Jurnal Sistem Informasi dan Teknik Informatika
STMIK Dharma Wacana
VOL : 1 NO: 1
ISSN : 2549-8436 Email :lp3m@stmikdharmawacana.ac.id Website :http://ojs.stmikdharmawacana.ac.id/ E-ISSN :

PENGUJIAN METODE CHECKSUM DAN ANTI DUMP SOFTWARE DENGAN TEKNIK REVERSE ENGINERING

Oleh

M. Reza Redo I[1], Septian D Chandra[2]
Dosen STMIK Dharma Wacana Metro Lampung[12]
Email: lp3m@stmikdharmawacana.ac.id website: http://ojs.stmikdharmawacana.ac.id

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 Checsum Obfuscation/ Obfuscated Code dan Anti Dumping dimana dari hasil patching tersebut akan di uji kualitas dan diukur Panjang Kode , Output File Hasil, dan Banyaknya perubahan yang terjadi 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 oleh tahapan 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 penulis bermaksud menerapkan teknik *reverse engineering* guna melakukan pengujian dengan cara melakukan patching pada perangkat lunak (*software*) terutama pada sisi keamanan (*security*).

2.Metode Penelitian

2.1 Penelitian Experiment

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 di berikan kepada beberapa aplikasi perangkat lunak yang akan di jadikan bahan untuk experiment

Adapun karakteristik dari penelitian experimen meliputi :
a.Manipulasi Variabel
b.Kontrol Pada Subjek
c.Perlakuan subjek (Treatment)

2.2 Desain Penelitian (One Group Pretest-Posttest Eksperimen)

Table 2.1 (One Group Pretest-Posttest Eksperimen)

O2	X	O2
Pretest Object	Threatment	Posttest Object

2.3 Variabel Penelitian

Table 2.2 (Variabel)

Nama	Keterangan
A	Variabel awal dimana File yang akan di uji coba belum mendapatkan perlakukan khusus dari Experimen yang akan di jalankan
B	Variabel Threatmen dimana pada variabel ini adalah sebuah perlakukan khusus yang akan di tambahkan pada variabel A
C	Variabel Hasil yang mengandung hubungan antara Variabel A dan B

2.3.1 Identifikasi Variabel

2.3.1.1 Variabel A

Adalah kondisi dimana Sebuah program Berjalan Normal dimana perlakukan pada aplikasi tersebut belum mendapatkan perlakuan

Table 2.3 (Identifikasi variabel A)

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

2.3.1.2 Variabel B

Variabel Threatmen/perlakukan dimana pada variabel ini adalah sebuah perlakukan khusus yang akan di tambahkan pada variabel A, pada perlakuan yang akan di terapkan pada variabel A dapat di tentukan sebagai berikut:

Table 2.4 (Identifikasi variabel B)

Variabel	Threatment/Perlakuan
B_1	Checksum
B_2	Anti Dump

2.3.1.3 Variabel C

Variabel Hasil yang mengandung hubungan antara Variabel A dan B pada variabel C diterapkan perlakukan atau threatmen yang dapat di lihat pada tabel

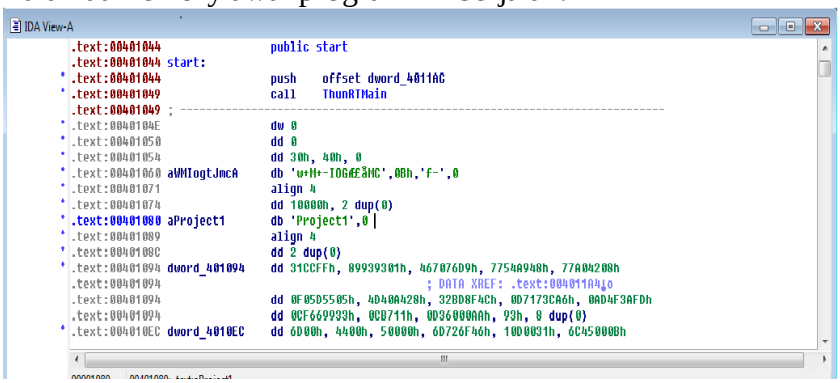
Table 2.5 (Identifikasi variabel C)

Variabel	Checksum	Anti Dumping
C_1	√	
C_2		√
C_3	√	√

3.Pembahasan

3.1 Pengujian Terhadap File Hasil

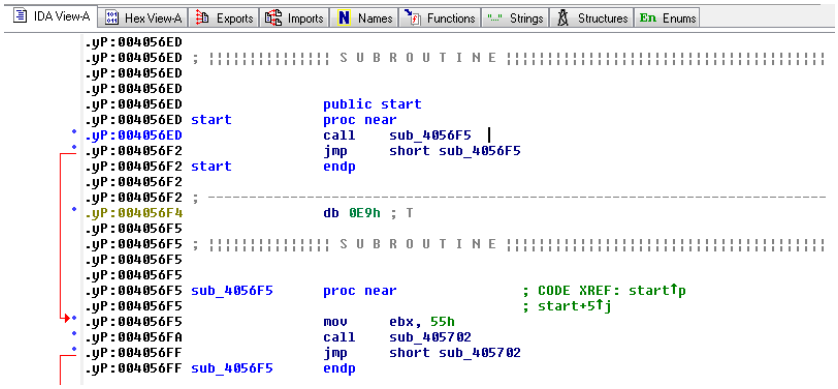
Tujuan melakukan pengujian ini adalah membuktikan bahwa File yang telah di Patch telah terpatching sesuai dengan ketentuan yang telah di tetapkan. Pada Baris Algoritma Assembly program dummy (variable A) yang akan di patching program berjalan di offset : *00401044* dan merupakan alamat memory awal program ini berjalan.



Gambar 3.1 Start Program Berjalan

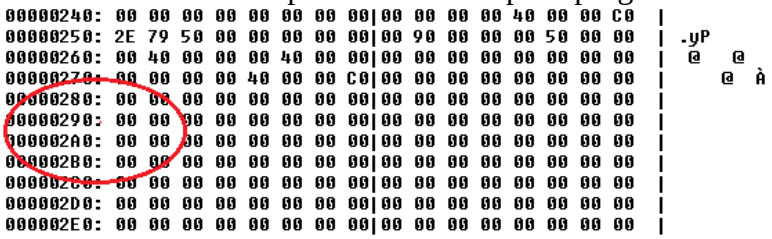
3.1.1 Pengujian Variabel C_1

Pada File yang telah Di Checksum (variable C_1) terdapat perubahan dimana offset start program berjalan di alamat memory 004056ED yang merupakan alamat memori relatif



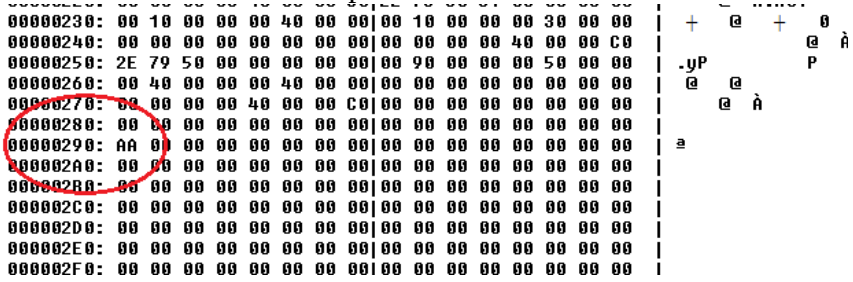
Gambar 3.2 Start Program Berjalan pada variabel C_1

Pada tahapan ini 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 untuk melakukan perubahan Kode pada program tersebut



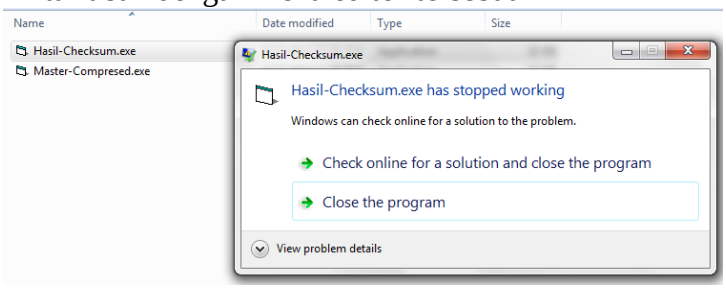
Gambar 3.3 Hexa Program setelah di buka

kita bisa saja merubah ke offset mana saja tapi tidak kita lakukan karena akan mengakibatkan perubahan data jadi 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



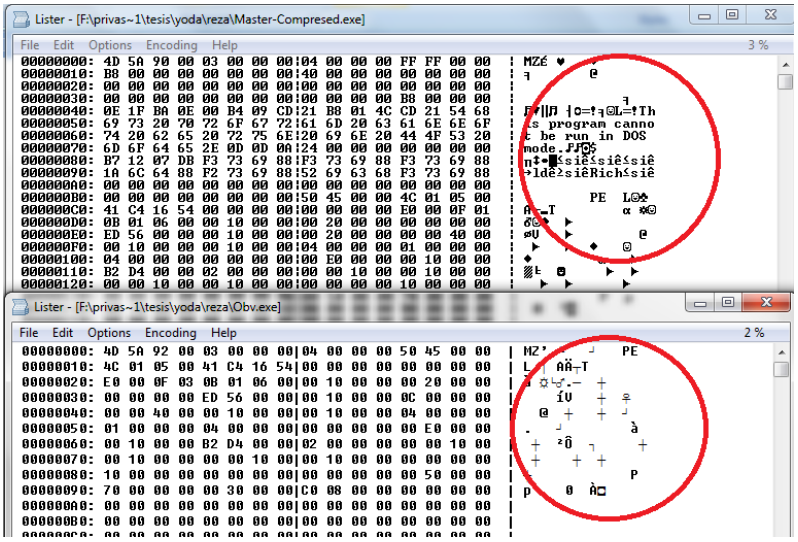
Gambar 3.3 Hexa Program (nilai AA pada offset ; 00000290)

Tujuan perubahan ini adalah untuk membuktikan apakah Program masih berjalan apabila terjadi Perubahan Kode yang tidak di inginkan , hal ini di maksudkan untuk menghindari terjadi nya serangan Virus atau proses Cracking sebuah software. Jika sudah maka jalankan Program yang telah kita rubah dengan Hexa editor tersebut



Gambar 4.8 Error reporting Windows7

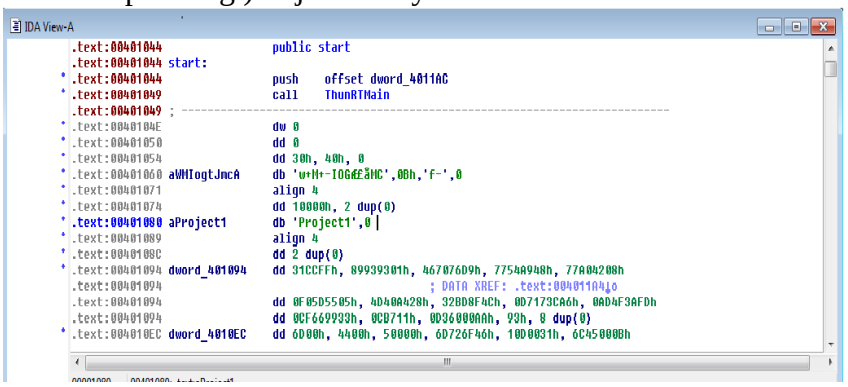
setelah itu yang kita lakukan adalah menguji kode tersebut dengan cara melihat informasi dengan mengunakan hex editor apakah obvogasi kode telah berhasil menghilangkan informasi file tersebut



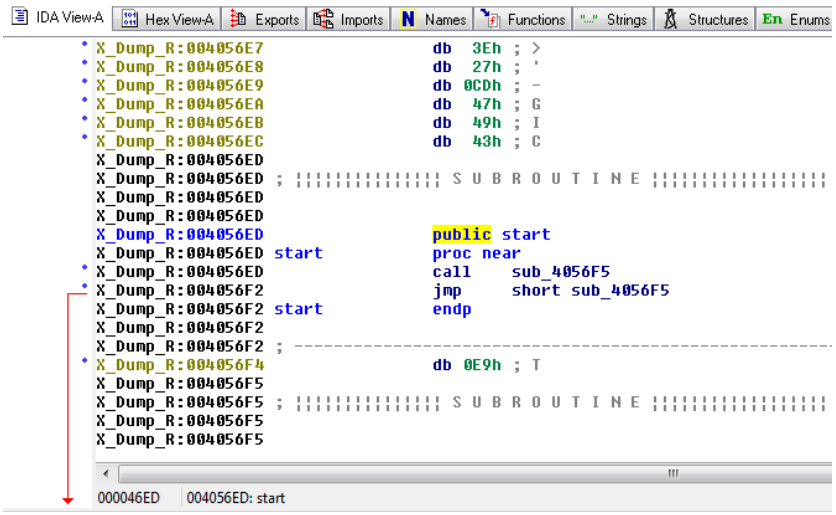
Gambar 4.9 Hex editor File Komparasi

3.1.2 Pengujian Variabel C_2

Pada saat kita telah melakukan patching anti dump terhadap software untuk menguji apakah software tersebut telah berhasil di patch pertama tama kita lakukan uji fisik sebelum dan sesudah software itu di patch. Uji fisik ini dimaksudkan untuk mengetahui apakah Kode yang sudah di patch sdah ter patch kedalam Kode source/sumber di program itu sendiri untuk itu kita lakukan prosedur standar dengan melihat isi kode dengan bantuan aplikasi debugging.
Sama seperti pada saat pengujian checksum yaitu mendokumentasi dari offset dimana file asli(File yang belum di patching) di jalankan yaitu di offset : 00401044

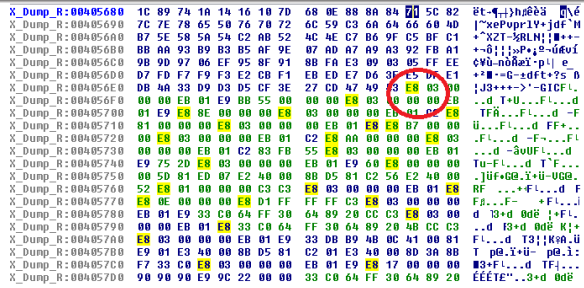


Gambar 4.10 IDA view File yang akan di beri Anti Dump



Gambar 4.11 IDA view File telah di patch terdapat kode segment XdumpR

Gambar di atas menunjukan bahwa setelah di patching program berjalan di offset : 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 di analisa Sebelum dimulai pengujian saya jelaskan sekali lagi tujian memory dump adalah melihat Kode program pada bagian memory di jalankan . Hal ini biasanya di gunakan para cracker atau hacker untuk mengetahui struktur atau pola sebuah program, atau bahkan mencuri data yang terkadung pada memory yang sedang berjalan, dengan kata lain untuk mengetahui bagaimana program itu berjalan biarkan dia menulis kesalahan nya sendiri pada file dump sehingga cracker atau hacker program dapat tau dimana kelemahan program ini.



Gambar 4.12 Hex Editor Informasi File tidak dapat di baca

4.Kesimpulan

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

1. 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
2. Dengan patching kita dapat menyembunyikan informasi file, hal ini biasa digunakan untuk menyembunyikan informasi tentang serial number untuk pencegahan hacking software
3. Aplikasi yang telah di patching akan mematikan dirinya apabila aplikasi tersebut terinfeksi oleh virus agar tidak menjadi media penyebar dari virus

Daftar Pustaka

Arora, Rohit, Anishka Singh, Himanshu Pareek, and Usha Rani Edara. 2013. "A Heuristics-Based Static Analysis Approach for Detecting Packed PE Binaries." *International Journal of Security and Its Applications* 7 (5): 257–68. doi:10.14257/ijsia.2013.7.5.24.

Chuan, Lee Ling, Mahamod Ismail, Kasmiran Jumari, and Chan Lee Yee. 2013. "Architecture of Malware Detector for Obfuscated Code Inspection." *Journal of Theoretical and Applied Information Technology* 49 (1): 59–69.

Collberg, Christian S, Clark Thomborson, and Senior Member. 2002. "ObfuscationÐTools for Software Protection." *Computer* 28 (8): 735–46. doi:10.1109/TSE.2002.1027797.

Faculty, The Academic, Monirul I Sharif, and In Partial Fulfillment. 2010. "Robust and Efficient Malware Analysis and Host-Based Monitoring Robust and Efficient Malware Analysis and Host-Based Monitoring," no. December. doi:862084502.

Latuconsina, Roswan. 2006. "Technical Report FUNGSI HASH : TIGER Sebuah Kajian Spesifikasi Dan Keamanan."

Mironov, Ilya. 2005. "Hash Functions : Theory, Attacks, and Applications Theory of Hash Functions." *Microsoft Research, Silicon Valley Campus*, 1–22. http://research.microsoft.com/pubs/64588/hash_survey.pdf.

Richey, Rodger. 1998. "Calculating Program Memory Checksums Using a PIC16F87X," 1–6.

Ross, Steven M, Gary R Morrison, Robert D Hannafin, Michael Young, Rita C Richey, James D Klein, Jan van den Akker, and Wilmad Kuiper. 2008. "Research Designs." *Handbook of Research on Educational Communications and Technology, 3rd Ed.*, 751–61.

Setiawan, Okie, Rina Fiati, and Tri Listyorini. 2014. "Algoritma Enkripsi Rc4 Sebagai Metode Obfuscation Source Code Php." *Prosiding SINATIF* 1: 113–20.

Spraao8, and Spraao8. 2009. "Common Object File Format." *Application Report*, no. April: 1–15.