JETT
Jurnal Elektro Telekomunikasi Terapan

ANALISIS DAN DETEKSI MALWARE POISON IVY DENGAN METODE MALWARE ANALISIS DINAMIS DAN MALWARE ANALISIS STATIS

ANALYSIS AND DETECTION OF MALWARE POISON IVY WITH MALWARE DYNAMIC ANALYSIS METHOD AND MALWARE STATIC ANALYSIS

Rizky Taufiq Amdani[1], Hafidudin, S.T., M.T.[2], Muhammad Iqbal, S.T., M.T.[3]

[1,2,3]Program Studi D3 Teknologi Telekomunikasi, Fakultas Ilmu Terapan, Universitas Telkom
[1]rizkytaufiq@student.telkomuniversity.ac.id, [2] hafidudin@telkomuniversity.ac.id, [3]miqbal@telkomuniversity.ac.id

Abstrak
Malware adalah sebuah perangkat lunak atau yang diciptakan untuk melakukan penyusupan atau melakukan perusakan pada sistem komputer. Untuk penyebaran malware saat ini begitu mudah baik melalui flashdisk, iklan-iklan tertentu pada website, dan media lainnya. Semuanya sangat erat kaitannya dengan tindak kejahatan seperti pencurian file, kartu kredit, internet banking dan lain sebagainya. Salah satu malware yang dapat mengendalikan komputer pengguna secara diam-diam dan dari jarak yang jauh adalah malware poison ivy, dikenal sebagai "Remote Acces Trojan" karena dapat memberikan kontrol penuh melalui pintu belakang (backdoor).
Pada proyek tingkat ini malware Poison Ivy akan di analisa untuk mengetahui cara kerja dari malware Poison ivy tersebut, Analisa dilakukan dengan metode analisis statis dan dinamis dimana kombinasi dari metoda tersebut merupakan kombinasi yang sesuai untuk menganalisa cara kerja dari sebuah malware.
Berdasarkan Analisa ini Malware Poison Ivy bekerja dengan cara menginject file dll yaitu loadlibrary.dll untuk mengontrol system pada computer target lalu penambahan file dan registry pada komputer target dan saat server melakukan koneksi dengan komputer target lalu sebelum melakukan koneksi malware poison ivy melakukan proses pencocokan password dan jika password cocok maka komputer server dapat mengakses komputer target. Dan malware poison ivy tidak memakan terlalu banyak resource pada sebuah system computer.
Kata kunci : Poison Ivy, Remote Access Trojan(RAT), Malware

Abstract
Malware is a software or created to infiltrate or destroy computer systems. For the current spread of malware is so easy both through flash, certain advertisements on the website, and other media. Everything is very closely related to crimes such as file theft, credit cards, internet banking and so on. One malware that can control a user's computer secretly and remotely is poison ivy malware, known as "Remote Access Trojan" because it can give full control through the backdoor.
In this level of malware poison ivy project will be analyzed to find out how the poison ivy malware works, analysis is done by static and dynamic analysis method where the combination of methods is the appropriate combination to analyze the workings of a malware.
Based on this analysis Poison Ivy Malware works by injecting files etc. namely loadlibrary.dll to control the system on the target computer and then adding files and registry on the target computer and when the server connections with the target computer and then before connecting poison ivy malware performs a password matching process and if the password matches then the server computer can access the target computer. And poison ivy malware doesn't take up too much resources on a computer system.
Keywords: Poison Ivy, Remote Access Trojan(RAT), Malware

1. PENDAHULUAN

Dalam era teknologi yang semakin berkembang pesat saat ini, komputer digunakan untuk memudahkan pekerjaan manusia, dalam pengoperasiannya ada *software* yang berjalan diatas sistem operasi, dan sangat berperan penting dalam melakukan tugas-tugas yang dikerjakan oleh pengguna. Karena melalui *software* inilah suatu komputer dapat menjalankan perintah sehingga membantu pengguna dalam menyelesaikan pekerjaannya. Namun tidak semua *software* dapat membantu dan memudahkan manusia dalam melakukan pekerjaannya, ada pula jenis *software* yang diciptakan untuk melakukan perusakan atau tindak kejahatan yang dapat merugikan orang lain, *software* tersebut dikategorikan sebagai *Malicous Software*. Salah satu media yang digunakan oleh *intruder* untuk mengendalikan komputer pengguna secara diam-diam dari jarak jauh adalah *malware poison ivy*, dikenal sebagai "*trojan access remote*" karena dapat memberikan kontrol penuh kepada *intruder* melalui pintu belakang (*backdoor*). Kemampuan *malware poison ivy* mengadopsi dari *software Remote Access Trojan* (RAT). Pada Penelitian [1] Cara kerja *malware Flawed Ammy RAT* ini tidak dapat berjalan ketika korban sedang keadaan mode DOS. *Malware* melakukan infeksi pada system, membaca dan menyimpan data yang ada pada komputer. Pada penelitian[2] Hasil yang diperoleh yaitu mengetahui cara kerja program tersebut pada sistem komputer, kombinasi metode untuk menganalisa cara kerja *malware* (*poison ivy*) dengan beberapa *signature, filename* dan *string* sehingga dapat melakukan proses *login* secara *remote* tanpa diketahui oleh pemilik komputer.

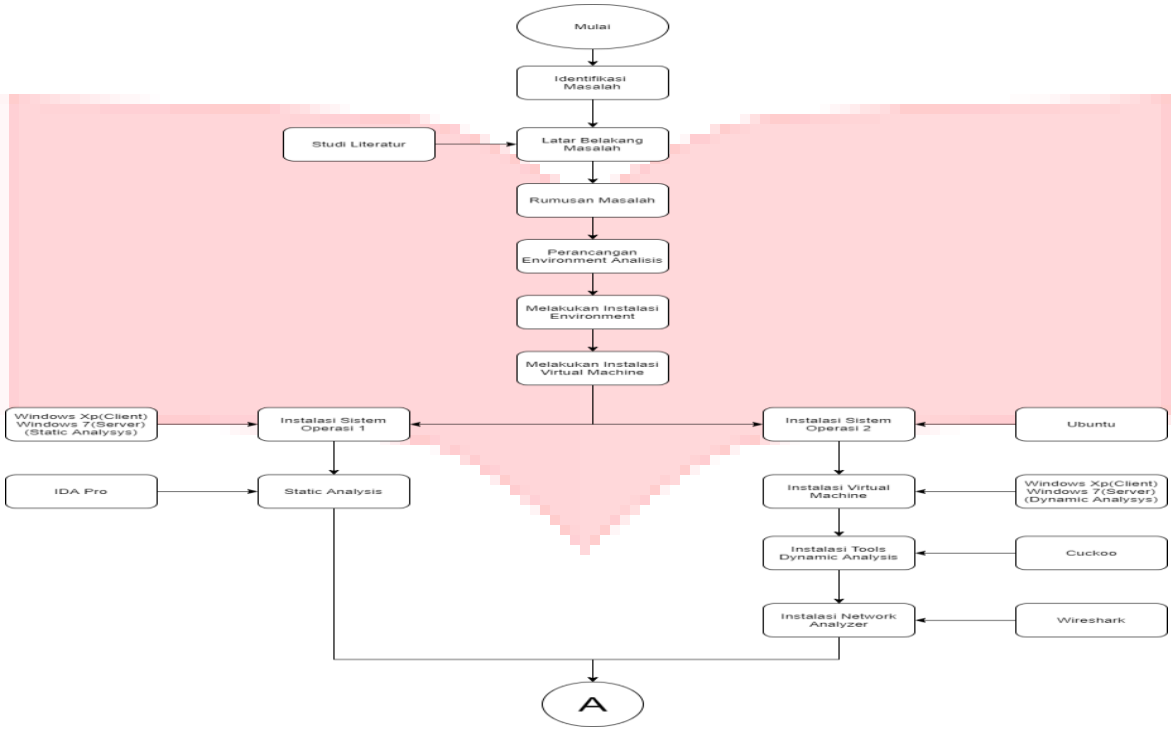
Oleh karena itu pada penelitian ini dilakukan untuk mengetahui bagaimana proses koneksi antara client dengan server dan bagaimana server malware posison ivy berkomunikasi dengan client, serta fungsi-fungsi dari system opersi yang digunakan oleh malware poison ivy. Pada penelitian ini juga digunakan aplikasi yaitu memory forensik untuk menganalisa memory dari komputer yang terkena Malware posion ivy.

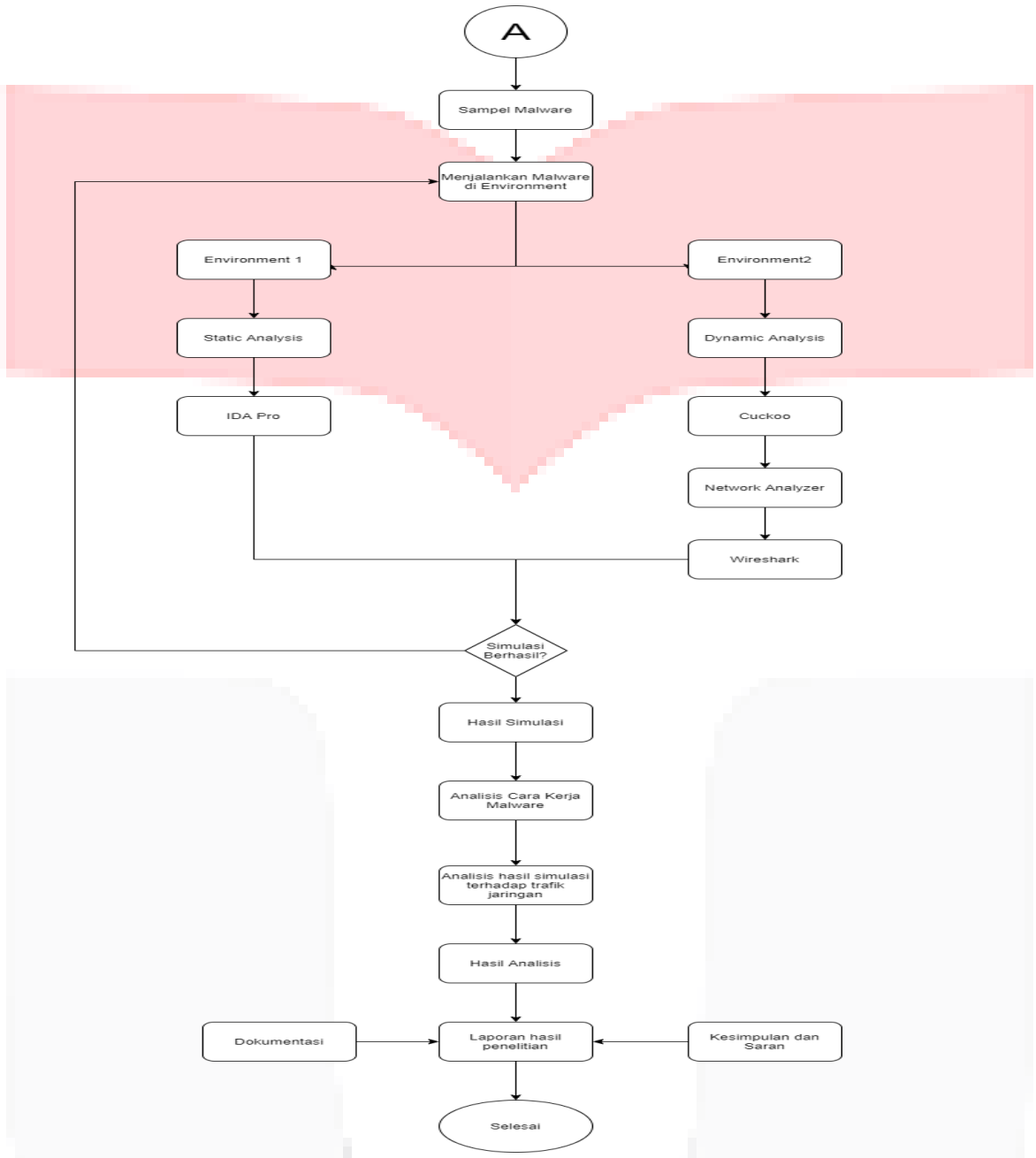
2. DASAR TEORI & PERANCANGAN SISTEM

2.2 Model Sistem

Model Sistem dilakukan dengan menggunakan Virtual Box dan menggunakan system operasi windows XP sebagai client dan windows 7 sebagai server. Setelah system operasi berhasil diinstall lalu pada windows 7 akan dibuat server dari *poison ivy*, lalu mengenerate program yang akan dijalankan di Windows XP (client). Analisis dinamis akan dilakukan dengan aplikasi Cuckoo dan analisis statis akan dilakukan menggunakan aplikasi IDA pro.

JETT
Jurnal Elektro Telekomunikasi Terapan





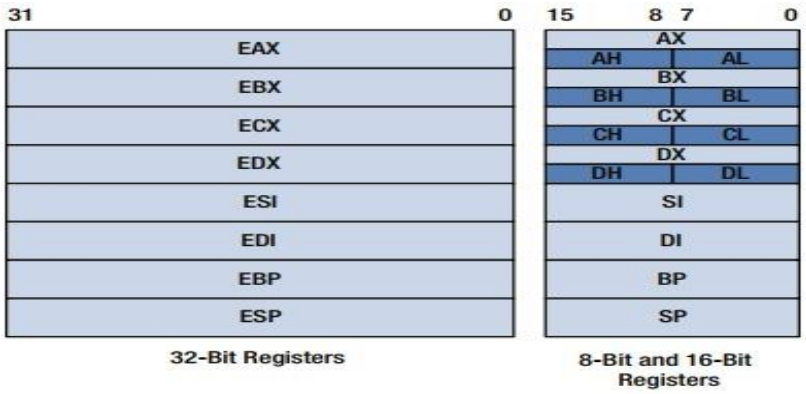
Gambar 2. Flowchart model sistem

3. ANALISIS SIMULASI PERENCANAAN

3.1 Mempelajari Teknik Reverse-Engineering

Pada bab ini untuk bisa melakukan Reverse-Engineering adalah dengan memahami *Binary Code* atau biasa disebut bahasa Assembly / Instructions Set Architecture, bahasa Assembly adalah sebuah bahasa *native* langsung yang berada dalam masing-masing *processor*.

Processor memiliki arsitekur, arsitektur adalah sebuah kapasitas *mnemonic* yang biasanya bisa ditampung oleh data. Arsitektur *processor* yang biasanya banyak dipakai kebanyakan orang adalah arsitektur x86 atau x86_64. Perbedaan tersebut sebagai contoh adalah pada *register*.



Gambar 3. x86-32 general-purpose register

Tabel 1. fungsi dari 3 x86-32 general-purpose register

Register	Conventional Use
EAX	Accumulator
EBX	Memory pointer, base register
ECX	Loop control, Counter
EDX	Interger Multiplication, integer division
ESI	String instruction source pointer, index register
EDI	String instruction destination pointer, index register
ESP	Stack Pointer
EBP	Stack Frame Base Pointer

3.4 Analisis Statis

3.4.1 Segment Program Poison Ivy

Name	Start	End	R	W	X	D	L	Align	Base	Type	Class	AD	es	ss	ds	fs	gs
HEADER	00400000	00400200	?	?	?	.	L	page	0002	public	DATA	32	FFFFF...	FFFFF...	0003	FFFFF...	FFFFF...
.idata	00400200	00400208	R	.	X	.	L	para	0001	public	CODE	32	0000	0000	0003	FFFFF...	FFFFF...
.text	00400208	00400400	R	.	X	.	L	para	0001	public	CODE	32	0000	0000	0003	FFFFF...	FFFFF...
.data	00400400	00401800	R	W	.	.	L	para	0003	public	DATA	32	0000	0000	0003	FFFFF...	FFFFF...
GAP	00401800	00402000	R	W	.	.	L	byte	0000	private	DATA	32	0000	0000	0003	FFFFF...	FFFFF...

Gambar 4. Segment Program Poison Ivy

Pada gambar 4, merupakan struktur dari program *malware Poison ivy*, dimana program tersebut memiliki 5 segment yaitu HEADER, .idata, .text, .data, GAP. Penjelasan segment program *poison ivy* :

1. HEADER

   Merupakan segment yang mengandung informasi awal terkait program

2. idata

   Mengandung informasi terkait file .dll yang dibutuhkan program.

3. .text

   Segmen teks mengandung instruksi-instruksi yang akan di eksekusi oleh komputer.

4. .data

   Segmen data adalah bagian dari sebuah file objek atau alamat yang sesuai dari program yang berisi variabel statis variabel global dan variabel lokal. Ukuran pada segmen ini ditentukan oleh ukuran nilai-nilai dalam kode sumber program, dan tidak berubah pada saat dijalankan.

5. GAP

Segmen yang mengandung informasi Permissions pada program.

3.4.2 Proses injeksi .dll

```
.data:00401228                 call    near ptr loc_401235+1
.data:0040122D                 popa
.data:0040122E                 db      64h
.data:0040122E                 jbe     short loc_401292
.data:00401231                 jo      short loc_40129C
.data:00401233                 xor     esi, [edx]
.data:00401235
.data:00401235 loc_401235:                             ; CODE XREF: sub_400400+E28↑p
.data:00401235                 add     bh, bh
.data:00401237                 xchg    eax, ebp
.data:00401238                 and     ecx, esi
```

Gambar 5. Proses Injeksi

Seperti yang kita lihat pada source kode program yang ada pada gambar 5 , program memanggil sebuah fungsi yang berada di alamat loc_401235+1 , dimana pada source code di atas fungsi untuk loc_401235+1 itu tidak terlihat , yang ada adalah fungsi loc_401235. Jika kita teliti lebih cermat byte dari funsgi loc_401235 tersebut adalah

0x00 0xff 0x95 0x21 0xf1 0xff 0xff

ADD BH,BH
XCHG EAX,EBP
AND ECX,ESI

Jika kita memanggil loc_401235+1, perintah di atas akan berubah total menjadi perintah untuk memanggil sebuah fungsi yang ada di alamat ebp-0xedf dimana alamat tersebut mengarah ke fungsi LoadLibrary dari kernel32.dll.

```
loc_401236:
call    [ebp+var_EDF]
; END OF FUNCTION CHUNK FOR sub_400400
```

Gambar 6. Isi perintah di loc_401236

3.4.3 Proses Koneksi

```
loc_400697:
                mov     [ebp-1A4h], eax
                lea     eax, [ebp-1A8h]
                push    eax
                pop     dword ptr [esi+125h]
                push    10h
                lea     eax, [ebp-1A8h]
                push    eax
                push    dword ptr [ebp-5]
                call    dword ptr [esi+5] ; sent ping to server
                or      eax, eax
                jnz     loc_40095C
```

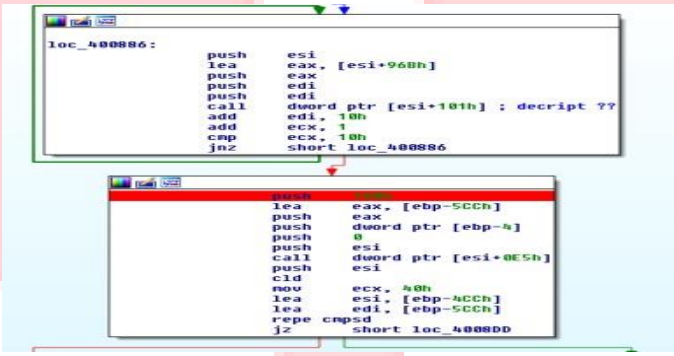
```
                mov     dword ptr [ebp-5], 0FFFFFFFFh
                cmp     byte ptr [esi+0AEFh], 0 ; cek if server respon
                jle     loc_400851
```

Gambar 7. Thread proses koneksi

Pada source code diatas program akan mengecek nilai dari **ZF,** jika nilai dari ZF 0 maka program akan melompat ke exit program, dan jika nilai ZF tidak sama dengan 0 maka program akan melanjutkan mengeksekusi intruksi untuk mengecek apakah server merespon atau tidak. Jika server tidak merespon maka program akan langsung

menjalankan exit program, jika server merespon program akan mengeksekusi instruksi selanjutnya.

3.4.4 Proses Cek Password

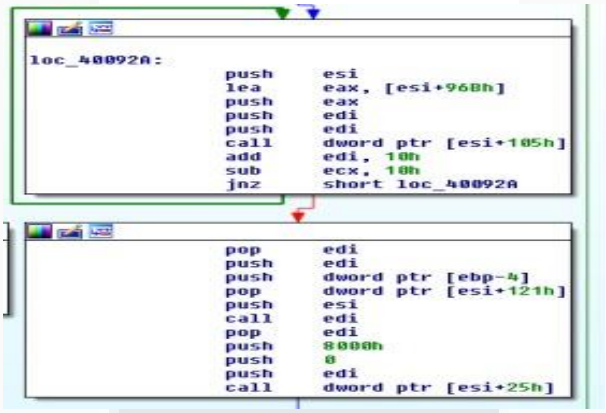


Gambar 8. Thread proses cek password

Pada blok pertama yang ada pada gambar 8 merupakan sebuah code program untuk melakukan proses *looping*, dimana proses tersebut bertujuan untuk mengencrypt sebuah data yang nantinya data tersebut akan dicocokan dengan data dari server.

Pada blok kedua source code diatas program akan membandingkan nilai dari server dengan nilai yang ada di client. Dimana edi menyimpan nilai yang dikirimkan oleh server, jika hasil dari perbandingan tersebut tidak sama maka program akan langsung exit, jika perbandingannya bernilai sama maka intruksi selanjutnya akan dieksekusi.

3.4.5 Proses control Client



Gambar 9. Thread proses control client

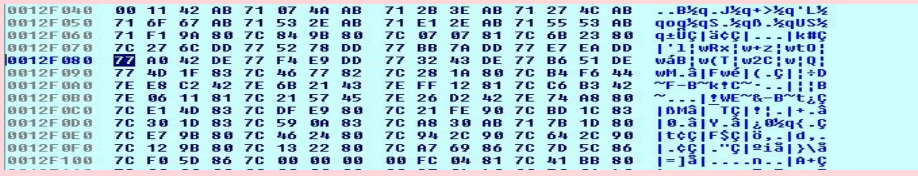
Pada blok pertama pada program diatas melakukan sebuah proses perulangan dimana perulangan tersebut berfungsi untuk memindahkan sebuah data dari satu lokasi kelokasi lain. Data yang dipindahkan tersebut merupakan sebuah intruksi baru, dimana intruksi tersebut akan membuat server dapat mengontrol client.

Pada blok kedua, setelah selasai memindahkan data-data maka program akan memanggil data yang dipindahkan tadi, lalu address dari data tersebut disimpan ke register edi, lalu program

JETT
Jurnal Elektro Telekomunikasi Terapan

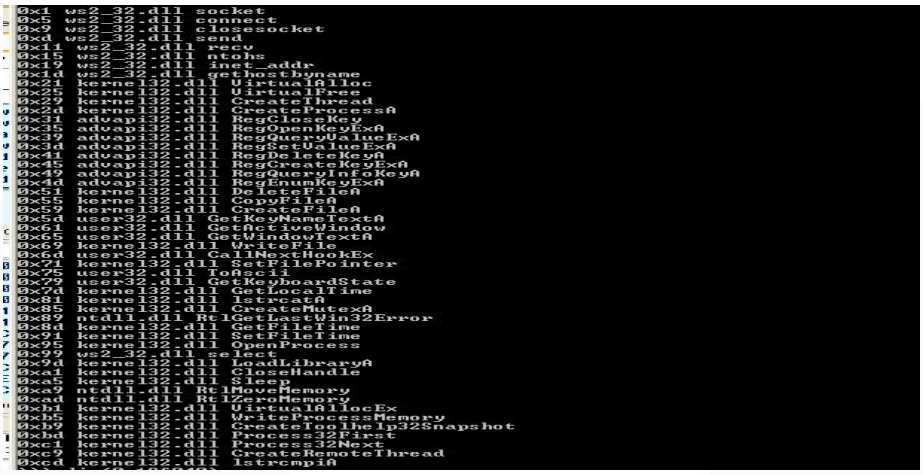
mengeksekusi perintah call edi. Setelah perintah call edi dieksekusi maka server dapat mengontrol client

3.4.6 File .dll serta fungsi yang digunakan Poison Ivy



Gambar 10. Array pada IDAPro yang menyimpan alamat dari fungsi .dll

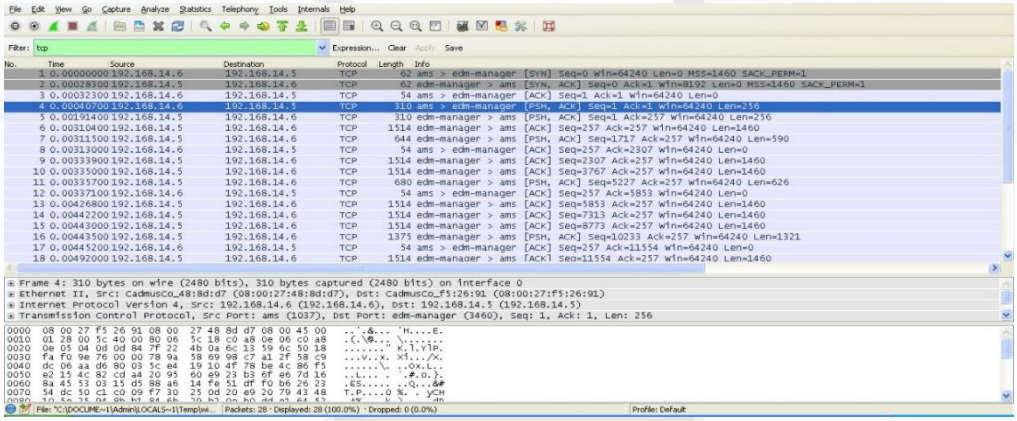
Gambar 10 merupakan fungsi yang digunakan oleh *Poison ivy*, dimana jika dilihat melalui IDA pro fungsi dibawah tidak akan terlihat, fungsi dibawah didapat dengan cara melakukan analisis memory pada system operasi client menggunakan Volatility. Penyebab IDApro tidak dapat menganalisa fungsi-fungsi dibawah karena program *poison ivy* menyimpan alamat-alamat dari fungsi tersebut kedalam sebuah *array* seperti yang terlihat pada gambar 11 :



Gambar 11. Fungsi .dll yang dipakai poison ivy

3.5 Analisis Dinamik

3.5.1 Wireshark

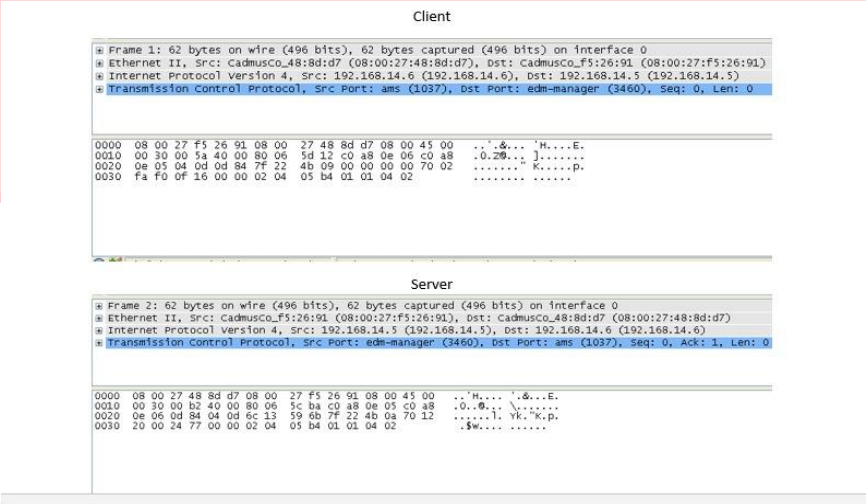


Gambar 12. Raw paket antara server dan client

Pada gambar 12 merupakan capture dari paket yang terjadi antara client dan server *poison ivy*. Disini *poison ivy* server dan client melakukan komunikasi dengan menggunakan protocol *TCP (Transmision Control Protocol).*

Port yang digunakan oleh server adalah port 3460 dan client menggunakan port 1037:

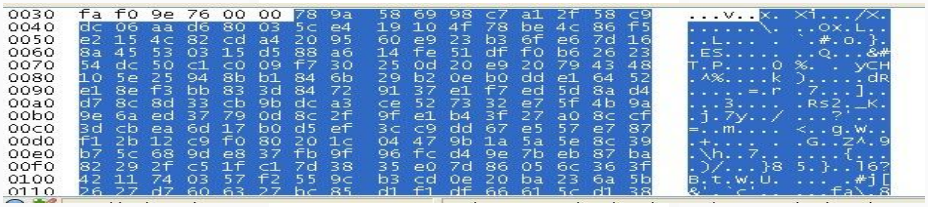
1. Thread pertama



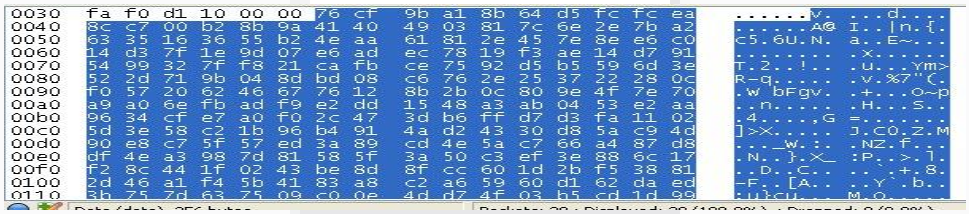
Gambar 13. Thread pertama

Pada thread ini client mencoba melakukan komunikasi ke server untuk meencek apakah server aktif atau tidak aktif, jika tidak aktif maka program dari client akan melakukan looping sampai server merespon atau program dihentikan secara paksa.

2. Thread kedua



Gambar 14. Paket yang dikirim client

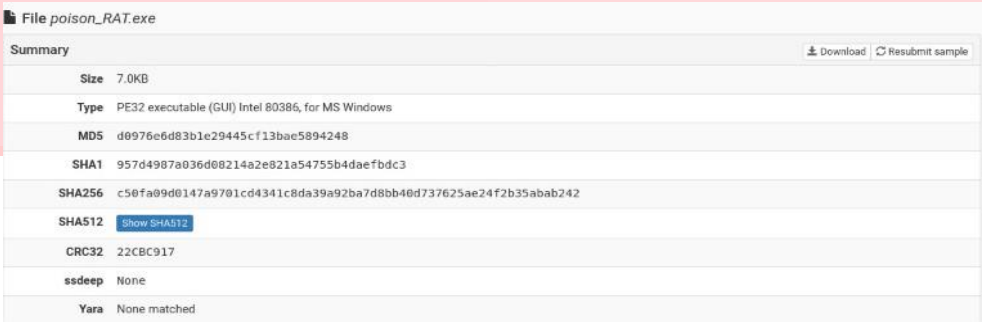


Gambar 15. Paket yang dikirim server

Pada thread ini setelah client mengirimkan paket untuk mengecek apakah server aktif atau tidak dan server merespon selanjutnya client akan mengirimkan sebuah data sebesar 256 byte ke server. Di server data yang dikirimkan oleh client tadi akan dienkripsi menggunakan algoritma yang sudah ada di program, lalu hasil dari enkripsi tersebut akan dikirimkan Kembali ke client. Pada sisi client data yang sudah di enkripsi tersebut akan dicocokan dengan data yang ada pada client.

Proses ini merupakan validasi password antara client dengan server. Apabila hasil enkripsi dari server berbeda dengan hasil enkripsi dari client maka komunikasi tidak akan berlanjut dan program akan melakukan looping terus menerus sampai program dihentikan. Dan apabila hasil enkripsi server dan client sama maka client dan server terhubung lalu server dapat mengakses client.

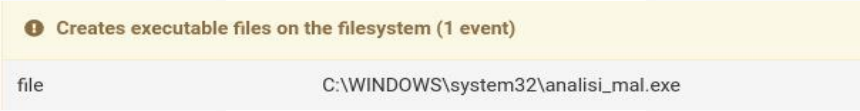
3.5.2 Analisis Dinamis menggunakan CUCKOO



Gambar 16. Informasi file poison_RAT.exe

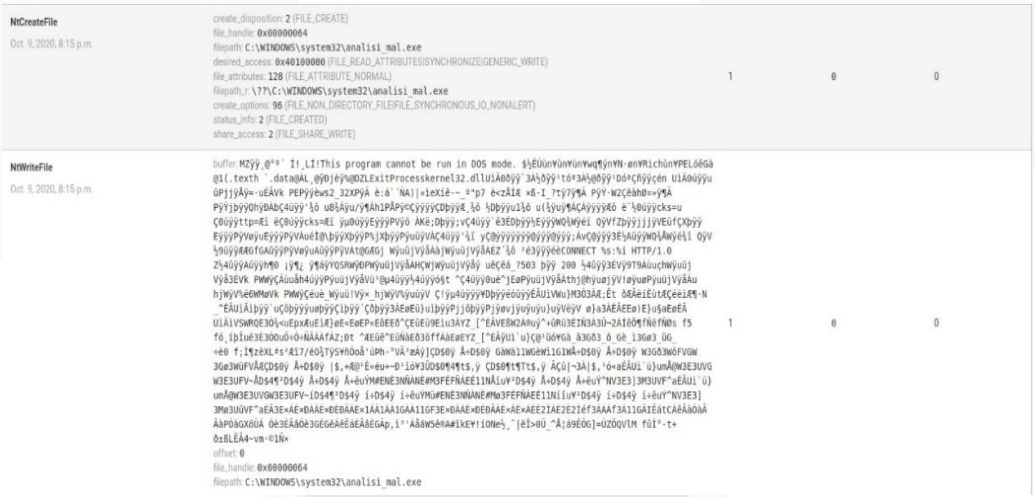
Gambar 16 merupakan hasil dari analisis menggukan mesin cuckoo sandbox, dapat dilihat pada gambar 16 terdapat informasi-informasi dari program Poison_RAT.exe. Saat file Poison_RAT.exe berjalan di mesin cuckoo terdapat perilaku-perilaku yang terdeteksi oleh mesin cuckoo, perilaku-perilaku tersebut antara lain:

1. Membuat file baru pada file system



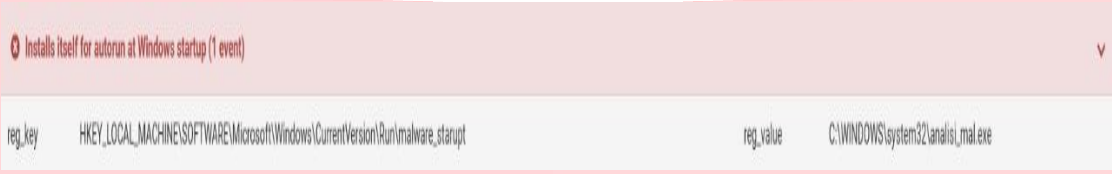
Gambar 17. Lokasi file baru yang dibuat

Mesin cuckoo mendeteksi sebuah perilaku dari *malware* poison_RAT.exe dimana *malware* ini akan membuat file baru bernama “analisi_mal.exe” pada folder “C:\WINDOWS\system32\”. Poison_RAT.exe meggunkan fungsi NtCreateFile dari file dll winternl.h untuk membuat file tersebut, lalu menggunakan fungsi NtWriteFile untuk menulis program analisi_mal.exe.



Gambar 18. Upaya pembuatan file oleh poison_RAT.exe

2. Menulis ke File registry windows



Gambar 19. File registry

Pada proses ini *malware* mencoba membuat registry baru pada windows , dimana regisitry ini akan membuat *malware* berjalan secara otomatis saat mesin dijalankan .registry yang dibuat ada pada:

HKEY_LOCAL_MACHINE*SOFTWARE*\Microsoft\Windows\CurrentVersion\Run.

Path ini merupakan path dimana semua program yang ada pada windows akan di jalankan saat mesin menyala. Pada proses ini *malware* akan menambahkan value baru dimana value tersebut mengarah ke file yang telah di buat *malware* yang ada pada filesystem dengan nama file analisi_mal.exe.

Gambar 20. Fungsi yang digunakan untuk membuat registry

Dengan menggunakan fungsi RegOpenKeyExA *malware* akan membuka registry, lalu dengan menggunakan RegSetValueExA *malware* akan menuliskan value ke registry tersebut.

3. Pembuatan Mutual exclusive (Mutex)

Gambar 21. File mutex

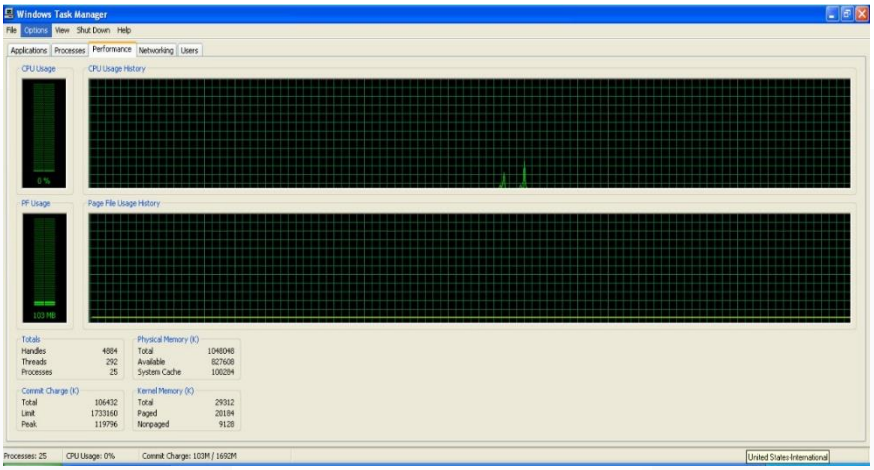
Pada proses mesin cuckoo mendeteksi bahwa *malware* membuat sebuah mutex dimana mesin cuckoo sudah mengetahui bahwa mutex ini merupakan mutex yang biasa digunakan pada *malware Poison ivy*.

Gambar 22. Fungsi yang digunakan dalam membuat mutex

Pada gambar 22 kita dapat melihat bahwa *malware* menggunakan fungsi NtCreateMutant dari NTDLL library untuk membuat mutex.

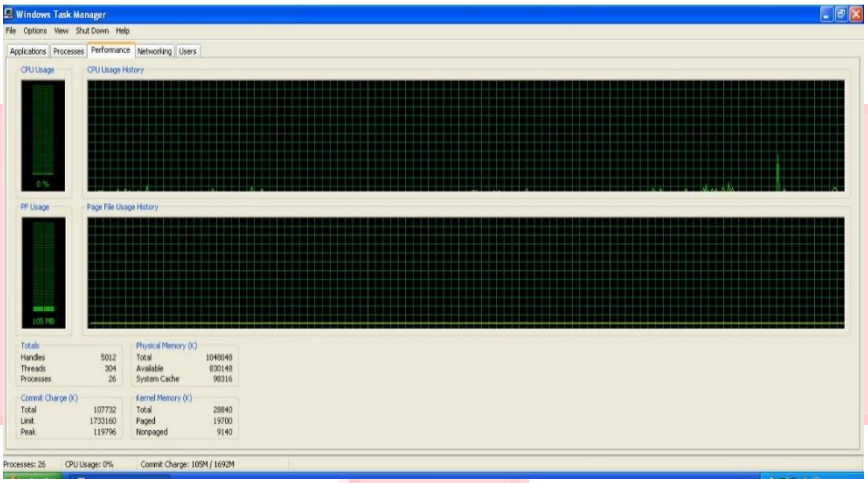
3.5.3 Performa pada System komputer

Pada gambar 23, merupakan performa dari windows xp yang tidak terinfeksi *malware poison ivy*. Trafik pada gambar tersebut diambil dalam waktu 5 menit. Trafik tersebut menunjukan bahwa windows xp berjalan dengan lancar, dimana penggunaan cpu memiliki kisaran 0% dan penggunaan memori sebesar 103MB dimana ini adalah hal wajar pada trafik performa windows xp.



Gambar 23. Performa sebelum terinfeksi poison ivy

Pada gambar 24, merupakan performa dari windows xp yang sudah terinfeksi *malware poison ivy*. Pada trafik tersebut performa dari windows xp tidak menunjukan adanya proses yang mencurigakan seperti penggunaan cpu dan memory yang berlebih. Pada trafik sebelum terinfeksi dan sesudah terinfeksi performa tidak memiliki perbedaan yang signifikan, dimana hal ini akan membuat korban kesulitan untuk mendeteksi *malware* melalui trafik performa.



Gambar 24. Performa sesudah terinfeksi poison ivy

4. KESIMPULAN

Berdasarkan hasil perancangan, pengujian dan analisa yang telah dilakukan maka dapat diambil beberapa kesimpulan sebagai berikut:

1. Malware *poison ivy* bekerja dengan cara menginject file dll loadlibrary.dll untuk mengontrol sebuah sistem computer.

2. Protokol yang digunakan malware Poison ivy adalah protocol TCP, untuk melakukan koneksi Malware *poison ivy* mengecek terlebih dahulu password sesuai atau tidak dengan yang ada di server.

3. Malware poison *ivy* tidak memakan terlalu banyak resource pada sebuah system computer.

4. Malware poison ivy menggunakan sebuah enkripsi untuk komunikasi client dan server, enkripsi tersebut diambil berdasarkan password yang ada pada client dan server.

REFERENCE

[1] T. Pajar Setia, N. Widiyasono, and A. Putra Aldya, “Analysis *Malware* Flawed Ammyy RAT Dengan Metode Reverse Engineering,” *J. Inform. J. Pengemb. IT*, vol. 3, no. 3, pp. 371–379, 2018, doi: 10.30591/jpit.v3i3.1019.

[2] T. A. Cahyanto, V. Wahanggara, and D. Ramadana, “Analisis dan Deteksi *Malware* Menggunakan Metode *Malware* Analisis Dinamis dan *Malware* Analisis Statis,” *Justindo, J. Sist. Teknol. Inf. Indones.*, vol. 2, no. 1, pp. 19–30, 2017.

[3] R. Adenansi and L. A. Novarina, “*Malware* dynamic,” *J. Educ. Inf. Commun. Technol.*, vol. 1, no. 1, pp. 37–43, 2017.

[4] “The Volatility Foundation - Open Source Memory Forensics.” [Online]. Available: https://www.volatilityfoundation.org/. [Accessed: 28-Sep-2020].

[5] Eldad Eilam, *Reversing__Secrets_of_Reverse_Engineering-Wiley(2005)*.

[6] I. L. Pribadi, R. Munadi, and Y. Purwanto, “IMPLEMENTASI SISTEM KEAMANAN

JARINGAN MENGGUNAKAN HONEYPOT DIONAEA, IDS, DAN CUCKOO SANDBOX," 2013.

[7] D. Kusswurm, *Modern X86 Assembly Language Programming*.

[8] Zelster, Lenny, 2001, Reverse Engineering Malware, www.zelster.com