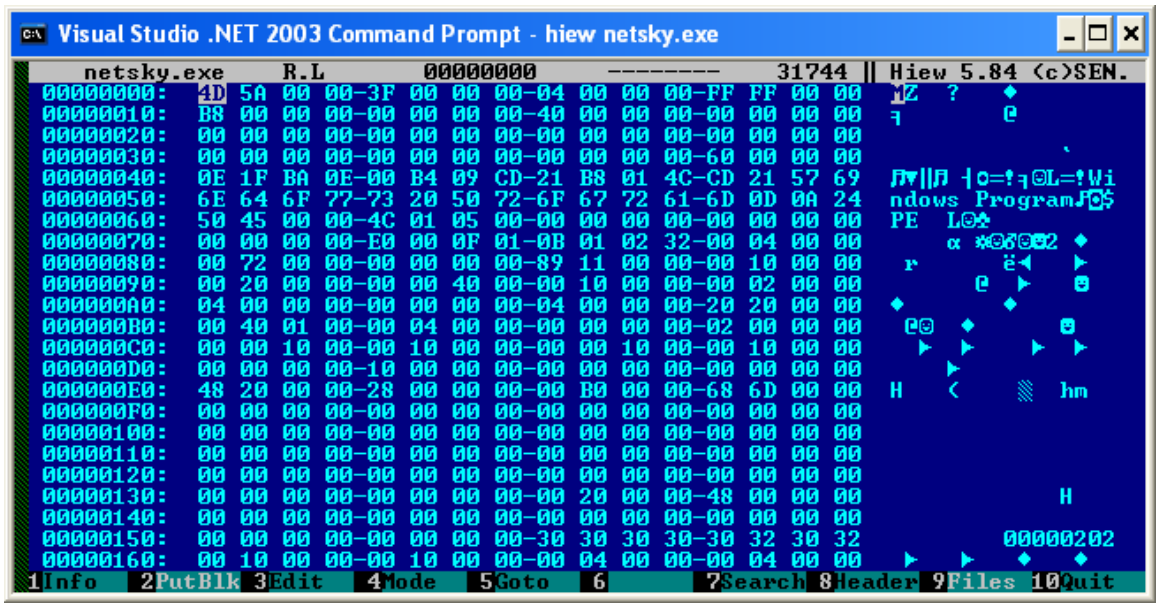
dimana dengan mereduksi bagian DOS-16 Header.

Sedangkan pada Worms Klez memiliki PE header yang normal



Gambar 6. PE Header poda worms Klez
(Sumber: Ekplorasi oleh peneliti)

dan pada Worms Netsky memiliki DOS16-header yang normal



Gambar 7. PE Header poda worms Blackmal
(Sumber: Ekplorasi oleh peneliti)

serta pada Worms Blackmal memiliki DOS16-header yang normal.

Sehingga 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 :

```
function getPEinfo(fname:string; var AddrOfEP, SectionAlign, BaseOfCode, CodeOffset : DWORD):boolean;
var
```

```
  fhandle : file;
  Buf : Array[1..512]of Char;
  e_lfanew : Word;
  nt_header : IMAGE_NT_HEADERS;
  section_header :IMAGE_SECTION_HEADER;
begin
  assignfile(fhandle,fname);
  FileMode := fmOpenRead;
  {$I-}
  reset(fhandle,1);
  {$I+}
  if IoResult <> 0 then
     begin
     result := false;
     end
  else
     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;
end;
```

8.2. Menggambil pola dari Worms

Untuk pembuatan pola yang akan digunakan sebagai signature untuk pendeteksian keberadaan worms, dilakukan secara sembarang, tetapi perlu melakukan kalkulasi untuk mendapatkan posisi fisik dari `AddressOfEntryPoint` pada file executable kita membutuhkan field AddrOfEp, BaseOfCode, CodeOffset dengan rumusan sebagai berikut :

fPos = AddrOfEp-BaseOfCode+CodeOffset

Setelah mendapatkan posisi fisik AddrOfEp, kita dapat membaca sejumlah byte awal executable untuk disimpan sebagai pola pengenal bagi Worms tersebut.

```
function getPatternfromFile(fname:string; fpos:longint; plen:word;var
pattern:string):boolean;
var
  fhandle : file;
  buf : TBuf;
begin
  assignfile(fhandle,fname);
  FileMode := fmOpenRead;
  {$I-}
```

```
  reset(fhandle,1);
  {$I+}
  if IoResult <> 0 then
     begin
     result := false;
     end
  else
     begin
     Seek(fhandle, fpos);
     blockread(fhandle,buf,plen);
     pattern := datatohex(buf,plen);
     result := true;
     close(fhandle);
     end;
end;
```

8.3. Pendeteksian worms

Pendeteksian worms pada media penyimpanan dapat dilakukan dengan membaca dari lokasi sejumlah byte yang sama pada signature yang telah disiapkan sebelumnya.

```
function TFileUnit.matchpattern(pattern:string;relatif:longint):boolean;
var
  Buf : TBuf;
  fpos  : longint;
  plen  : word;
  ret   : longint;
  pdata1,
  pdata2 : string;

begin
  result := false;

  fpos   := StrToInt('$'+Copy(pattern,31,8));
  plen   := StrToInt('$'+Copy(pattern,39,4));
  pdata1 := Copy(pattern,43,64);

  seek(ftarget,fpos+relatif);
  fillchar(Buf,32,0);
  blockread(ftarget,Buf,plen,ret);
  pdata2 := datatohex(Buf,plen);
  If pdata1 = pdata2 then
     result := true;
end;
```

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

```
function TMemUnit.matchPattern(pattern:string; ModBaseAddr:DWord):boolean;
var
  Buf : TBuf;
  fpos  : longint;
  plen  : word;
  pdata : string;
  retvalue : boolean;
  ret :cardinal;
```

```
begin
  result := false;


  fpos := StrToInt('$'+Copy(pattern,31,8));
  plen := StrToInt('$'+Copy(pattern,39,4));
  pdata:= Copy(pattern,43,64);

  retvalue := ReadProcessMemory(hProc,ptr(ModBaseAddr+fPos),@Buf,plen,ret);

  if retvalue then
     result := (pdata = datatohex(Buf,plen));
end;
```

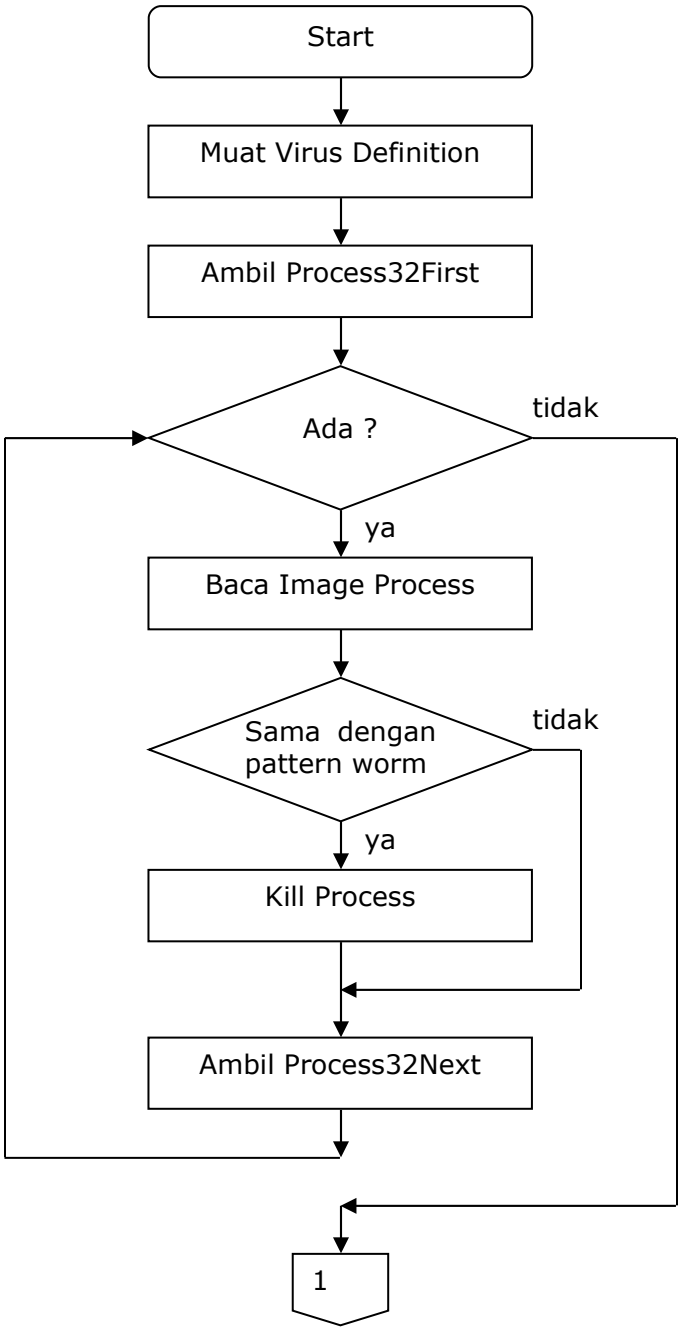
8.4. Pemulihan Registry Sistem

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

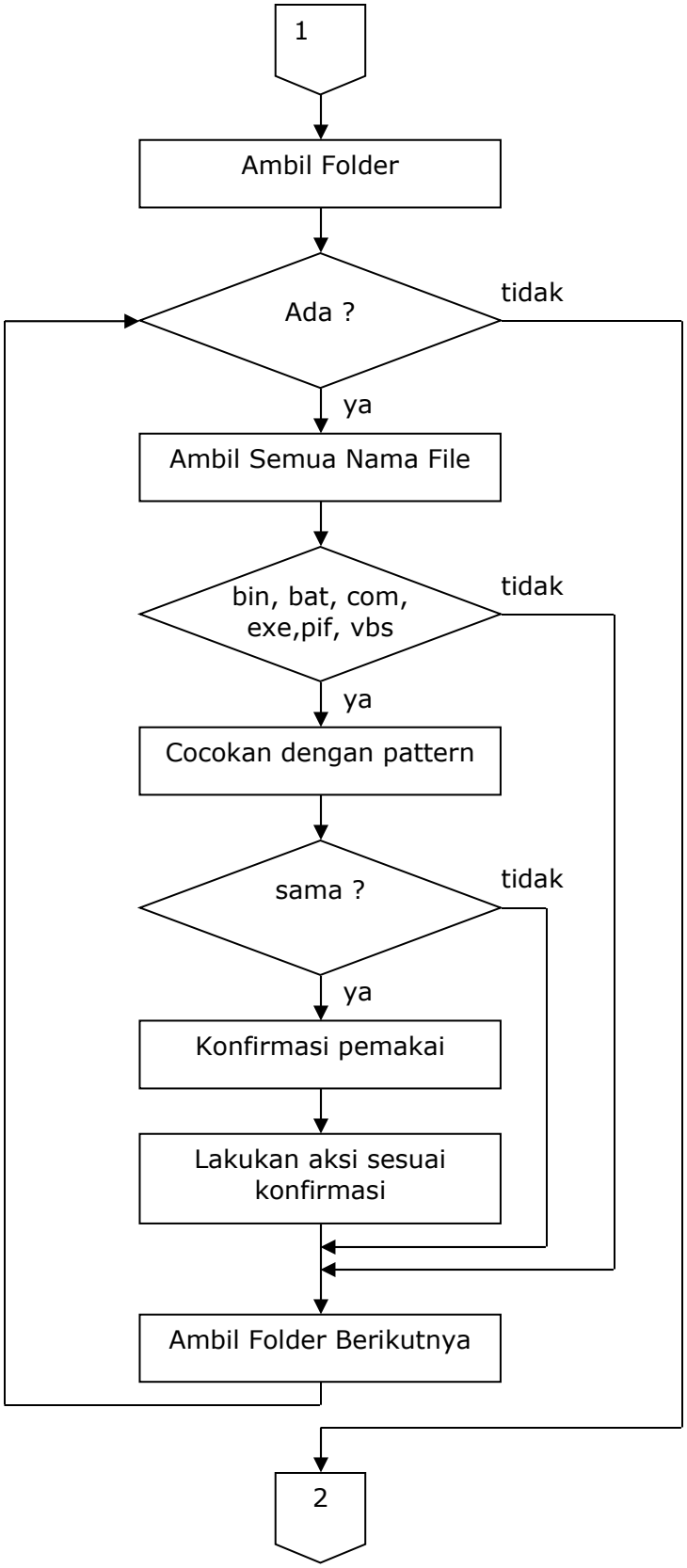
```
procedure TFScanEngine.RecoverRegistry;
begin
{recover shell command}
Regwrite('HKCR','\exefile\shell\open\command','', '"%1" %*','');
Regwrite('HKCR','\comfile\shell\open\command','', '"%1" %*','');
Regwrite('HKCR','\piffile\shell\open\command','', '"%1" %*','');
Regwrite('HKCR','\scrfile\shell\open\command','', '"%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\Control\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','');
```

9. Perancangan Removal Tools

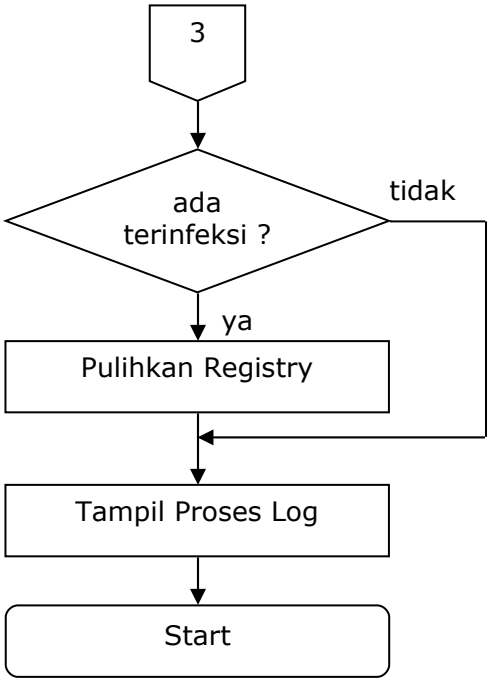
Adapun logika proses program aplikasi pendeteksi dan penghapus worms komputer digambarkan sebagai flowchart berikut ini :



Gambar 5. Flowchart logika proses program penghapus Worms



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



Gambar 7. Flowchart logika proses program penghapus Worms (lanjutan)

Gambaran aplikasi penghapus worms adalah sebagai berikut, mula-mula program akan memuat suatu daftar definisi worms yang berisi pola-pola worms untuk pendeteksian keberadaan process worms di system komputer maupun pada file. Kemudian program akan mengambil semua PID dari semua yang sedang aktif di system komputer, dan selanjutnya mengambil image process berdasarkan PID, dan image ini akan diperiksa dengan mencocokan dengan masing-masing pola worms yang telah dipersiapkan sebelumnya, jika ternyata image tersebut sama dengan salah satu pola, maka program akan menghentikan process berdasarkan PID, dan memberikan pesan kepada pemakai.

Tahapan selanjutnya adalah melakukan pencarian file-file yang berada pada system komputer, dengan melakukan proses pengambilan nama file. Berdasarkan nama file tersebut, program akan mengambil data dari file untuk dicocokan dengan masing-masing pola worms yang telah dipersiapkan, jika ternyata image tersebut sama dengan salah satu pola, maka program akan meminta konfirmasi pemakai akan tindakan yang perlu dilakukan dengan pilihan Delete (hapus) , Quarantine (karantina), atau Ignore (abaikan).

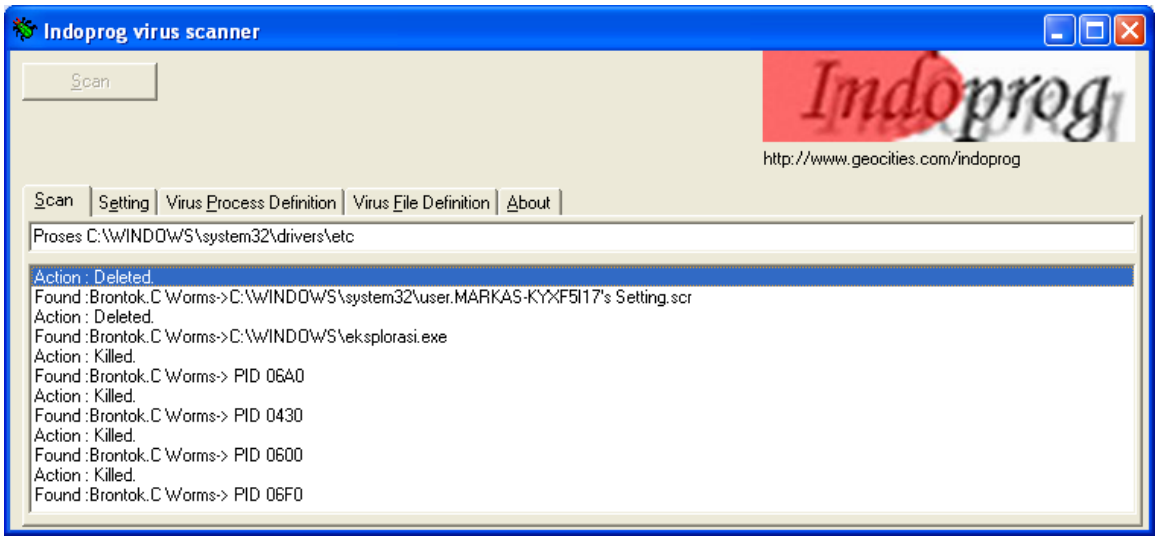
Selanjutnya program akan membersihkan registry yang diekploitasi oleh worms, dengan menghapus maupun mengembalikan nilai defaultnya.

9.1. Rancangan Antar Muka Pemakai

Rancangan Antar Muka Pemakai akan menggunakan SDI (Single Data Interface), yang dilengkapi dengan suatu TPageControl, dan tombol scan (TButton).

9.1.1 Halaman Scan

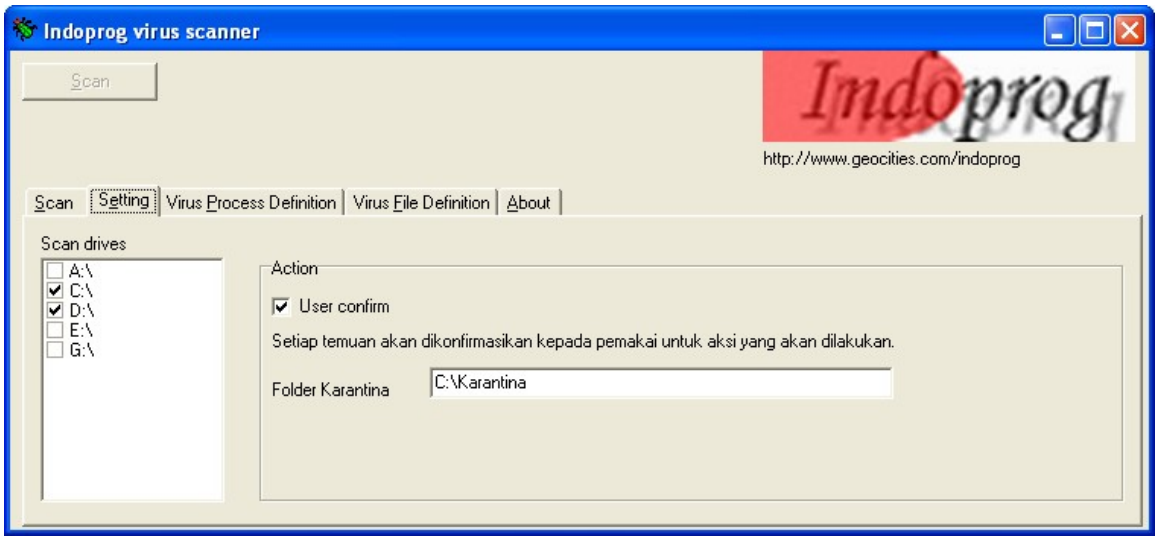
Halaman ini akan menampilkan aktifitas yang dilakukan oleh program selama scaning, yang menampilkan folder yang sedang diproses (TEdit), dan aksi yang dilakukan dalam kaitannya dengan temuan-temuan proses scanning (TListBox).



Gambar 8. Rancangan layar untuk halaman Scan

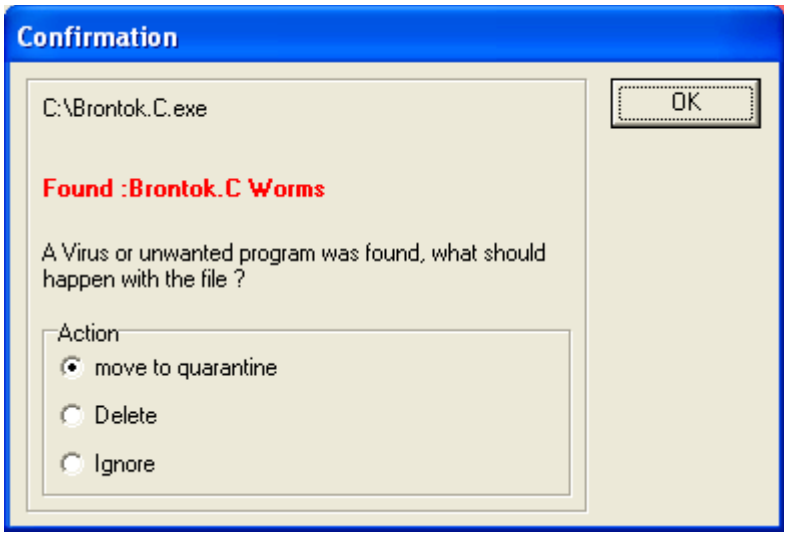
9.1.2 Halaman Setting

Halaman ini berisi opsi setting yang dapat dilakukan oleh pemakai, yang terdiri dari sebuah daftar pilihan drive yang akan discan (TCheckList), dan sebuah opsi pilihan aksi yang akan dilakukan terhadap setiap temuan (TCheckBox), serta Folder Karantina (TEdit)



Gambar 9. Rancangan layar untuk halaman Setting

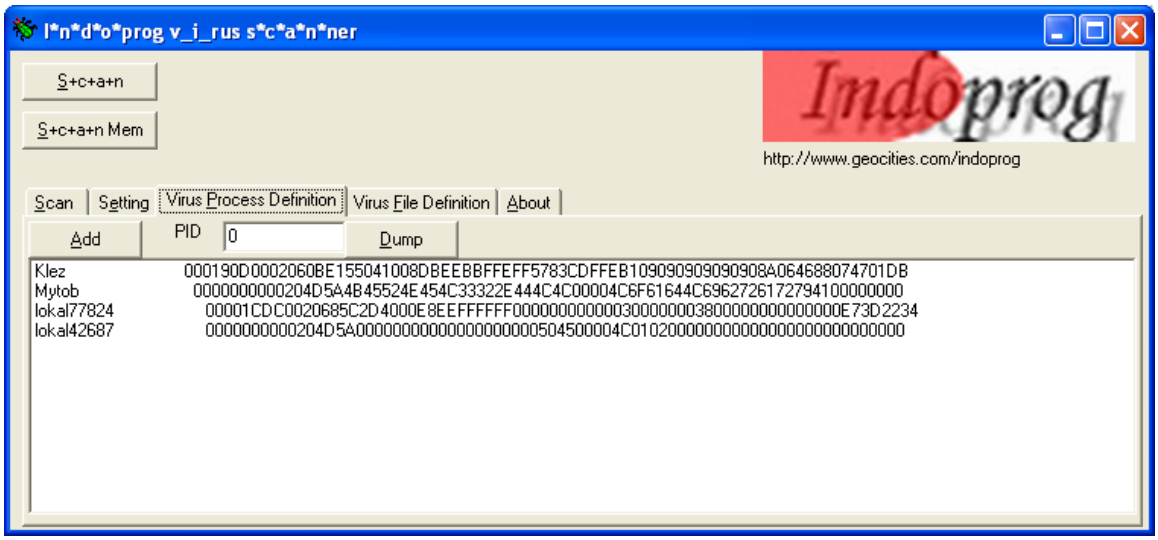
Jika pilihan konfirmasi User confirm, maka setiap temuan akan menampilkan dialog pilihan aksi yang akan dilakukan :



Gambar 10. Rancangan layar untuk halaman Confirmation

9.1.3 Halaman Virus Process Definition

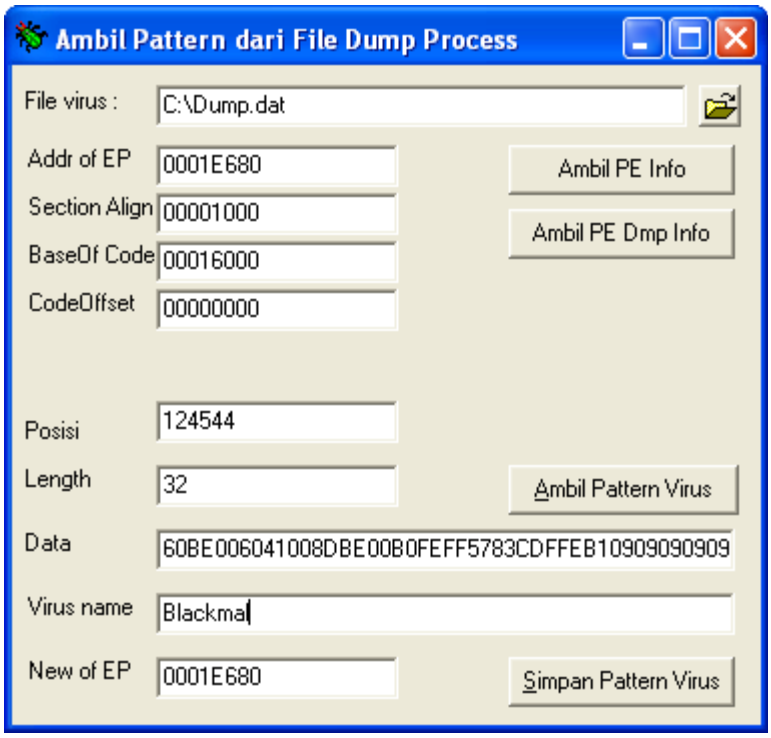
Halaman ini menampilkan daftar Worms yang dapat dikenali oleh program aplikasi dan beserta dengan polanya (TListBox), dan sebuah button Add (TButton) yang memberikan kesempatan kepada pemakai untuk menambahkan pola ke daftar Worms.



Gambar 11. Rancangan layar untuk halaman Virus Process Definition

Jika pemakai melakukan klik pada Button Add, maka akan ditampilkan suatu jendela dialog pengisian oleh pemakai :

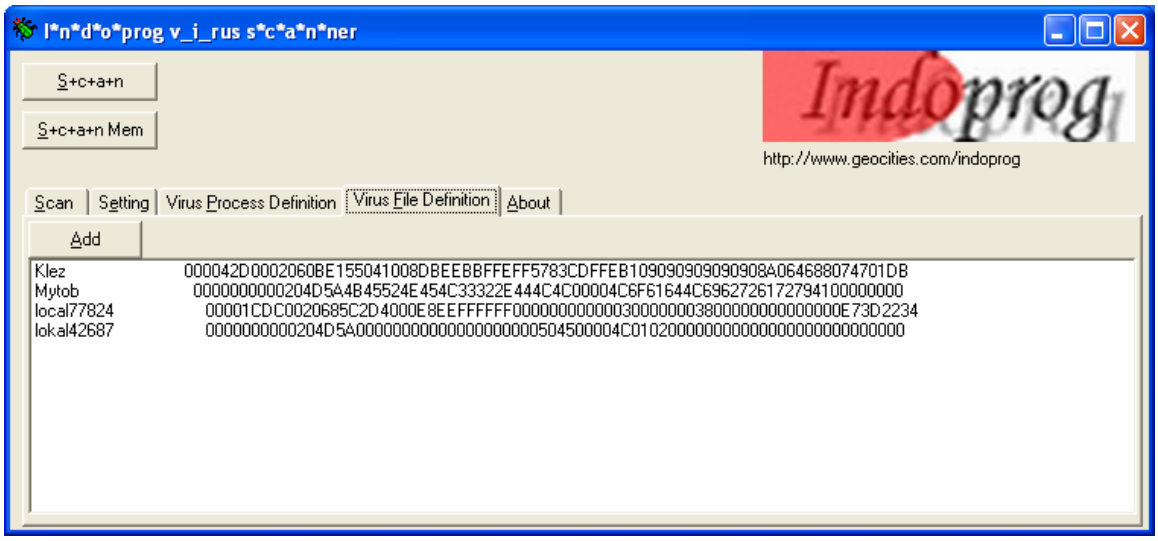
Dialog ini terdiri isian dialog File virus(TEdit), Addr of EP (TEdit), Section Align (TEdit), BaseOf Code (TEdit), CodeOffset(TEdit), Posisi (TEdit), Length (TEdit), Data (TEdit), dan Virus name (TEdit), Button Ambil Pola Virus (TButton) yang berfungsi menggambil pola worms dari File virus, dan Simpan Pola Virus (TButton) yang berfungsi mengaktifkan fungsi penyimpanan pola baru ke file MVirus.dat



Gambar 12. Rancangan layar menambah pola Worms

9.1.4 Halaman Virus File Definition

Menyerupai halaman Virus Process Definition, halaman ini berisi pola untuk virus file.



Gambar 13. Rancangan layar untuk halaman Virus File Definition

9.1.5 Halaman About

Halaman ini berisi informasi singkat tentang program aplikasi, dan versi serta platform pengoperasian program.

Gambar 14. Rancangan layar untuk halaman About

9.2. Rancangan Basis Data

Untuk penyimpanan data pola-pola worms, maka dibutuhkan media penyimpanan permanen dalam bentuk file, adapun struktur penyimpanan untuk file pola tersebut adalah sebagai berikut :

Field	Jenis	Ukuran	Keterangan
WormsName	Char	30	Menyimpan nama worms
Offset	Char	8	Menyimpan posisi offset dimana posisi data image maupun file untuk pemeriksaan pencocokan. (disimpan dalam string hexadecimal)
Ukuran	Char	4	Menyimpan ukuran pola (disimpan dalam string hexadecimal)
Pola	Char	64	Pola (disimpan dalam string hexadecimal)

Tabel 1. Struktur File Pola Worms

Karena struktur penyimpanan cukup sederhana, maka penyimpanan akan dilakukan pada suatu flat file, sehingga proses pembacaan maupun penulisan dapat dilakukan secara ringan (lightweight), dan tidak membutuhkan komponen-komponen engine database[2].

Contoh :

Brontok.A Worms
0000000000204D5A0000000000000000000000504500004C010200000000000000000000000000000
0

Dimana :

WormsName = Brontok.A Worms

[2] Tujuannya supaya program dapat didistribusikan dengan hanya menduplikasi file executable, sehingga tidak diperlukan proses instalasi.

Offset = 0
Ukuran = 32
Pola = 4D5A00000000000000000000504500004C010200000000000000000000000000

Pola Virus akan disimpan pada dua file yaitu MVirus.dat (pola untuk process image) dan FVirus.dat (pola untuk file image).

11.Implementasi

Dalam implementasi berikut ini, pemakai memakai contoh Virus Brontok.A yang memiliki ukuran executable 81920 byte.

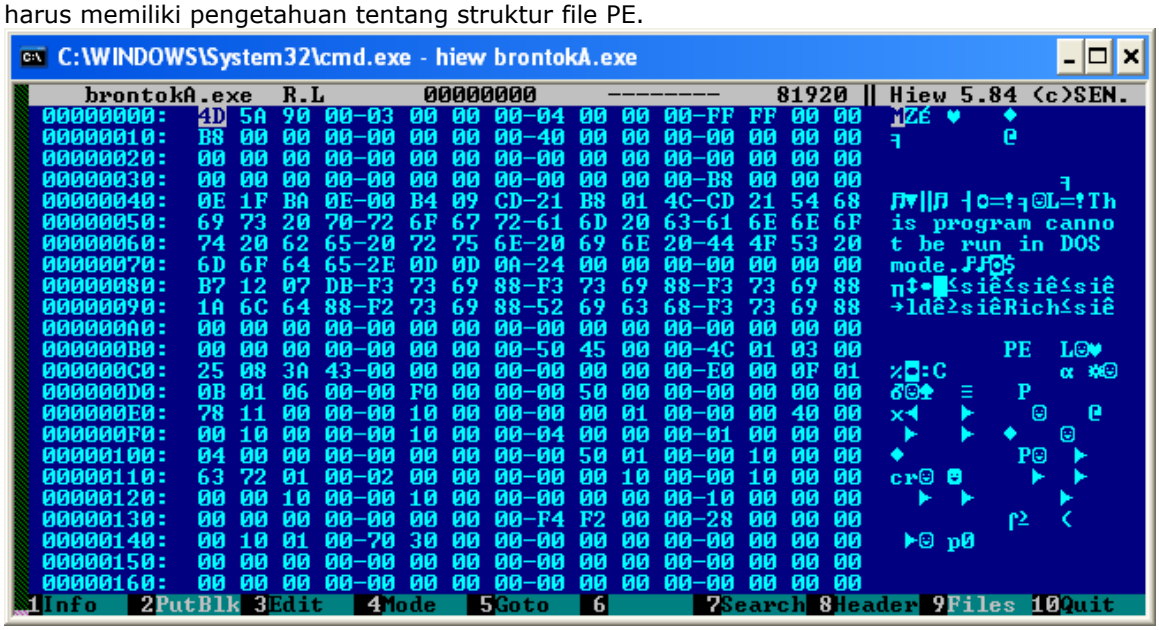
11.1 Inisialiasi Pola Worms

Agar program aplikasi dapat mengenali Worms Brontok.A, maka kita harus memasukan pola Brontok.A kedalam Virus Memory Definition dan Virus File Definition.

Untuk menggambil pola worms, pertama kita harus melakukan proses view terhadap isi file worms yang dimaksud. Adapun tools bantuan yang dapat digunakan adalah Hexa View yang dapat diaktifkan dengan :

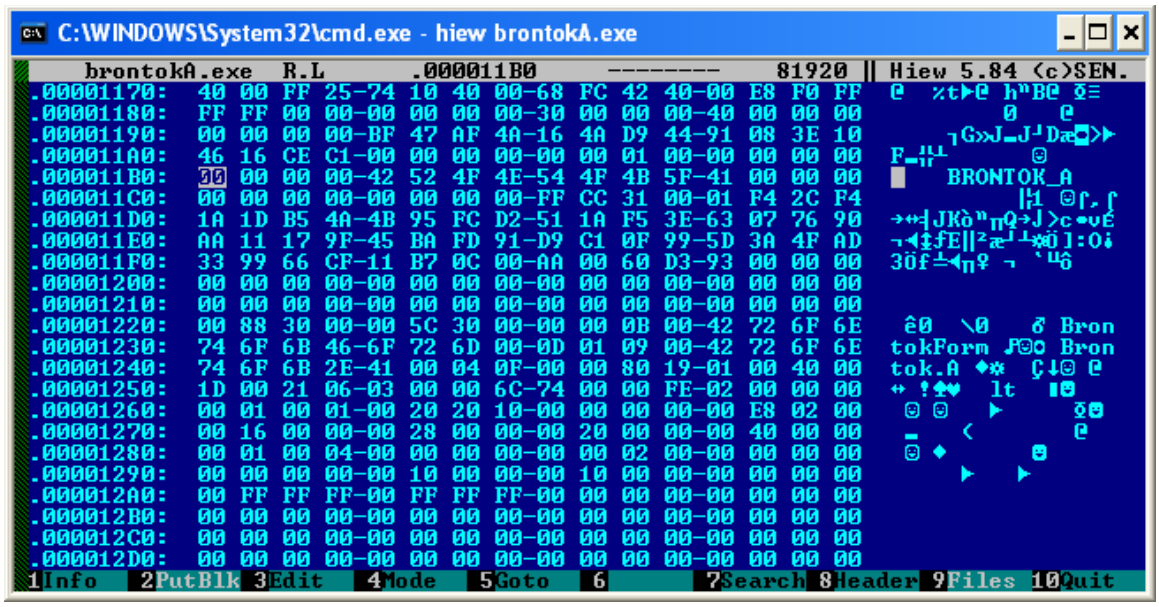
C:\Hiew BrontokA.exe

Untuk memudahkan anda dalam melakukan analisa terhadap isi file worms, biasanya anda harus memiliki pengetahuan tentang struktur file PE.



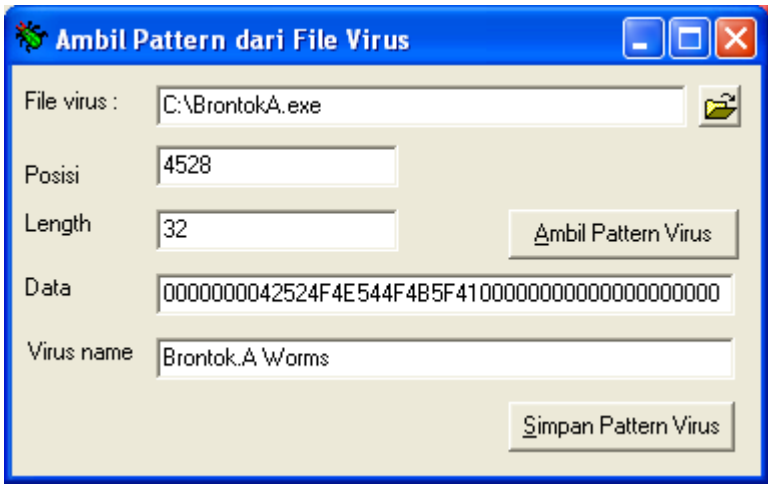
Gambar 15. Tampilan Hexa MSDOS Header program Worms Brontok.A
(Sumber: Ekplorasi oleh peneliti)

Selanjutnya kita dapat melakukan view terhadap isi file, dan mencari suatu pola yang mudah dikenali, dalam hal ini kita dapat menemukan kata Brontok.A dalam isi file worms yang dimaksud disekitar offset 11B0 hexa (4528 desimal)



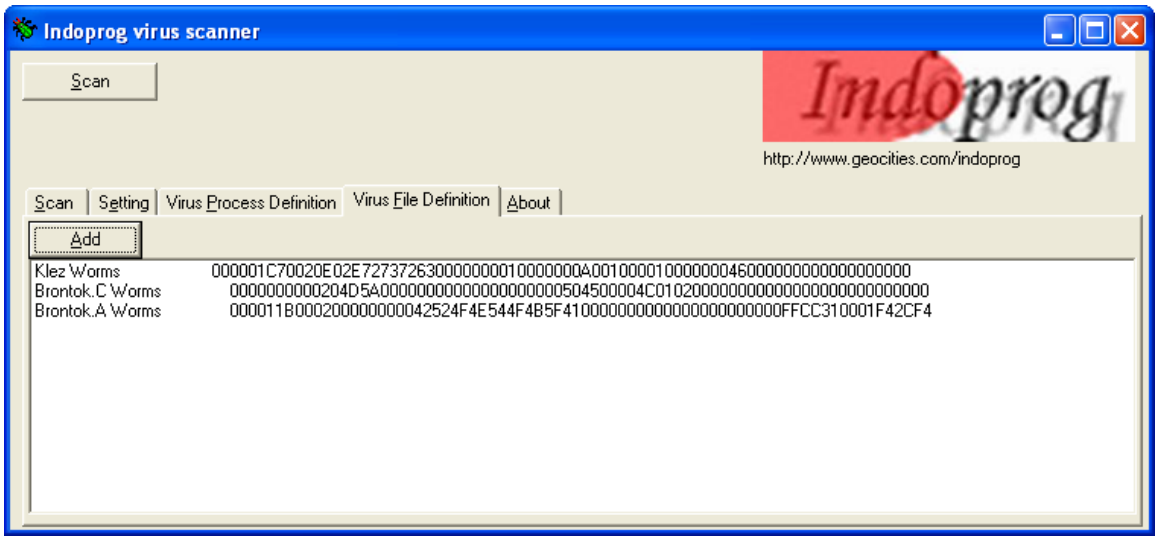
Gambar 16. Kata Brontok.A pada offset 11B0 hexa untuk pola worms.
(Sumber: Ekplorasi oleh peneliti)

Kemudian aktifkan program aplikasi, dan klik pada tombol Add pada halaman Virus Memory Definition, dan isikan data yang sesuai, klik pada Ambil Pola Virus, dan klik pada Simpan Pola Virus.



Gambar 17. Menambah Pola Brontok.A Worms ke Virus Definition.

Setelah penambahan pola, tutup dan aktifkan kembali aplikasi, dan pola Brontok.A telah dicantumkan ke database virus definition.



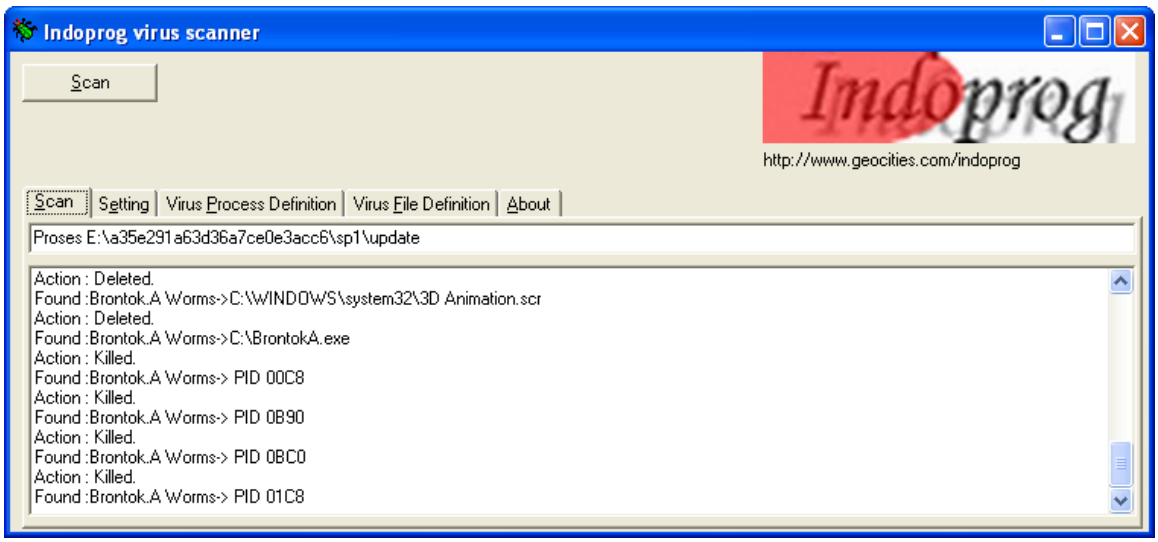
Gambar 18. Daftar Virus Defininiton

11.2. Pengujian Aplikasi.

Untuk melakukan pengujian program aplikasi, maka dibutuhkan suatu sistem komputer yang telah terinfeksi dengan worms Brontok.A.

11.2.1 Deteksi Process Worms

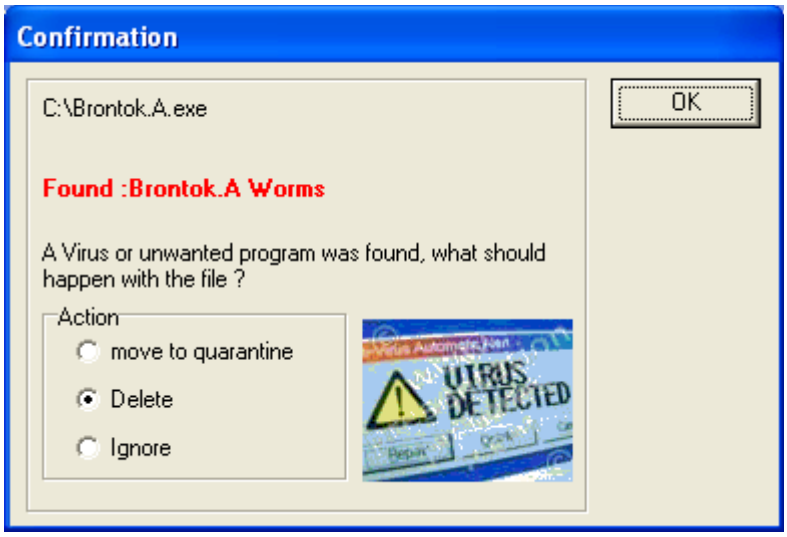
Jika program aplikasi berhasil mendeteksi adanya process yang menyerupai pola Worms yang tersimpan pada Virus Definition, maka akan menampilkan PID dari process tersebut dan menghentikannya (Killed).



Gambar 19. Berhasil ditemukan process Bontrok.Worms A

11.2.2. Deteksi file Worms

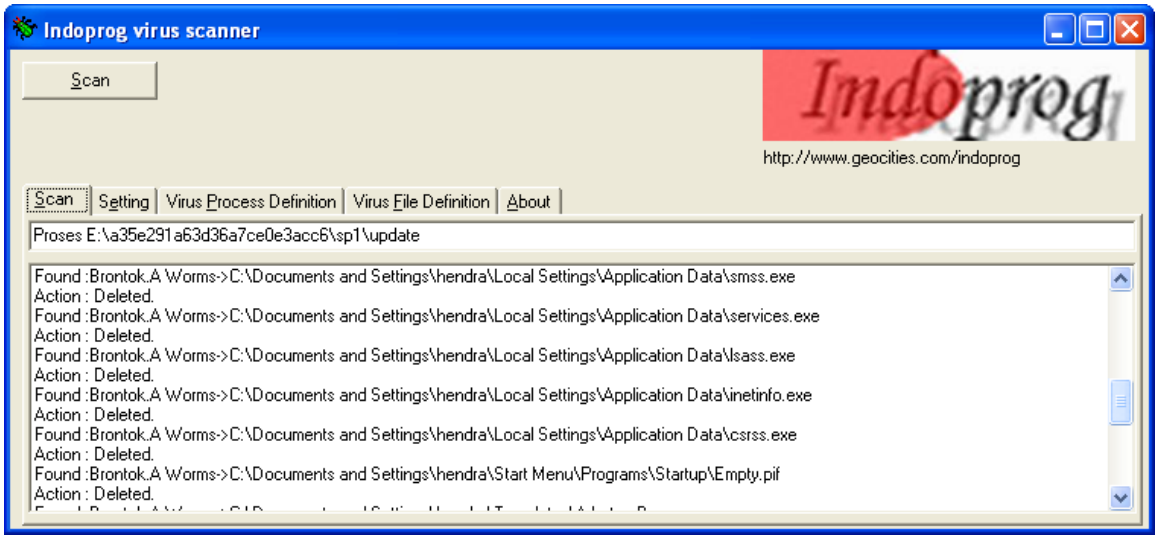
Selanjutnya program aplikasi akan mendeteksi semua file pada sistem dan menberikan konfirmasi terhadap aksi yang akan dilakukan :



Gambar 20. Konfirmasi temuan file Brontok.A Worms

11.2.3 Akhir proses

Akhirnya akan ditampilkan pesan dan log hasil pendeteksian.



Gambar 21. Log Hasil Pendeteksian

12. Kesimpulan

1. Sebagaimana file executable Windows biasanya, program Worms komputer memiliki struktur PE file.
2. Kunci utama pendeteksian worms computer adalah ketersediaan signature unik yang digunakan sebagai pola pendeteksian.
3. Pengambilan beberapa byte awal executable worms dikombinasikan dengan alamat fisik AddrOfEP dapat dijadikan sebagai signature untuk mendeteksi worms tersebut.
4. Aplikasi tidak perlu dijalankan secara safemode karena memiliki kemampuan deteksi process worms dimemori dan menghentikannya.
5. Aplikasi telah dilengkapi dengan kemampuan pemulihan registry ke nilai default.

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

13. Saran

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

Daftar Pustaka

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

John Shoch, Jon Hupp (1982), The "Worms" Programs - Early Experience with a Distributed Computation", URL : http://vx.netlux.org/lib/ajm01.html.

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.

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

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

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

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

Biodata Penulis :

Nama Lengkap	: Hendra
Gelar yang diperoleh	: ST. (Jurusan Teknik Industri, USU-Medan) MT. (Jurusan Teknik Industri, USU-Medan)
Afiliasi	: Teknik Informasi, STMIK IBBI
Jabatan	: Dosen Tetap
Mata kuliah yang diajar	: Bahasa Rakitan, Keamanan Komputer, Database Lanjut