PENDEKATAN REVERSE ENGINEERING UNTUK PENGUJIAN KEAMANAN GUNA MENINGKATKAN KUALITAS PERANGKAT LUNAK

Muhammad Reza Redo I

Staf pengajar pada program studi Ilmu Komputer Jurusan Teknik Informatika
Sekolah Tinggi Manajemen Informatika dan Komputer Darmawacana
Jl. Kenanga No. 3 Mulyojati Metro Barat, Metro
082186663000
Emai. reza.redo@hotmail.com

ABSTRACT

A software development will take time and high cost. Reverse engineering is a technique or a way to find the principles of the technology of a product by analyzing the structure, function and how the product. By studying the structure of the functions of a product, this study aims to patching into software to improve the quality of the software itself. Patching which will be tested in this method is Checksum, Obfuscation / Obfuscated Code, API Redirection, Anti-Dumping where the results of the patching will be tested quality and the measured length of the Code, the Output File Results, and the number of changes that occur so as to increase the quality of the software itself

Keywords : Reverse engineering, Patching, Checksum, Obfuscation / Obfuscated Code, API Redirection method, Anti Dumping Software

ABSTRAK

Pengembangan sebuah perangkat lunak akan memakan waktu dan biaya tidak sedikit. Reverse engineering merupakan teknik atau cara untuk menemukan prinsip-prinsip teknologi suatu produk dengan cara menganalisa struktur, fungsi dan cara pada produk tersebut. Dengan mempelajari struktur fungsi dari sebuah produk penelitian ini bertujuan untuk melakukan patching kedalam sebuah perangkat lunak guna meningkatkan kualitas dari perangkat lunak itu sendiri. Patching yang akan diujikan dalam metode ini adalah Checksum, Obfuscation/ Obfuscated Code , API Redirection, Anti Dumping dimana dari hasil patching tersebut akan di uji kualitas dan diukur Panjang Kode , Output File Hasil, dan Banyaknya perubahan yangterjadi sehingga terjadi peningkatan kualitas dari perangkat lunak itu sendiri

Kata kunci : Reverse engineering, Patching, Checksum, Obfuscation/ Obfuscated Code, API Redirection method, Anti Dumping Software

1. Pendahuluan

Dalam proses mengembangkan suatu piranti lunak pada masalah apapun, akan diawali olehtahapan analisa kebutuhan yang akan menghasilkan kebutuhan. Permasalahan nya adalah pengembangan sebuah *software* akan memakan waktu dan biaya tidak sedikit dan apabila kita melakukan riset akan yang memakan waktu lama untuk perbaiki kekurangan nya. Untuk mendesain ulang *software* dari awal diperlukan sebuah teknik, sementara itu dalam proses untuk menganalisa sebuah perangkat lunak teknik *reverse engineering* merupakan suatu proses menemukan prinsip-prinsip teknologi suatu produk dengan cara menganalisa struktur, fungsi dan cara pada produk tersebut. Pada penelitian ini diharapakan dengan dengan teknik *reverse engineering* dapat dilakukan pengujian dan patching pada perangkat lunak (*software*) terutama pada sisi keamanan (*security*). Kenapa produknya yang harus di buat aman?, karena aman itu bukan hanya terbebas dari serangan virus, akan tetapi perangkat lunak tersebut juga tidak akan menjadi media penyebar virus itu sendiri. Karena tujuan dari keamanan adalah memberikan kenyamanan akan tetapi proses yang dilakukan untuk mengamankan sendiri sering berbanding terbalik dengan kenyamanan, hal ini disebabkan karena terlalu banyak kaidah/aturan yang berlaku untuk mengamankan perangkat lunak tersebut dan jika seseorang ingin merasa aman maka dia harus memasang sebuah program antivirus ke dalam komputernya, sementara antivirus sendiri tidak akan memberikan jaminan aman 100% terhadap serangan virus, Keamanan yang dimaksud di sini adalah melakukan patching dengan melakukan pemberian *checksum* pada file, pemberian *Redirect API*, pemberian *Anti Dump* dan pemberian *Obfuscation*, dengan harapan setelah pengujian tersebut akan di dapatkan sebuah *software* sama yang jauh lebih baik dari sisi keamanannya tanpa harus memasang antivirus.

Adapun Tujuan penelitian yaitu bertujuan untuk :

a. Untuk melakukan patching ter hadap sebuah aplikasi
b. Untuk menghasilkan Sebuah Produk baru dari kode sumber yang sama dengan produk lama
c. Untuk mencari tahu banyak

perubahan kode yang terjadi setelah dilakukan patching,

d. Untuk melakukan perban dingan Besar (size) file sebelum dan sesudah di lakukan patching

e. Untuk melakukan pembandingan dan mengetahui panjang File header panjang

badan File dan panjang ruang kosong Sebelum dan sesudah

f. Meningkatkan keamanan pada aplikasi yang akan di uji.***Tabel 1*** *Tahapan Proses pada Penelitian*

No	Proses	Sub proses
1	Meneliti literatur yang berhubungan dengan masalah penelitian	
2	Mengidentifikasi dan membatasi masalah	
3	Merumuskan recana	
4	Menyusun rencana secara lengkap dan operasional, meliputi:	Menentukan variabel Memilih desain yang digunakan
5	Melaksanakan eksperimen	Menerapkan perlakukan sesuai dengan metode algoritma dan desain yang telah di tetapkan Mengumpulkan data dari hasil eksperimen
6	Menyusun data	
7	Melakukan pengujian dan perbandingan	Melakukan pengujian Melakukan perbandingan dari subject yang telah diuji

2. Metode penelitian

Pada penelitian eksperimen dikenal beberapa variabel. Variabel adalah segala sesuatu yang berkaitan dengan kondisi, keadaan, faktor, perlakuan, atau tindakan yang diperkirakan dapat memengaruhi hasil eksperimen, dalam hal ini kondisi dan tindakan tententu akan diberikan kepada beberapa aplikasi perangkat lunak yang akan di jadikan bahan untuk experiment oleh penulis. Variabel yang berkaitan secara langsung dan diberlakukan untuk mengetahui suatu keadaan tertentu dan diharapkan mendapatkan dampak/akibat yang memungkinkan menghasilkan suatu kesimpulan untuk dapat di analisis.

Kerangka berfikir dan tahapan Proses Eksperimen. :

Variabel	Checksum	Obfugasi	API redirect	Anti Dumping
C_1	√			
C_2		√		
C_3			√	
C_4				
C_5	√	√		
C_6	√		√	
C_7	√			
C_8	√	√	√	
C_{10}	√	√	√	

Variabel yang akan di gunakan pada Proses penelitian :

Variabel "A" :

Variabel awal dimana File yang akan di uji coba belum mendapatkan perlakukan khusus dari Experimen yang akan di jalankan

Variabel "B" :

Variabel Threatmen dimana pada variabel ini adalah sebuah perlakukan khusus yang akan di tambahkan pada variabel A

Variabel "C" :

Variabel Hasil yang mengandung hubungan antara Variabel A dan B

Tabel 2 *Penentuan Indikator Variabel A*

Aplikasi	Threatmen	Diberikan	
		Ya	Tidak
A	Check Sum		√
	Anti Debug		√
	API redirection		√
	Anti Dumping		√
	Obfugasi		√
	Kompresi File		√
	Protect Debug Info		√

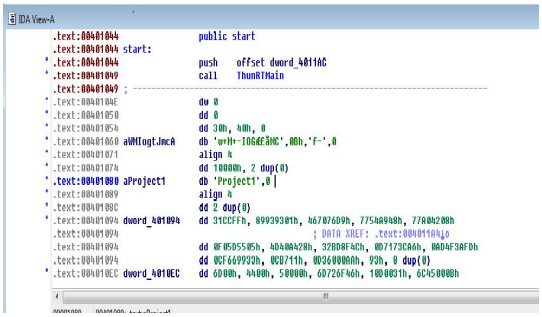
Tabel 4 *Identifikasi perlakuan untuk variabel C*

Variabel	Threatment/Perlakuan
B_1	Checksum
B_2	Obfugasi
B_3	API redirect
B_4	Anti Dump

3. Pembahasan

Pada Baris Algoritma Assembly program dummy (variable A) yang akan di patching program berjalan di offset : 00401044 <- merupakan alamat memory awal program ini berjalan.

```
.text:00401044              public start
.text:00401044                                  start:
.text:00401044              push    offset dword_4011AC
.text:00401049              call    ThunRTMain
.text:00401049 ; ---------------------------------------------------------
-----------
.text:0040104E              dw 0
.text:00401050              dd 0
.text:00401054              dd 30h, 40h, 0
.text:00401060 aWMIogtJmcA      db 'w+M+-
IOGÆ£åMC',0Bh,'f-',0
.text:00401071              align 4
.text:00401074              dd 10000h, 2 dup(0)
.text:00401080 aProject1        db 'Project1',0
```



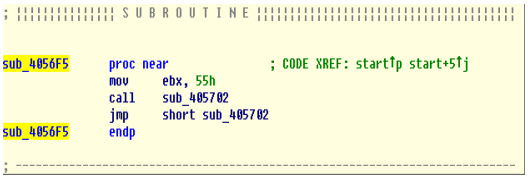
Gambar 1. *variabel C_1 offset pada IDA program*

Pada File yang telah Di Checksum (variable C_1) terdapat perubahan dimana

offset start program berjalan di alamat memory 004056ED yang merupakan alamat memori relative Pada sub rutin terjadi pemanggilan

004056ED **call sub_4056F5**
004056F2 jmp short sub_4056F5

Prosedur yang di panggil adalah : KODE XREF : Start +5, Yang merupakan salah satu algoritma untuk melakukan checksum pada file



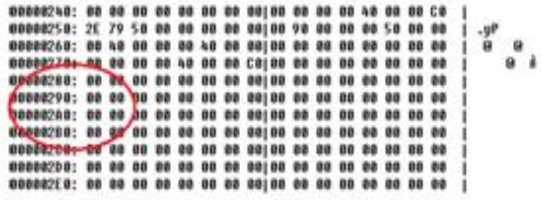
Gambar 2. *IDA View Sub Rutin Program awal di panggil*

Pengujian terhadap file yang telah di Checksum

Pengujian dan pembuktian terhadap File hasil patching akan di gunakan dengan merubah baris Kode hexa pada program itu sendiri untuk itu digunakan program bantuan Hexa Editor.

Gambar 3. *Hex Editor View File Checksum*

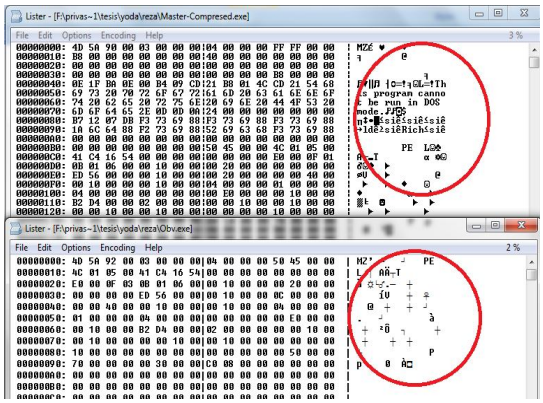
Penanda Offset awal menentukan apakah file tersebut ber jenis apa dalam hal ini offset awal berisi data 4D 5A (Portable executable) 8 bit awal penanda sebuah file. Pada pengujian ini akan dilakukan perubahan pada Offset: 00000290, yang memiliki data kosong 00



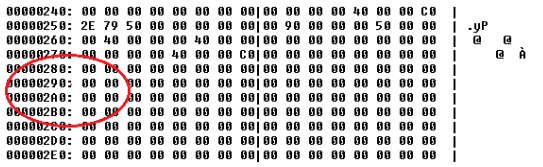
Gambar 4. *Hex Editor Zero data*

Kita bisa saja merubah ke offset mana saja tapi tidak kita lakukan karena perubahan data data sebaiknya di gunakan di bit yang bernilai kosong agar besar File tidak berubah, dan setelah perubahan dilakukan dengan menambahkan nilai AA pada offset ; 00000290 maka hasil hex Editor akan seperti ini

hal ini di maksudkan untuk mensimulasi terjadinya serangan Virus atau proses Cracking sebuah software. Jika sudah maka jalankan Program yang telah kita rubah dengan Hexa editor tersebut



Gambar 5. *Hex editor File Komparasi*



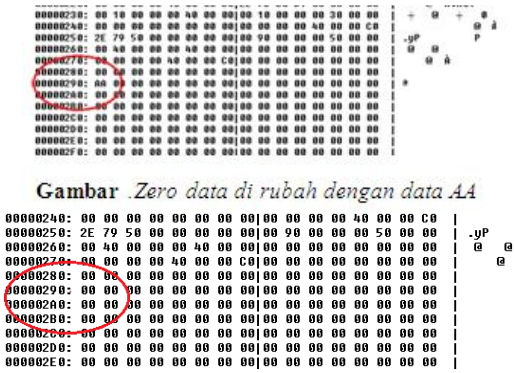
Gambar 6. *Error reporting Windows7*

Dari pembuktian tersebut maka dapat dibuktikan bahwa hasil patching Checksum pada program telah berhasil, dengan pemberian Checksum pada Program maka dapat dipastikan.

Program yang terinveksi Virus tidak akan menjadi media penyebar Virus itu sendiri, dan program tersebut akan otomatis tidak dapat dijalankan

Pengujian Pada File Yang Telah Di Berikan Obfuscation

Tujuan melakukan pengujian ini adalah untuk membuktikan apakah kode obfugasi telah berhasil terpatching kedalam program tersebut untuk itu yang kita lakukan adalah uji fisik kedalam kode program dengan cara membandingkan kode sebelum di patching dan esudah patching setelah itu yang kita lakukan adalah menguji kode tersebut dengan cara m elihat informasi dengan mengunakan hex editor



Gambar *.Zero data di rubah dengan data AA*

Gambar 7. *Hex editor zero data*

terdapat 2 file hasil perbandingan dengan mengunakan Hex Viewer dan file di gambar atas adalah file asli yang masih menyimpan informasi Headervdimana tertulis informasi “ this program cant run in MS DOS mode “ sementara pada file dibawah nya hasil obfugasi informasi tersebut sudah tidak dapat dibaca

Pengujian pada file yang telah API Redirection

API merupakan kumpulan fungsi-fungsi eksternal yang disediakan library windows untuk mengatur kemampuan dan tingkah laku setiap element di Windows

(dari tampilan di desktop hingga alokasi memory) sehingga dapat dimanfaatkan suatu program untuk meningkatkan kemampuan program tersebut.

Pada Baris Algoritma Assembly program dummy yang akan di patching sebelum patching dengan menggunakan Program IDA di tuliskan bahwa program berjalan di offset : 00401044 <- merupakan alamat memory awal program ini berjalan. Offset ini bisa saja berubah apabila program berjalan di mesin yang berbeda untuk itu dokumentasi sangat di perlukan pada saat penelitian ini di lakukan

```
.text:00401044          public start
.text:00401044 start:
.text:00401044          push    offset dword_4011AC
.text:00401049          call    ThunRTMain (fungsi yang digunakan untuk memanggil API di VB6 / program ini dibuat dengan VB6 )
.text:00401049 ; ---------------------------------------------------------------------------
.text:0040104E          dw 0
.text:00401050          dd 0
.text:00401054          dd 30h, 40h, 0
.text:00401060 aWMlogtJmcA db 'w+M+-IOG&EÂÂMC',0Bh,'f-',0
.text:00401071          align 4
.text:00401074          dd 10000h, 2 dup(0)
.text:00401080 aProject1    db 'Project1',0
```

Gambar 8. *View File sebelum API redirection*

Setelah proses Pacthing *API Redirection* dilakukan maka Kode Program akan berubah menjadi seperti ini :

```
API_Reza:004056ED ; ''''''''''''''' S U B R O U T I N E
'''''''''''''''''''''''''''''''''''''''
API_Reza:004056ED
API_Reza:004056ED
API_Reza:004056ED               public start
API_Reza:004056ED start         proc near
API_Reza:004056ED               call    sub_4056F5
API_Reza:004056F2               jmp     short
sub_4056F5
API_Reza:004056F2 start         endp
API_Reza:004056F2
```

Program akan dijalankan dimulai dari Offset : 004056F5 disana Fungsi API relocation pada

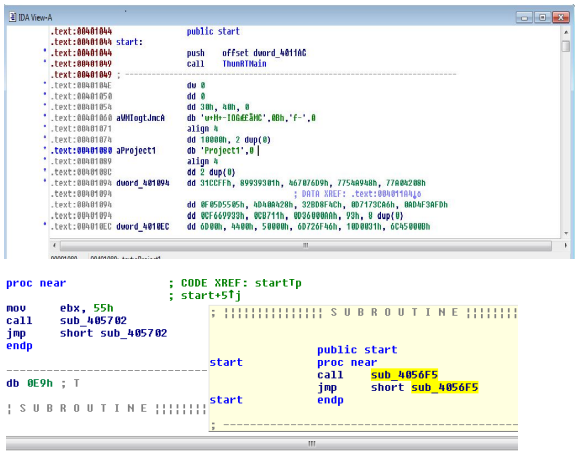
program di include kan kedalam baris perintah sehingga merubah Offset awal yang tadi nya di

mulai dari offset : 004011AC

Sudah tidak ada lagi Kode

```
00401044                push    offset dword_4011AC
.text:00401049          call    ThunRTMain
```

Dimana API fungsi sudah tidak terbaca lagi pada program ini



Gambar *9.View File API sesudah patch*

```
API_Reza:004056ED               public start
API_Reza:004056ED               start     proc near
API_Reza:004056ED               call
sub_4056F5
```

Offset : 004056ED Program mulai di jalankan

Kemudian memanggil prosedur yang berada di offset : 004056F5

Dimana offset : 004056F5 adalah Sub Rutin yang melakukan Fungsi API relocation yang kemudian me refer kembali ke rutin ke dua yaitu

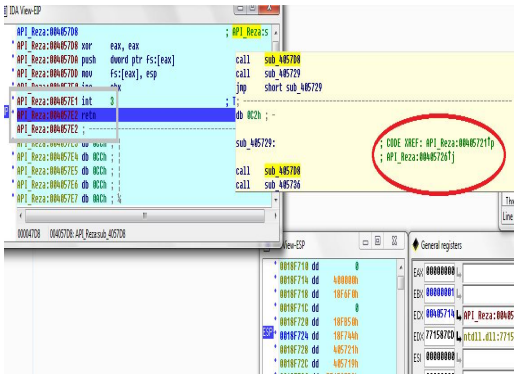
```
; ---------------  S U B R O U T I N E  ---------------------------------------

sub_4056F5      proc near               ; CODE XREF: start↑p start+5↑j
                mov     ebx, 55h
                call    sub_405702
                jmp     short sub_405702
sub_4056F5      endp

; ---------------------------------------------------------------------------
```

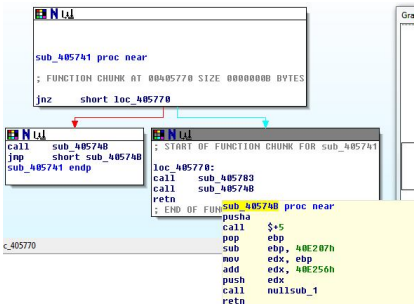
***Gambar** 10.View Rutin tambahan yang sebelumnya tidak ada*

Setelah itu pada Subrutin ke dua salin isi register EBX dengan 55h, dalam hal ini kita memang tidak mengetahui apa isi data 55h biasanya register BX digunakan untuk menunjukkan suatu alamat offset dari suatu segmen. Pada gambar dibawah terdapat potongan fungsi yang berisi operasi dimana jika sub rutin di offset 00405770 bernilai tidak kosong maka dia akan melakukan rutin ke potongan fungsi lain nya dimana potongan rutin tersebut akan memanggil sub rutin yang ada di offset 00045783 kemudian melanjutkan proses dibawah nya sampai ke offset 0040574B hingga selesai.



***Gambar 11.** IDA View File API*

Setelah kita tahu bahwa API relocation Fungtion/ Proteksi fungsi API telah dilakukan saat nya melakukan pengujian apakah fungsi API sudah benar benar samar dengan cara menjalankan program dan mengunakan tools ke dua untuk membuktikan apakah Proteksi API sudah berjalan dan ketika di buka dengan Olly debugger maka program olly debugger menampilkan pesan seperti di bawah ini . yang berarti telah terjadi modifikasi dari File yang kita ujikan sebelum nya, hal ini cukup membuktikan bahwa file tersebut telah berhasil terproteksi. Dengan menggunakan IDA pro program sudah tidak dapat di debug trace

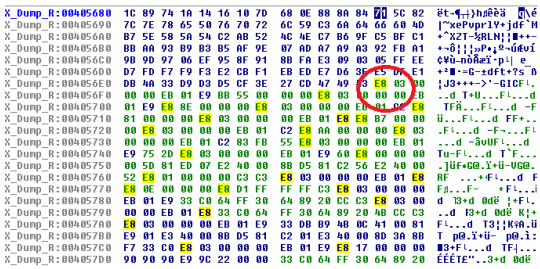


***Gambar** 12.File Tidak bisa Di trace*

Pada garis Lingkaran merah dengan text hijau menandakan proteksi API telah berjalan dan pada garis kotak warna abu-abu menandakan bahwa program tidak mau melanjutkan operasi operasi selanjutnya.

Pengujian Pada File Yang Telah Di Beri Anti Dumping

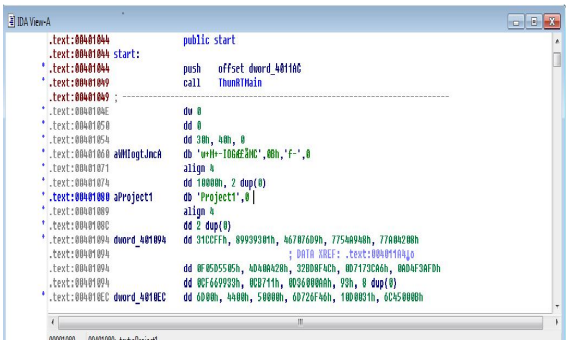
Sama seperti pada saat pengujian checksum yaitu mendokumentasi dari offset dimana file asli (File yang belum di patching) di jalankan yaitu di offset : 00401044



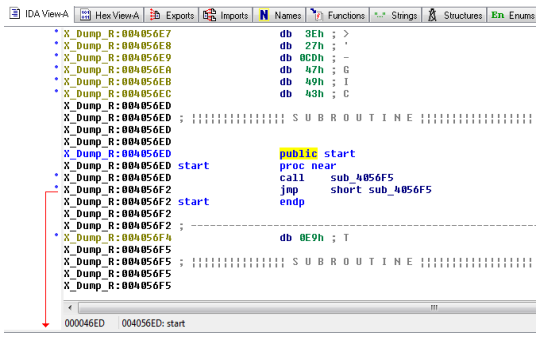
Gambar.13 *IDA view File yang akan di beri Anti Dump*

004056ED dengan Segment “X_Dump_R” setelah di bedah dilihat ternyata Kode patching telah berjalan di program ini setelah itu adalah melakukan pengujian apakah patching ini berhasil dengan cara membuat program membentuk sebuah dump dari memory yang dia buat kemudian dianalisa Sebelum dimulai pengujian saya jelaskan sekali lagi tujian memory dump adalah melihat

Kode program pada bagian memory di jalankan

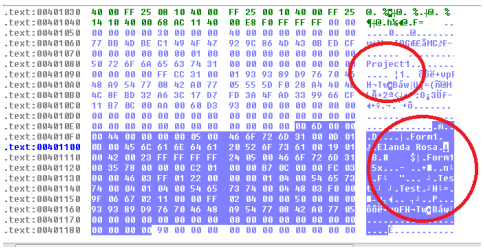


Gambar.14 *IDA view File telah di patch terdapat kode segment XdumpR*



***Gambar 15.** Hex Editor Informasi File tidak dapat di baca*

pada saat program ini dijalankan dia berjalan di addres awal : 0004056ED



Gambar 16. *Informasi masih bisa di akses sebelum file di Patch*

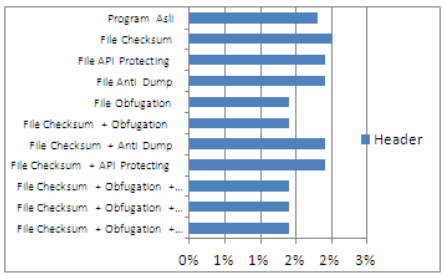
Pada program awal setelah di dump informasi pada memory masih bisa di akses sehingga dengan ini membuktikan bahwa program yang telah di patching sudah tidak memberikan informasi yang relevan **.**

Hasil pengujian dari analisa File

Dari hasil pengujian dan analisa File yang telah di lakukan sebelum nya maka di dapatkan data berupa jumlah perubahan kode program , panjang header pada file program, panjang badan program dan besar ruang kosong pada Program itu sendiri

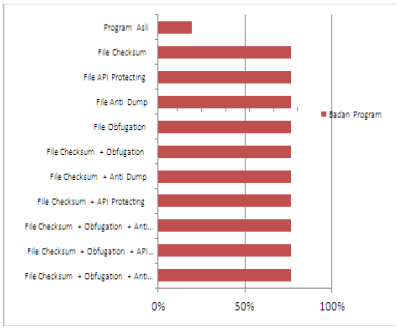
Header program

Grafik 1 *panjang header file pada tiap program*



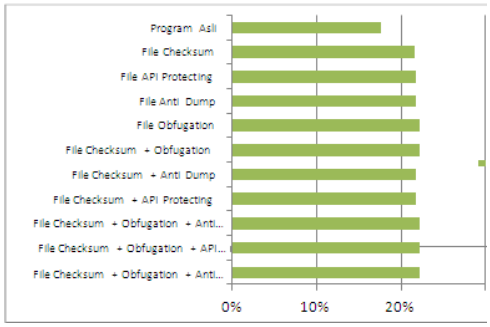
Badan Program

Grafik 2 *panjang badan program tiap file*



Ruang Kosong pada Program

Grafik 3 *panjang ruang kosong tiap file*



Grafik 4 *skala perbandingan yang terjadi dalam 10 bentuk hasil file yang berbeda*

Dari data hasil penelitian dapat di gambarkan secara keseluruhan dalam skala persentase perubahan yang terjadi dari 10 bentuk hasil file patching sehingga menghasilkan sebuah data yang digambarkan dalam bentuk sebuah grafik seperti di atas dan banyak Kode yang berubah tiap file tentunya tidak sama sehingga di dapatkan sebuah grafik perubahan kode.

Kesimpulan

Kesimpulan yang dapat di ambil dari penelitian tesis ini adalah sebagai berikut :

1. Untuk mengamankan sebuah aplikasi tidak harus menginstall sebuah antivirus kedalam sistem
2. Untuk mengamankan sebuah sistem, aplikasi yang telah di patching akan mematikan dirinya apabila aplikasi tersebut terinfeksi oleh virus agar tidak menjadi media penyebar dari virus
3. Apabila terjadi perubahan kode pada aplikasi yang telah di patch. Baik secara di sengaja ataupun tidak dengan sengaja aplikasi tersebut tidak akan bisa di jalankan
4. Dengan algoritma berbeda yang di tambahkan kedalam aplikasi tersebut pada aplikasi yang sama akan mengakibatkan besar dari file yang telah di patch berbeda-beda

Daftar Pustaka

[1] Alsa, Asmadi. (2004) *Pendekatan Kuantitatif Kualitatif dalam Penelitian Psikologi*. Yogyakarta: Pustaka Pelajar

Anharku, -checksumcrc32 dokumen pdf IlmuKomputer.Com - Desember 2013

[2] An Anti-Reverse Engineering Guide – -Guide/ Agustus 2014

[3] Chuan,lee,ling. ,Ismail,Mahamod. ,Jumari,kasmiran. ,Yee,Chan,Lee, (10 Maret,2013)

[4] *Architecture Of Malware Detector For Obfuscated Code Inspection* Journal of Theoretical and Applied Information Technology, Department of Electrical, Electronic and System Engineering Nation University of Malaysia, Malaysia

[5] Coliberg CS., Thomborson, Clark (2002), *Watermarking, Temper-Proofing and Obfuscation – Tools for Software Protection*, Journal IEEE Transc On Software Engineering Vol 28 No 6

[6] Executable and Linkable Format (ELF) - Januari 2014

[7] Eagle, Chris, (2011) *The Ida Pro Book, 2nd Edition* ,Copyright : No Starch Press, Inc.38 Ringold

Street, San Francisco, CA 94103

[8] Eilam, Eldad, (2005) *Reversing : Secret of Reverse Engineering*, Copyright Wiley Publishing,Inc 10475. Indianapolis IN 46256,

Canada.

[9] Ferguson, Justin., Kaminsky,Dan. Larsen,Jason. Miras,Luis. Pearce,Walter, (2008) *Reverse Engineering Code with IDA Pro*. Copyright : by Elsevier, Inc published by : Syngress

[10] Publishing, Inc. Elsevier, Inc. 30 Corporate Drive Burlington, MA 01803

[11] Hadi, Sutrisno. (1985) *Metodologi Research Jilid 4* Yogyakarta: Yayasan Penerbit Fakultas Psikologi UGM.

[12] Latipun. (2002) *Psikologi Eksperimen*. Malang: UMM Press

[13] Mironov, Ilya November 14, (2005) : *Hash functions: Theory, attacks, and applications.*

[14] Nylander, Erik ,(June 24, 2014) *Improved code obfuscation through automatic construction of hidden execution paths,* Thesis , Department of Electrical and Information Technology Lund University, Sweden