Virus Scanning Simulation dengan Menggunakan Algoritma KMP

Achmad Giovani / 13508073

Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
if18073@students.if.itb.ac.id

Makalah ini menbahas tentang *Virus Scanning Simulation* dengan Menggunakan Algoritma KMP. Konsep utama dari simulasi *Virus Scanning* sederhana ini adalah menggunakan prinsip *pattern matching*. *Pattern matching* memiliki berbagai jenis algoritma, termasuk algoritma KMP (Knuth-Morris-Pratt) yang digunakan pada makalah ini. *Scanning* dilakukan terhadap daftar nama *file* (TOC/table of contents) dan isi dari *file*. Setiap *file* pada folder repository dicek apakah mengandung *pattern* yang merupakan ciri khas *Virus* dan ciri-ciri lain *Virus*, seperti mengubah ekstensi *file*. Karena ini hanya *Virus scanner* sederhana, maka yang *file* yang dicek hanya pada satu folder repository dan *Virus* yang teridentifikasi hanya 5 macam. Dan jika ada *file* lain yang mengandung *pattern* khas *Virus* (*Virus signature*) saja namun ciri lainnya tidak diketahui, *scanner* hanya melaporkan *file* tersebut sebagai *Virus* Suspected, tidak ada nama *Virus*nya. Selain berupa algoritma dan pseudocode, dalam makalah ini juga dilampirkan pranala program (program dibuat *web-based*, dengan bahasa *PHP*) di bagian akhir makalah. Pada pranala tersebut, *Virus Scanning* sederhana ini dapat dieksekusi.

Kata kunci : Virus, scan, pattern, KMP

1. PENGENALAN

Virus komputer adalah program komputer yang dapat memperbanyak dirinya sendiri dan dapat menginfeksi sistem operasi. Target dari *Virus* adalah :

a. *Binary executable file*
b. *Volume boot records* dari Floppy Disk
c. *Master Boot Record* dari harddisk
d. *File script* serbaguna, misalnya *file* VBScript.
e. *Autorun script file*
f. Dokumen seperti *Ms. Word*, *Excel*, atau *Access*
g. Celah pada program

Cara *Virus* untuk menghindari deteksi antara lain :

a. Tidak mengubah atribut "last modified" dari sebuah *file*
b. Tidak mengubah ukuran *file*, dengan cara mengoverwrite isi *file*
c. Menghindari *file* umpan (biasanya dibuat oleh *antivirus)*
d. Menyamar, seakan-akan seperti proses normal dari sistem
e. Menghentikan aktivitas *antivirus*
f. Memodifikasi dirinya sendiri, agar *Virus signature* berubah, sehingga tidak terdeteksi
g. Mengenkripsi diri
h. *Polymorphic code.* Mirip dengan enkripsi, namun kode berubah pada setiap kali infeksi, namun tetap serupa.
i. *Metamorphic code*. *Virus* me-*rewrite* (menulis ulang) dirinya sendiri tiap kali *Virus* menginfeksi *file executable* lainnya.

Jika dicermati, Perkembangan *Virus* dari waktu ke waktu terus menarik, ini terkait dengan perkembangan teknologi komputer/ teknologi informasi yang tumbuh dengan pesat.sejarah perkembangan *Virus* sendiri cukup menarik untuk diikuti.

Sayangnya tidak ada kepastian kapan pertama kali *Virus* komputer ditemukan.Ada sumber yang mengatakan bahwa sejarah *Virus* berawal Pada tahun 1949 ketika John Von Neuman, mengungkapkan "teori self altering automata" yang merupakan hasil riset dari para ahli matematika.

Kemudian pada tahun 1960 para ahli di lab BELL (AT&T) mencoba-coba teori yang diungkapkan oleh John V Neuman, mereka bermain-main dengan teori tersebut untuk suatu jenis permainan/game. Para ahli tersebut membuat program yang dapat memperbanyak dirinya dan dapat menghancurkan program buatan lawan. Program yang mampu bertahan dan menghancurkan semua program lain, maka akan dianggap sebagai pemenangnya. Permainan ini akhirnya menjadi permainan favorit ditiap-tiap lab komputer. semakin lama mereka pun sadar dan mulai mewaspadai permainan ini dikarenakan program yang diciptakan makin lama makin berbahaya, sehingga mereka melakukan pengawasan dan pengamanan yang ketat.

Sementara *Virus Scan* Software menemukan *Virus* pertama kali pada awal th 1970-an dimana dua program komputer yang diberi nama *Pervading Animal* dan *Christmas Tree* menginfeksi sistem berbasis Univac

1108 dan IBM 360/370. Karena sifatnya tidak merusak, kedua program ini belum dikenal sebagai *Virus*.

Namun sumber-sumber lain seperti wikipedia menyebutkan sebuah program komputer bernama "Elk Cloner" diyakini sebagai *Virus* komputer pertama yang dibuat Rich Screnta pada tahun 1982 dan menyebar pada sistem operasi Apple Disk Operating System (DOS) Versi 3.3 melalui media floppy disk

Pada tahun 1987 muncullah jenis *Virus* baru yang mulai menerapkan algoritma replikasi di kode programnya. Sebut saja the Leigh,*Virus* yang menginfeksi *file* command.com ini berhasil menular ke banyak sistem yang mengoperasikan DOS. Ternyata banyak yang terinspirasi dengan The Leigh ini, karena setahun berikutnya muncul *Virus* Jerussalem yang menginfeksi hanya pada tgl 13 tiap bulannya.Jerussalem dikenal sebagai *Virus* pertama yang dikategorikan penyebab kerusakan.*Virus* ini menghapus program yang sedang berjalan pada tgl penularan.

Karena perkembangan *Virus* ini dinilai sudah mengganggu pengguna komputer, mulailah dicari cara pencegahan program jahat ini beberapa pengembang independen mulai membuat program untuk menghilangkan atau menghentikan aktivitas *Virus*, yang kemudian dikenal dengan *antivirus*. Perusahaan besar pertama yang membuat *antivirus* adalah Symantec dengan produk norton *antivirus* di th 1990.

Uniknya, begitu *antivirus* ditemukan, perkembangan *Virus* justru semakin menjadi-jadi. Di tahun 1992 tercatat sebanyak 1300 *Virus* beraktivitas menginfeksi komputer di seluruh dunia, atau meningkat 420 % dari tahun 1990. jika di awal perkembangan *Virus file* program menjadi sasaran utamanya , th 1995 *Virus* hadir dengan metode penularan yang berbeda. *File* dokumen produksi Microsoft (Microsoft Word) menjadi sasaran baru *Virus* yang memanfaatkan kelemahan feature makro yang ada di program Microsoft, sehingga *Virus* dikenal dengan *Virus* makro.

Perkembangan berikutnya *Virus* komputer mulai melirik media internet sebagai basis penyebaran program mereka. Ide penyebaran *Virus* ini berawal dari keberhasilan *Virus* Melissa (W97M/Melissa) yang memanfaatkan kelemahan *file* mikro yang disisipkan ke e-mail, karena pada saat itu banyak pengguna e-Mail yang menggunakan aplikasi e-mail klien seperti Outlook, dan menyimpan kontak penerima disana sehingga dengan mudah melissa menyebar ke daftar kotak yang ada.

Perkembangan *Virus* ini terus dikonsentrasikan pada penyebaran via Internet, karena selain cepat disebarkan, *Virus* ini juga dapat mencuri data menggunakan media yang sama. Akan tetapi, berbeda dengan pendahulunya , *Virus* yang ada saat ini justru tidak memiliki, atau mungkin sengaja tidak difokuskan untuk merusak sistem. Beberapa *Virus* hanya sekedar unjuk gigi dalam melewati program *antivirus.*

Nama yang disandang oleh generasi baru *Virus* pun ikut bergeser. Kini *Virus* sudah memiliki saudara seperjuangan bernama *worm* dan *trojan.* Apa pun nama atau sebutan program mungil ini , ide utamanya ditujukan untuk mengganggu sistem normal yang sedang berjalan.

Perkembangan *antivirus* dari masa ke masa :

1. Generasi pertama : "*scanner* sederhana". *Antivirus* mens*can* program untuk menemukan *signature Virus.* Teknis ini terbatas untuk deteksi *Virus*-*Virus* yang telah dikenal.

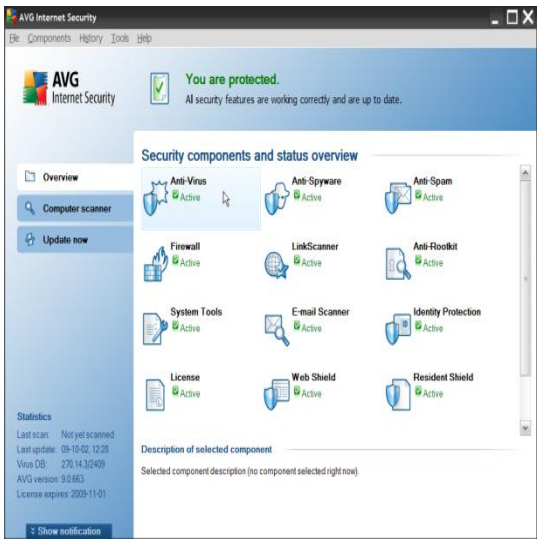
2. Generasi kedua : "*scanner* yang pintar" (***heuristic scanner***). *Antivirus* menggunakan aturan-aturan pintar (heuristic rules) untuk mencari kemungkinan infeksi *Virus.*

3. Generasi ketiga : jebakan-jebakan aktivitas (***activity trap***). Program *antivirus* merupakan program yang menetap di memori (memory resident program). Program ini mengidentifikasi *Virus* melalui aksi- aksinya bukan dari struktur program yang diinfeksi.

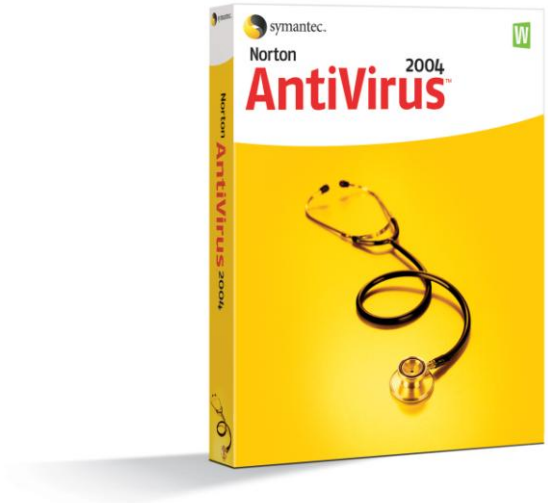
4. Generasi keempat : proteksi penuh (***full featured protection***). *Antivirus* generasi ini menggunakan beragam teknik *antivirus* secara bersamaan. Teknik-teknik ini meliputi *Scanning* dan jebakan-jebakan aktivitas.

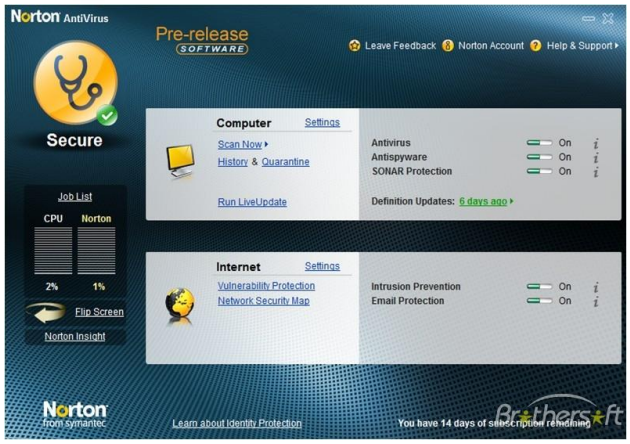
Berikut ini adalah beberapa contoh *antivirus* :

1. AVG *Antivirus*

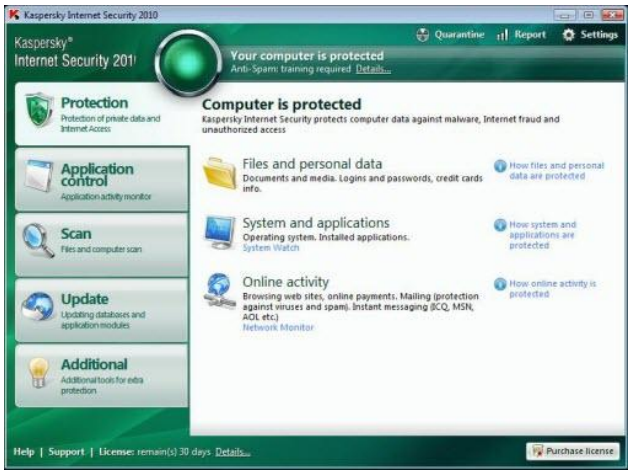


2. Norton *Antivirus*

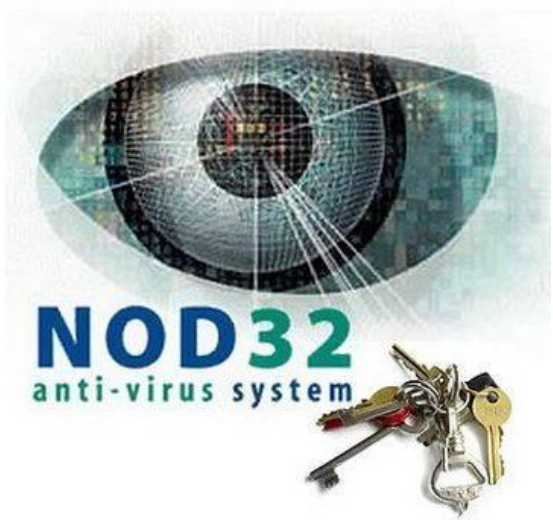


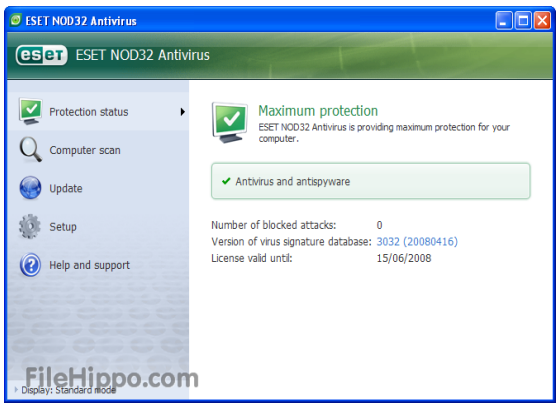


3. Kaspersky

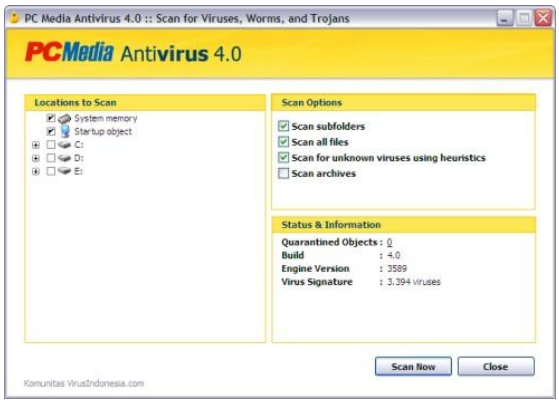


4. Eset NOD32





5. **PC Media *Antivirus***



6. **SMADAV**

Virus Scanning adalah proses pemindaian *file-file* yang tersimpan pada suatu tempat (direktori atau *drive*) untuk mengecek apakah ada *Virus* pada tempat tersebut. Jika ada maka *Virus scanner* dapat memberikan peringatan atau bahkan menghapus *Virus* tersebut. Atau lebih canggih lagi, *Virus scanner* dapat menyembuhkan sistem operasi dari infeksi *Virus* secara total. Namun, pada makalah ini, *Virus scanner* yang dibuat hanya yang sangat sederhana. Kemampuannya hanya sebatas mengidentifikasi dan menghapus *file-file* aneh yang seharusnya tidak ada.

Sedangkan *pattern matching* adalah proses pengecekan keberadaan sebuah *pattern* (pola) pada suatu *string*. *Pattern matching* sangat berguna dalam banyak hal, seperti untuk *search engine*, mengidentifikasi sekuens DNA, dan lain-lain. Ada banyak algoritma yang dapat digunakan untuk melakukan *pattern matching*, yakni algoritma Brute Force, Boyer-Moore, dan tentu saja KMP, serta masih banyak lagi teknik lain untuk melakukan *pattern matching* (misalnya dengan menggunakan *regular expression*).

Algoritma KMP yang digunakan dalam makalah ini merupakan optimisasi dari algoritma Brute Force. Prinsip Brute Force adalah melakukan pencarian dengan menggeser *pattern* per karakter, ini kurang mangkus jika dibandingkan KMP yang memanfaatkan fungsi pinggiran. Fungsi pinggiran ini tidak menambah kompleksitas karena hanya dibuat sesaat sebelum pencarian, kemudian langkah pencarian selanjutnya dilakukan dengan memanfaatkan tabel yang berisi hasil dari penghitungan fungsi pinggiran.

2. METODE

Metode *Virus Scanning Simulation* ini adalah dengan menggunakan *pattern matching*. Algoritma yang digunakan adalah KMP (Knuth-Morris-Pratt). Cara kerja algoritma ini adalah :

a. Pencocokan karakter dilakukan dari kiri ke kanan

b. Mencari *prefix* terpanjang dari P[0..j-1] yang juga merupakan *suffix* dari P[1..j-1], untuk menghindari pergeseran yang tidak perlu.
c. Hasil dari pencarian *prefix* terpanjang disimpan pada tabel. Ini disebut juga *Failure Function*.
d. Misalkan panjang *string* yang telah diperiksa dan cocok = N dan nilai dari *Failure Function* adalah M, maka pergeseran dilakukan sebanyan (N-M).

Contoh :

String : abacaabaccabacabaabb
Pattern : abacab

Tabel *Failure* :

0	1	2	3	4
0	0	1	0	1

Langkah-langkah:

```
abacaabaccabacabaabb
abacab


abacaabaccabacabaabb
    abacab

abacaabaccabacabaabb
     abacab

abacaabaccabacabaabb
         abacab

abacaabaccabacabaabb
          abacab
```

Contoh penjelasan lengkap ada pada pranala berikut ini :
http://oak.cs.ucla.edu/cs144/examples/KMPSearch.html

Untuk identifikasi *Virus*, metode yang digunakan adalah :
a. Pindai semua *file* pada folder repository
b. Cek *pattern* khas *Virus* (dalam makalah ini, *pattern* (*Virus signature*) yang digunakan adalah "~~~~~~~~~"), jika ada maka *Virus* Suspected.
c. Cek ciri-ciri dari kelima *Virus*, apakah ditemukan? Jika ditemukan, maka ditampilkan nama *Virus*nya. *Virus* yang dibuat dalam makalah ini hanya lima macam :
   - ***Virus* Maho**
     Menyisipkan *Pattern* "SAYA MAHO SAYA MAHO SAYA MAHO"
   - ***Virus* Unyu**
     Menyisipkan *pattern* ":3 :3 :3"
   - ***Virus* Iseng**
     Mengganti ekstensi *file* menjadi .iseng dan menyisipkan *string* "WEEEEEEEEEK!!! :P". Nama *file* tetap seperti aslinya.
   - ***Virus* Zzzz**
     Menambahkan sebuah *file* bernama "zzzz.wew"
   - ***Virus* BB**
     Menambahkan sebuah *file* bernama "hotgirls.bb"
d. Jika ditemukan beberapa *Virus* tersebut, maka akan muncul peringatan berupa teks. Khusus untuk *Virus* Zzzz dan BB, *file* aneh yang ditambahkan akan langsung dihapus.

3. IMPELEMENTASI

Program *Virus Scanner* dibuat *web-based* dengan menggunakan bahasa PHP, namun yang dicantumkan di makalah ini hanya notasi algoritmik dan *pseudocode* saja. Berikut ini adalah *header* dari setiap fungsi yang digunakan, berikut penjelasannya :

```
function storeActualFileName() -> array of string

/* menyimpan daftar nama file */
```

Fungsi ini menyimpan daftar nama *file* yang ada pada direktori "*file*_repo" pada saat itu, tipenya *array of string* (meskipun dalam PHP, tipe variabel tidak perlu dipikirkan).

```
function computeFail(pattern : string) -> array of integer

/*Fungsi untuk menghitung Failure Function */
```

Fungsi ini digunakan pada fungsi searchKMP (fungsi yang melakukan pencarian *string* dengan algoritma KMP).

```
function searchKMP(text : string, pattern : string) -> integer

/* Fungsi pencarian pattern dengan algoritma KMP  */
```

Fungsi ini untuk mencari *pattern* dengan menggunakan algoritma KMP.

```
function searchOneFile(namafile : string, pattern : string) -> boolean

/* Fungsi untuk mencari pattern dalam satu file */
```

Fungsi ini untuk mencari *pattern* dalam suatu *file*. pencarian dilakukan dengan pembacaan per baris dari isi *file*, kemudian dimasukan dalam variabel yang bertipe

string, untuk kemudian dibandingkan dengan *pattern* yang ingin dicari.

```
function readTOC() -> array of string

/*Fungsi untuk membaca TOC (daftar nama file
yang disimpan dalam file TOC.txt */
```

Fungsi ini untuk membaca TOC yang disimpan dalam *file* TOC.txt. Daftar nama *file* disimpan dalam *array of string*. Hasil dari fungsi ini nantinya akan dibandingkan dengan hasil fungsi *storeActualFileName()* untuk identifikasi *Virus*.

```
function cekVirMaho() -> array of string

function cekVirUnyu()-> array of string

function cekVirIseng()-> array of string

function cekVirZzzz()-> array of string

function cekVirBB()-> array of string

/* Lima fungsi untuk mengecek keberadaan
masing-masing jenis Virus */
```

Kelima fungsi di atas adalah untuk melacak keberadaan lima jenis *Virus* yang sudah dijelaskan sebelumnya (Maho, Unyu, Iseng, Zzzz, dan BB). Fungsi ini menghasilkan daftar *file* yang terinfeksi (kecuali untuk *cekVirZZZ()* dan *cekVirBB()*). Masing-masing *virus* punya metode pencarian yang mirip, yaitu :

1. Cek jumlah dan nama file sebenarnya (simpan sebagai *array of string*), apakah sama atau beda dengan TOC. (hanya untuk *virus* Zzzz dan BB)
2. Baca semua file, cek *virus* signature (“~~~~~~~~”)
3. Kemudian cek ciri khas *virus*, misalnya pattern “:3 :3 :3” untuk *virus* Unyu.
4. Jika ada, simpan nomor urut file ke dalam *array of integer*
5. Cocokkan nomor urut dengan nama file yang telah disimpan. Simpan ke dalam *array of string* yang berisi nama file *virus*
6. Khusus *virus* Iseng, jika telah ditemukan file yang berbeda dengan TOC, cek nama file, apakah sama dengan aslinya, kemudian cek ekstensinya, apakah “.iseng”, kemudian cek isi filenya. Jika ada *pattern* “WEEEEEEEEK!!! :P”, maka bisa dipastikan itu *virus* Iseng
7. Untuk pengecekan *virus* Suspected, yang dilakukan hanya mengecek keberadaan *pattern* “~~~~~~~~”.
8. Tampilkan semua nama file *virus*.

```
function cekSemuaVirus() -> array of (array of
string)

/*Fungsi yang merupakan inti dari program ini
*/
```

Fungsi ini mengecek keberadaan *Virus* dengan cara mencari *pattern* khas *Virus*, yaitu “~~~~~~~~” dan mengidentifikasi jenis *Virus* berdasarkan ciri-ciri yang telah dijelaskan sebelumnya. Fungsi ini menjalankan kelima fungsi pengecekan *Virus*. Fungsi ini menghasilkan daftar *file* terinfeksi, dikelompokkan berdasarkan jenis *Virus*nya. Tipenya adalah array of (array of *string*).

Pseudocode dari cekSemua*Virus*() adalah :

```
function cekSemuaVirus(){

       //simpan daftar nama file
       allfilename = storeActualFileName()
      //simpan daftar nama file yang ada di TOC
       toc = readTOC()
       //inisialisasi
       for (semua file){
              ketemu[i] = false
       }
       j = 0;

//untuk semua file
       for (semua file){
              namafile = "file_repo/" +
allfilename[i]

              if (nama file bukan TOC.txt){
                     cari pattern "~~~~~~~~"
menggunakan KMP

              }
              if (ketemu[i]) {
                     nomorfile[j] = i
       //simpan nama file yang Virus Suspected
                     namaFileTerinfeksi[j] =
toc[nomorfile[j]]
                     j++
              }
       }

//keterangan output
       /*
       [0] = VirusMaho
       [1] = VirusUnyu
       [2] = VirusIseng
       [3] = VirusZzzz
       [4] = VirusBB
       [5] = suspected
       */
//cek kelima jenis Virus
       output[3] = cekVirZzzz()
       output[4] = cekVirBB()
       output[0] = cekVirMaho()
       output[1] = cekVirUnyu()
       output[2] = cekVirIseng()
       output[5] = namaFileTerinfeksi

       cekSemuaVirus =  output
}
```

4. KESIMPULAN

Prinsip *pattern matching* dapat digunakan dalam pencarian *Virus* (*Virus Scanning*). Karena pada dasarnya, setiap *Virus* memiliki ciri khas. Dalam makalah ini ciri khas tersebut digambarkan dengan *pattern*, sedangkan aslinya, *pattern Virus* ini dilacak pada sekuens bit dari *file* tersebut, dan tingkah laku *Virus* dilacak dengan cara memindai *registry*, *startup programs*, *services*, *process*, dan lain-lain. Namun itu sudah pada level yang lebih rendah (*low-level*), sangat dekat dengan sistem operasi, sedangkan *Virus scanner* pada makalah ini tergolong *high level*, karena hanya membaca isi *file* saja.

Pencarian dengan algoritma KMP juga cukup cepat. Ini membuktikan bahwa algoritma KMP lebih mangkus daripada algoritma Brute Force.

DAFTAR REFERENSI

[1] http://www.microsoft.com/security/antivirus/whatis.aspx
Tanggal akses : 4 Desember 2010

[2] http://ranger.uta.edu/~gdas/Courses/Fall2004/advAlgos/student_slides/W9Presentation.ppt
Tanggal akses : 5 Desember 2010

[3] http://www.antivirusworld.com/articles/history.php
Tanggal akses : 5 Desember 2010

[4] http://oak.cs.ucla.edu/cs144/examples/KMPSearch.html
Tanggal akses : 5 Desember 2010

[5] http://www.howstuffworks.com/Virus.htm
Tanggal akses : 6 Desember 2010

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 6 Desember 2010

Achmad Giovani
13508073

LAMPIRAN

Aplikasi *Virus Scanning Simulation* ini dapat dieksekusi pada pranala berikut :

http://achmadgiovani1712.freehostia.com/VirSimulator/