BAB II

TINJAUAN PUSTAKA

1. Tinjauan Pustaka

2.1 Keamanan Sistem Komputer

Pada dasarnya komputer memiliki dua buah perangkat utama meliputi perangkat keras (*hardware*) dan perangkat lunak (*software*). Komponen dari *hardware* adalah CPU, RAM, *motherboard*, *Processor*, kartu VGA, dll. Sementara pada unit *Software* memuat program pendukung seperti *Operating System* (OS) dan perangkat lunak lainnya. Menurut salah satu ahlinya Howard dalam bukunya *"An analysis of security incidents on the internet*" mengungkapkan bahwa: "Keamanan komputer adalah tindakan pencegahan dari serangan pengguna komputer atau pengakses jaringan yang tidak bertanggung jawab". Dalam aspeknya keamanan komputer memiliki 8 (delapan) aspek yaitu: *authenthication, intergrity, non repudation, authority, confidentiality, privacy, availability, access control.* Sedangan model-model penyerangannya terdiri dari 4 (empat) model yaitu: *Interruption, interception, modification, fabrication*. (Sari, et al., 2020).

2.2 *Backdoor*

Backdoor (Pintu belakang) merupakan akses khusus yang dibuat oleh penyerang untuk dapat masuk kedalam sistem tanpa diketahui oleh operator sistem (Hafiz, et al., 2020). Biasa juga dikenal dengan istilah *web shell* merupakan salah satu kode yang digunakan hacker untuk mempertahankan akses (*maintaining access)* sistem secara ilegal (Sopaheluwakan & Chandra, 2020) (Anon., 2020).

Pada penelitian ini digunakan 2 sampel *backdoor,* kedua sampel ini adalah *backdoor* berjenis *Remote Access Control* (RAT), yaitu jenis *backdoor* yang memberikan penyerang akses untuk melihat, memodifikasi file dan fungsi sebuah sistem, memantau aktifitas korban, dan menggunakan sistem yang dijangkitnya untuk menyerang sistem lain (Kara & Aydos, 2019). *Backdoor*

dapat dikembangkan dan disebarkan menggunakan banyak media, salah satunya seperti sampel kedua *backdoor* dalam penelitian ini menggunakan sebuah aplikasi pembantu yang digunakan dalam sebuah *chatting platform* bernama 'Slack', *backdoor* jenis ini disebut Slackbot. Kerja Slackbot adalah membantu pengguna 'Slack' untuk melakukan kerja yang berulang (monoton) atau juga biasa berperan sebagai alarm yang mengingatkan pengguna tentang hal yang harus dikerjakan, pengguna bisa membuat Slackbot-nya sendiri menggunakan Bahasa pemrograman *phyton* atau *java*, dan hal inilah yang dimanfaatkan oleh penyerang sebagai celah untuk menyebarkan *malware* (Haque, 2019).

2.3 *VirtualBox*

Oracle VM VirtualBox adalah perangkat lunak virtualisasi, yang dapat digunakan untuk mengeksekusi sistem operasi tambahan di dalam sistem operasi utama. *VirtualBox* berfungsi untuk melakukan virtualisasi sistem operasi. *VirtualBox* juga dapat digunakan untuk membuat virtualisasi jaringan komputer sederhana. Penggunaan *VirtualBox* ditargetkan untuk Server, desktop dan penggunaan embedded. Berdasarkan jenis VMM yang ada, *Virtualbox* merupakan jenis *hypervisor* tipe 2 (dua) (Anam, et al., 2020)

2.4 *Malware Analysis Lab*

Malware Analysis Lab adalah lingkungan yang aman untuk menganalisis *malware*. Pada dasarnya, ini adalah lingkungan terisolasi yang berisi banyak alat berguna untuk analis *malware* yang membantu mereka dalam menganalisis perangkat lunak berbahaya. Kita harus membangun lab *malware* agar lebih proaktif terhadap ancaman baru dan modern yang dapat tiba-tiba menyerang organisasi kita. Ini juga merupakan bentuk deteksi lanjutan sebelum vendor antivirus menemukan spesimen *malware* baru. Cakupan lab analisis *malware* dapat ditentukan dengan memeriksa proses yang akan terjadi dalam proses analisis *malware* (Peppers, 2018) (A, 2018).

Metode yang biasa dilakukan untuk melakukan analisis *malware* terbagi menjadi 2 (dua) buah cabang utama yaitu penelitian statis dan dinamis.

Walaupun tujuan dari kedua metode ini sama namun *tools*, waktu dan pengalaman yang dibutuhkan untuk melakukan kedua metode ini sangatlah berbeda, perbedaan dasar dari kedua metode ini akan dijelaskan pada sub-bab berikutnya (FEDAK & STULRAJTER, 2020).

2.5 Analisa Statis

Dalam metode analisis statis ini *file malware* tidak akan diaktifkan secara langsung melainkan ditelusuri dan diteliti serta dianalisis terhadap kode sumber yang dituliskan didalam program *malware* dengan melakukan tahapan pembedahan terhadap program *malware* tersebut, sehingga informasi yang didapatkan sangatlah lengkap dan bisa memberikan gambaran yang sangat detail tentang mekanisme kerja *malware* tersebut secara keseluruhan (Cahyanto, et al., 2017). Fungsi dari analisa statis adalah untuk mengidentifikasi maksud berbahaya dari sebuah aplikasi (Kabakus & Dogru, 2018). Dalam analogi lain, ini dapat dikaitkan dengan otopsi kode - membedah kode yang 'mati' (Peppers, 2018). Untuk penelitian ini, analisis statis akan dibagi menjadi 3 tahap yaitu *Unpacking file,* identifikasi *malware*, dan analisis string.

2.6 *Packer* dan *Unpacking File*

Teknik *packing* secara umum digunakan oleh pengembang *malware* untuk mengkompres dan mengenkrip kode eksekusi untuk mengurangi ukuran *file* dan untuk melindungi kekayaan intelektual sebuah produk *executable file* (.exe) sehingga *malware* yang dibuat dapat menghindari analisa *malware* baik analisa statis maupun dinamis, walaupun begitu, kode pembongkaran (*unpacking*) biasanya disematkan di *file* yang dapat dieksekusi. Sehingga *file* eksekusi yang sudah di bungkus (*packed)* masih dapat dieksekusi secara normal dan informasi yang berkaitan dengan pembungkusan *file* tersebut bisa digunakan untuk analisis dan mendeteksi *malware*, proses pembongkaran *file* ini disebut *unpacking techniques* atau *decompressing* (Jung, et al., 2018).

2.7 ExeInfoPE

ExeInfoPE adalah salah satu aplikasi gratis untuk mengekstrasi informasi-informasi tentang *file packer* dari sebuah *file* yang bisa dieksekusi

(*executable file* (.exe)). Versi terbaru dari aplikasi (ver. 0.0.6.3) ini dapat mendeteksi lebih dari 1086 ciri khas *file packer* yang sering digunakan oleh pembuat malware. Aplikasi ini juga memberikan banyak informasi dari PE *header* sebuah *file*, juga fitur-fitur tambahan lainnya jika dipasang *add-on* yang dapat diunduh dari internet (FEDAK & STULRAJTER, 2020).

2.8 UPX *Packer*

UPX adalah *file packer* (pembungkus *file*) yang memiliki sumber terbuka (*open source), packer* ini tersedia di *github* dimana semua orang bisa mengunduhnya. *Packer* ini merupakan salah satu kelompok *packer* yang terkenal. Penggunaan aplikasi ini bersifat 2 (dua) arah dimana aplikasi ini bisa melakukan packing dan juga *unpacking* PE *header* dari sebuah executable *file*. Setiap *packer* memiliki satu set/kelompok binari pembungkus yang sama sehingga dapat dideteksi (Mohanta & Saldanha, 2020).

2.9 PEStudio

PEStudio adalah aplikasi yang digunakan untuk menemukan atribut-atribut seperti *hash, libraries,* dan *signatures* yang dimiliki oleh sebuah *malware* (Panjatian, et al., 2021).

2.10 VirusTotal

VirusTotal adalah alat untuk memindai *malware* yang tersedia secara online. Alat ini memiliki lebih dari 70 mesin pemindai *anti-malware* dan juga menyediakan laporan analisis serta *metadata* yang kaya akan informasi tentang *malware*. Alat ini dapat memindai *file*, URL dan *hash* untuk menentukan apakah mereka memiliki niat yang jahat saat diakses. Alat ini berkerjasama dengan 68 vendor keamanan lain untuk membangun *database* gabungan (VirusTotal Database) untuk menyediakan informasi yang paling relevan tentang *file*, web URL atau *hash* yang diinput kedalam alat ini (Peng, et al., 19).

2.11 Ghidra

Ghidra adalah perangkat lunak untuk *reverse-engineering*, yang dikembangkan oleh NSA, untuk NSA. Namun pada Maret tahun 2019

perangkat lunak ini dirilis ke publik agar bisa digunakan secara gratis. Ghidra adalah perangkat lunak pertama yang menyediakan *decompiler* yang stabil bisa digunakn untuk arsitektur aplikasi yang bervariasi, juga menyediakan sistem fitur-fitur tambahan berbasis *Java* yang luas, namun dengan penampilan informasi yang mudah dimengerti (Rohleder, 2019)

2.12 Analisa Dinamis

Analisis perilaku interaktif, juga disebut analisis dinamis melibatkan pengamatan *malware* yang berjalan secara langsung (Peppers, 2018). Pada metode ini sebuah *file* yang diperiksa akan diaktifkan dalam sebuah lingkungan yang *safe* baik pada sebuah mesin fisik yang telah disediakan sebagai laboratorium *malware* maupun yang berupa virtual (mesin virtual) untuk selanjutnya mampu dikumpulkan informasi mengenai dampaknya terhadap komputer ketika *file malware* menjalankan prosesnya. Sehingga dapat diketahui kegiatan apa saja yang dilakukan oleh *malware* saat berhasil menginfeksi sebuah komputer, pemeriksaan yang tercangkup dalam analisis ini adalah pemeriksaan proses yang berjalan di komputer, perubahan *registry*, komunikasi internet, dan juga *file file* yang ada di komputer yang telah terinfeksi untuk mencari apakah ada *file* yang hilang, diganti atau ditambahkan oleh penyerang (Cahyanto, et al., 2017). *Tools* yang digunakan untuk melakukan analisis dinamis pada penelitian ini adalah "Proccess Monitor", "Proccess Hacker, "ProcDot".

2.13 *Process* Monitor

Process Monitor adalah aplikasi yang digunakan untuk memantau proses pada sebuah sistem secara waktu yang langsung (*real-time)*. Aplikasi ini mencatat operasi yang terjadi pada *file (write, read, move)* dan operasi yang terjadi pada *system registry* (*reading, edit, record*). Aplikasi ini memungkinkan fiterisasi informasi yang canggih dimana informasi ini kemudian diekspor menjadi *file* (Qbeitah & Aldwairi, 2018).

2.14 *Process* Hacker

Process Hacker membantu pemantauan pada sumber yang dimiliki oleh sistem, mengobservasi bagaimana sebuah proses berinteraksi dengan

lingkungan (mesin), *debugging* perangkat lunak, dan mendeteksi malware (Qbeitah & Aldwairi, 2018).

2.15 ProcDOT

ProcDOT digunakan untuk mengkorelasikan, memfilterisasi, dan memvisualisasi konten yang dicatat oleh *Process Monitor*, aplikasi ini membantu menterjemahkan dan menginterpretasi data, mengindentifikasi hubungan dan mengartikan apa yang dilakukan oleh program jahat (*malicious)* (Qbeitah & Aldwairi, 2018).

2.16 FlareVM

Semua *tools* yang digunakan dalam penelitian ini dapat diunduh secara gratis di internet, namun untuk memudahkan pembuatan *malware analysis lab*, sebaiknya seorang penganalisa menggunakan FlareVM. FlareVM adalah paket security gratis dan bersifat sumber terbuka yang berbasis Windows didesain untuk *reverse engineers, malware analysis, incident responders, forensic dan penetration testers* karena paket ini berisi sejumlah *tools* yang berguna untuk melakukan kegiatan-kegiatan tersebut (Peppers, 2018).

2. Penelitian Sejenis

Table 1 Table Penelitian Sejenis

No	Judul	Rumusan Masalah	Metode/Teknologi	Saran/Kelemahan
1	The Ghost in the System: Technical Analysis of Remote Access Trojan	Membahas analisis secara mendetail dari pendeteksian RAT *Malware* di kompuer korban yang	Metode yang digunakan untuk menganalisa aktifitas *malware* adalah analisis statik dan dinamik	Untuk mencegah serangan *malware*, penulis menyarankan para pengguna internet untuk selalu waspada dan sadar terhadap *malware* yang berkeliaran di internet tidak

		terkena serangan RAT.		sembarangan menginstall aplikasi, memasang antivirus dan melakukan backup data untuk recovery
2	Experimental Analysis of Trojan Horse and Worm Attacks in Windows Environment	Penelitian ini membahas tentang kerentanan sebuah jaringan dari serangan RAT dan Worm, mencari tahu bagaimana serangan-serangan jenis ini dapat terjadi dan berjalan sesuai keinginan penyerang	Testbed eksperimental disiapkan untuk mendemonstrasikan potensi serangan *malware* di Windows lingkungan dan melakukan analisis.	-
3	Analisis dan Deteksi *Malware* Menggunakan Metode	Untuk membuktikan suatu software	Metode *Malware* Analisis Dinamis dan Statis merupakan kombinasi metode	disimpulkan bahwa dalam *file poison ivy* terdapat beberapa signature,

Malware Analisis Dinamis dan *Malware* analisis Statis	dikatakan *malware* adalah dengan mengetahui cara kerja program tersebut pada sistem komputer.	yang sesuai untuk menganalisa cara kerja *malware*.	filename, dan string yang sudah diteliti ternyata dapat melakukan proses login secara remote tanpa diketahui oleh pemilik komputer.

BAB III

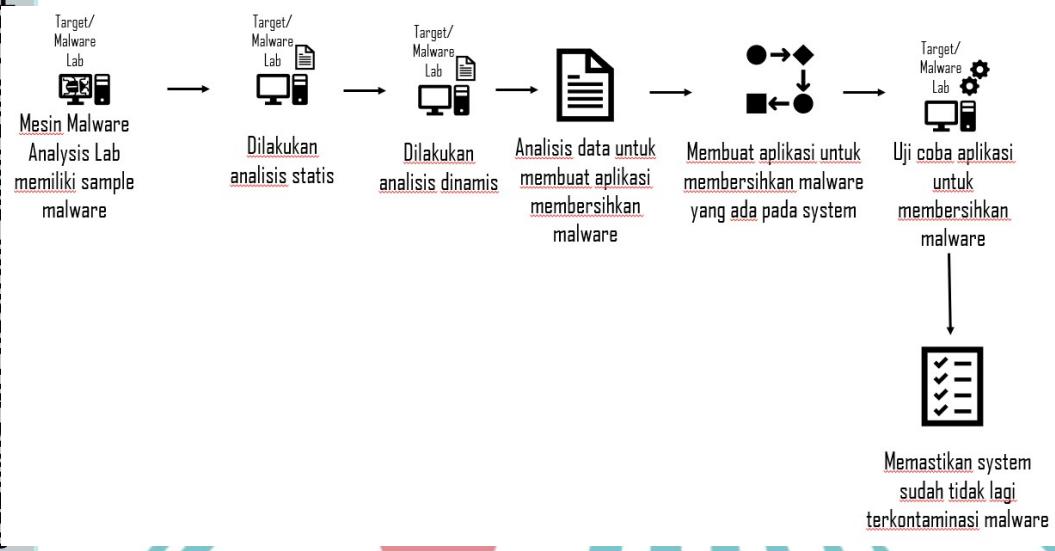
PERENCANAAN DAN REALISASI

3.1 Perancangan Sistem

Pada bab ini akan menjelaskan lebih rinci mengenai perencanaan dalam pembuatan lingkungan (*environment)* untuk menjalankan analisis serangan *malware*, dan perencanaan pembangunan aplikasi untuk membersihkan *malware backdoor* tersebut, metode yang digunakan adalah analisis dinamis dan analisis statis, sedangkan aplikasi dibuat menggunakan bahasa pemrograman *java* dan dibangun pada Apache NetBeans IDE. Penelitian ini dilakukan untuk mengetahui cara kerja sebuah *malware* jenis *Backdoor* dengan famili *Beast* Trojan dan *Slackbot*, akan dipaparkan rancangan menggunakan workflow dan skenario pengujian untuk memberikan gambaran yang lebih jelas kepada pembaca bagaimana berjalannya pengujian ini.

3.1.1 Workflow

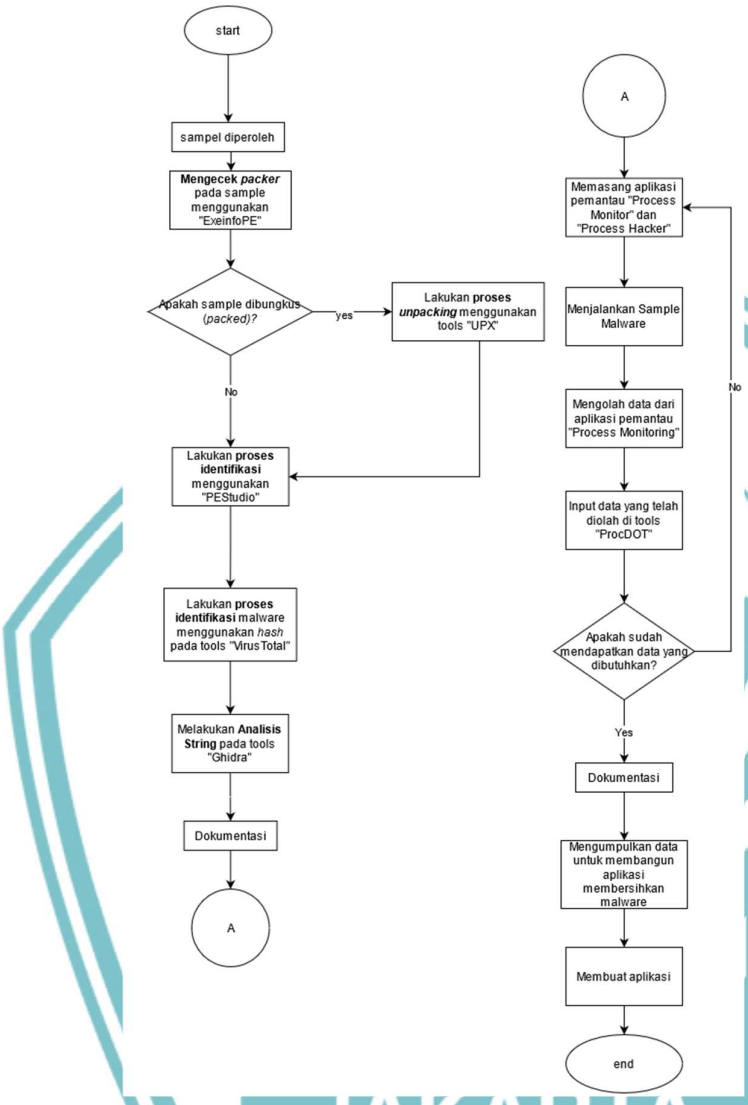
Seluruh pengerjaan dilakukan secara virtual menggunakan VirtualBox untuk menghindari kerusakan fatal pada data-data asli bila terjadi kesalahan dalam pelaksaan penelitian *malware*. Sampel *malware* yang telah dimiliki akan melewati proses analisis statis dimana proses ini dilakukan tanpa mengeksekusi *malware* tersebut dan hanya menganalisis kode-kode pada *malware* untuk mengetahui identitas dan kemampuan menyeluruh sebuah *malware,* selanjutnya *malware* akan melewati proses analisis dinamis yaitu untuk mengetahui bagaimana perilaku *malware* pada komputer, melihat perubahan-perubahan apa saja yang dilakukan oleh *malware* dari segi *file, process* dan *windows registry,* kemudian setelah cara kerja *malware* diketahui maka akan dibuat aplikasi untuk menghapus *malware* tersebut dari komputer yang telah terinfeksi. Alur penelitian ini dapat dilihat di *workflow* pada Gambar 3.1:



Gambar 3. 1 workflow dari penelitian

3.1.2 Skenario Analisa

Pada tahap analisa, hal yang dilakukan adalah melakukan analisa statis dan dinamis, tujuannya untuk mengambil sebanyak-banyaknya informasi tentang *malware* menggunakan kedua teknik tersebut, kemudian informasi yang didapatkan akan diolah untuk dimengerti sehingga dapat tercipta solusi yang dapat digunakan untuk membersihkan *malware* dari komputer. Sampel yang dimiliki akan melewati proses analisis statis yang dilakukan tanpa mengeksekusi *malware,* pada tahap ini *malware* akan melewati 3 (tiga) tahap yaitu pengecekan *packer*, kemudian tahap kedua adalah mengecek identitas *malware* tersebut dan pada tahap ketiga analisis *String* dimana kode-kode *malware* akan dicek agar diketahui kemampuan apa yang dimiliki *malware* tersebut. Setelah analisis statis maka *malware* akan melewati proses analisis dinamis, analisis ini dilakukan dengan memasang alat pemantau pada komputer agar perubahan yang dilakukan oleh *malware* dapat dipantau, lalu sampel *malware* akan dijalankan (dieksekusi) pada mesin komputer *malware analysis lab*. Semua data yang didapat dari proses analisis ini akan dianalisis agar didapatkan data untuk menyusun aplikasi yang dapat digunakan sebagai solusi untuk mengatasi *malware*. Secara grafik proses penelitian dapat dilihat pada *flowchart* di Gambar 3.2:



Gambar 3. 2 Flowchart analisa keseluruhan

3.1.3 Spesifikasi Perangkat dan *Software/Tools*

1) Mesin *Malware Analysis Lab*

Mesin yang digunakan untuk menampung *malware* lab ini diinstal pada VirtualBox. Untuk kepentingan penelitian, mesin ini juga berperan sebagai mesin target sehingga *malware* yang akan dianalisis dapat secara langsung dijalankan pada mesin dan dianalisa aktifitasnya. Berikut adalah spesifikasi dari mesin:

a. OS: Windows 10 Professional x64
b. Version: 20H2

c. OS Build: 19042.508
d. Processor: Intel(R0 Core™ i5-7200U CPU @2.50GHz 2.71GHz
e. Installed RAM: 2.00 GB
f. Disk Size: 80GB
g. Network Adapter: Network Connection Host Only
h. IP Address: 192.168.42.3
i. Windows Firewall: Domain, Private, Public network firewall aktif
j. Windows Defender: Tidak aktif sama sekali

2) *Tools* Untuk Analisis Statis

Dalam tahap ini dilakukan pengujian pada sampel *file malware Backdoor Beast* (server.exe) dan juga sampel *malware Backdoor Slackbot* (tnnbtib.exe) pada mesin *malware* analysis lab/target. Sampel *malware* akan diekstrasi sehingga didapatkan kode-kode yang diharapkan dapat menjelaskan perilaku *malware* secara lebih detail. Proses analisis ini dilakukan menggunakan *tools* sebagai berikut:

a. Exeinfope: *tools* untuk mengecek keadaan *file*
b. UPX: untuk membuka (*unpack) file*
c. PEStudio: untuk mengetahui identitas *malware*
d. VirusTotal: untuk mengetahui identitas *malware*
e. Ghidra: analisis string

Tools ini digunakan secara berurut karena setiap *tools* memiliki perannya masing-masing.

3) *Tools* Analisis Dinamis

Dalam tahap ini dilakukan pengujian dengan menjalankan/mengeksekusi sampel *file* RAT *malware Beast* (server.exe) dan sampel *malware Slackbot* (tnnbtib.exe) pada mesin *malware* analysis lab/target sehingga didapatkan informasi mengenai perilaku *malware* di komputer yang terinfeksi. Proses analisis ini dilakukan menggunakan *tools*:

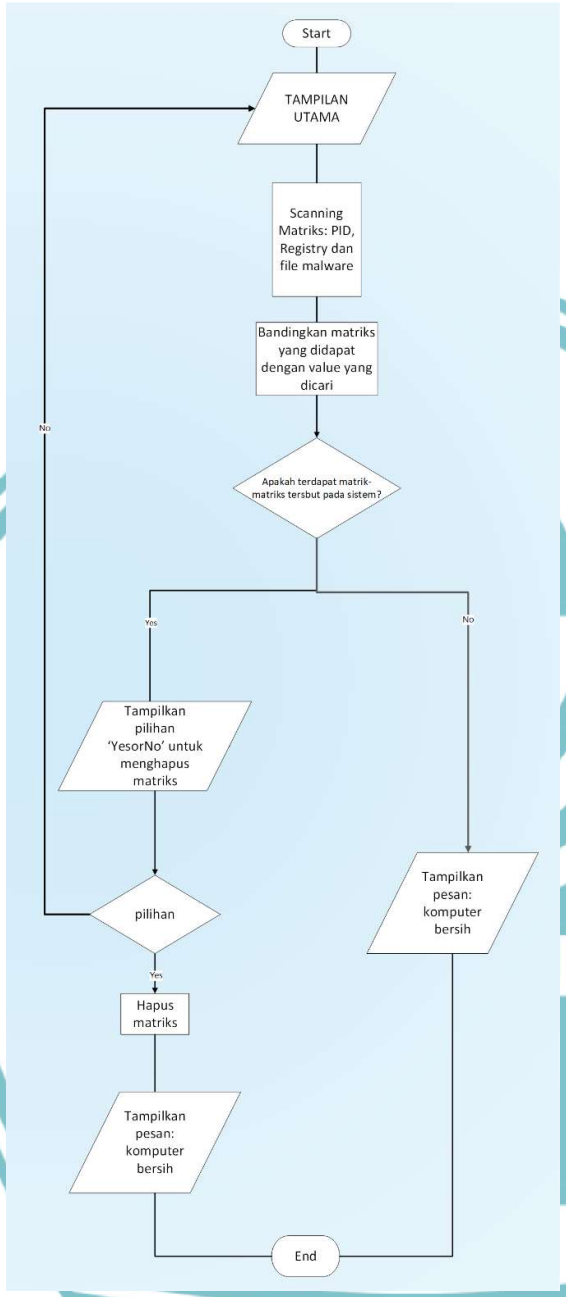
a. Proccess Monitor: untuk memantau aktifitas yang dilakukan *malware*
b. *Process* Hacker: untuk memantau bagaimana *malware* berinteraksi dengan sumber yang dimiliki mesin komputer

c. ProcDot: untuk menterjemahkan aktifitas yang ditangkap oleh *Process* Monitor
d. Cmd: melihat IP penyerang

Untuk analisis dinamis juga dilakukan secara berurutan.

4) Aplikasi untuk Melawan *Malware*

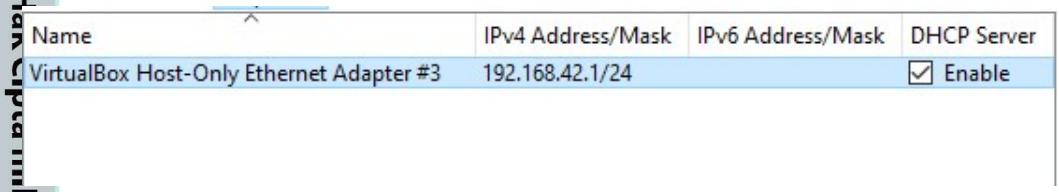
Setelah diketahui aktifitas apa saja yang dilakukan oleh *malware* maka selanjutnya pada penelitian ini akan dibuat aplikasi untuk menghapus *malware* dari komputer. Cara kerja dari aplikasi ini adalah untuk mencari matrik-matrik yang ditanam oleh *malware* yaitu *process id, windows registry* dan *file*, kemudian setiga matrik tersebut akan dihapus oleh aplikasi sehingga menghapus *malware* dari komputer. Matrik-matrik yang di-*input* dalam aplikasi ini berasal dari hasil analisa yang didapat saat melakukan analisa statis dan dinamis pada *malware*, harapan dari pembuatan aplikasi ini adalah untuk memudahkan pembersihan *malware* dengan satu kali aksi. Berikut flowchart struktur dari aplikasi yang akan dibuat dapat dilihat pada Gambar 3.3:



Gambar 3. 3 flowchart aplikasi untuk membersihkan malware

3.2 Realisasi Sistem

Semua mesin yang digunakan pada percobaan ini adalah mesin virtual untuk menghindari terinfeksinya data-data pribadi milik penulis, sehingga pada VirtualBox juga dilakukan kofigurasi *network* virtual dengan konfigurasi sebagai berikut terhilat pada Gambar 3.4:



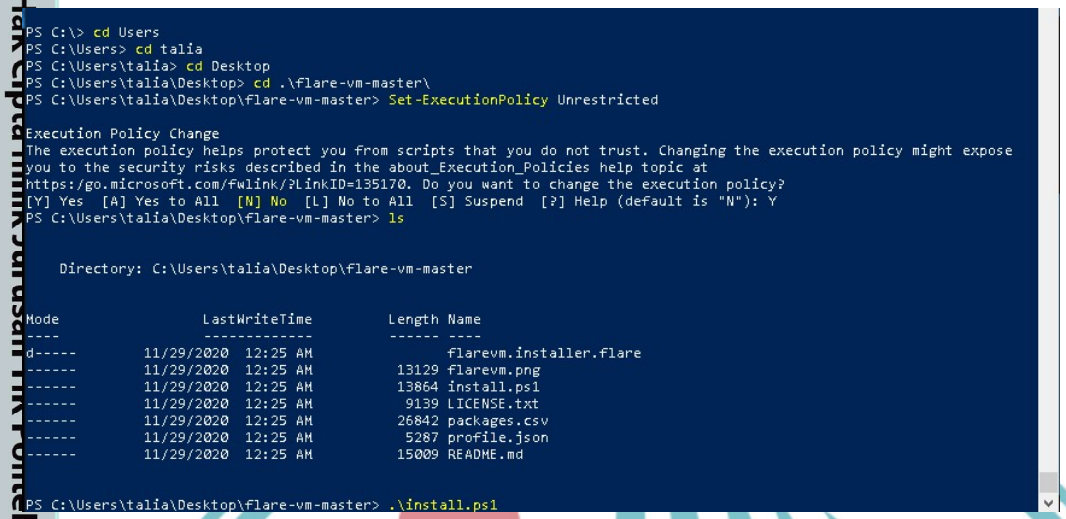
Gambar 3. 4 Jaringan Virtual yang Digunakan

Network setting pada mesin dipercobaan ini akan dikonfigurasi menjadi "Host Only" agar dapat tersambung ke jaringan virtual yang telah dibuat. Kemudian analisa dilakukan pada dua sampel *malware* bertipe sama yaitu *Backdoor* namun memiliki famili yang berbeda dimana sampel 1 adalah *malware* famili *Beast* dan sampel 2 adalah *malware Slackbot*.

3.2.1 Instalasi *Malware Analysis Lab*

Mesin yang berperan sebagai target dan juga berperan sebagai *Malware Analysis Lab* ini menggunakan Windows 10 versi 2H20 yang merupakan versi keluaran terbaru Windows pada saat penelitian dilakukan, matikan windows defender atau segala jenis anti-virus pada mesin ini agar aplikasi-aplikasi tersebut tidak menggangu proses analisis. Setelah dilakukan penginstalan OS pada mesin ini **diinstal Flare VM.**

Untuk menginstall FlareVM langkah pertama adalah mengunduh *file* distribusinya yang tersedia di internet. Setelah mengunduh maka instalasi dapat dilakukan. Installasi dilakukan menggunakan internet. Jalankan Windows PowerShell sebagai Administrator kemudian arahkan pindah ke direktori dimana FlareVM disimpan, kemudian jalankan command sebagai berikut dilampirkan pada Gambar 3.5:



Gambar 3. 5 installasi FlareVM menggunakan PowerShell

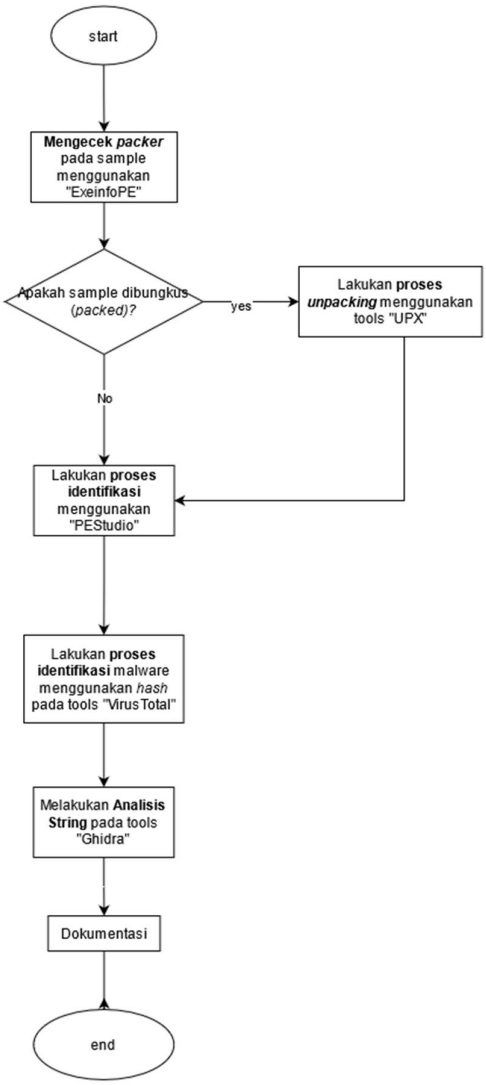
Command "set-executionpolicy" adalah untuk memberikan izin agar semua *file* yang berada pada folder ini dapat dieksekusi, kemudian jalankan *file* "install.ps1" dan tunggu sampai installasi selesai.

Gambar 3. 6 installasi FlareVM berhasil

Saat muncul tampilan seperti pada Gambar 3.6 maka FlareVM sudah berhasil terinstall dan siap dipakai.

3.2.2 Proses Analisa Statis

berikut *flowchart* pengerjaan analisis statis terlihat pada Gambar 3.7:

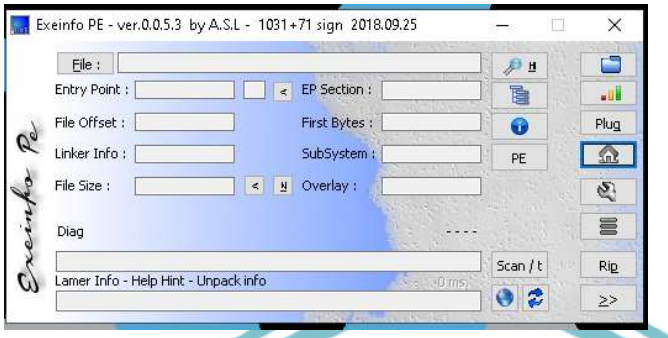


Gambar 3. 7 flowchart analisa statis

Analisis yang dibagi menjadi 3 (tiga) bagian ini menghasilkan data sebagai berikut:

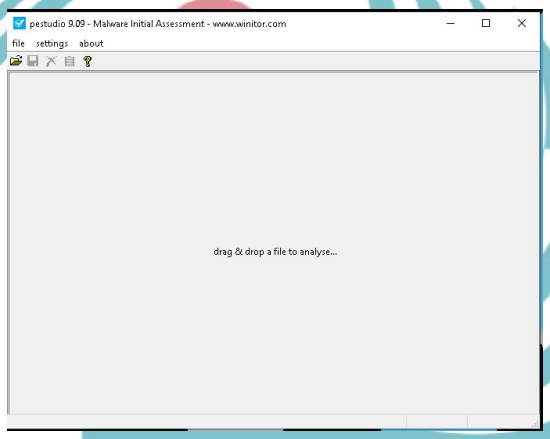
3.2.2.1 Proses *Unpacking*

Bagian pertama dari tahap statis ini adalah mengecek kondisi apakah *file* ini butuh melewati proses *unpacking* atau tidak, proses ini dilakukan menggunakan "exeinfope". Bila *malware* yang dimiliki adalah *file packed* (*file* yang telah dikompres kontennya) maka *file* harus melewati proses ***unpacking*** menggunakan aplikasi "upx". kedua *tools* ini dapat diunduh di internet dan proses penginstallan menggunakan *install wizard.* Tampilan utama kedua tools yang menunjukkan bahwa instalasi berhasil dapat dilihat pada Gambar 3.8:



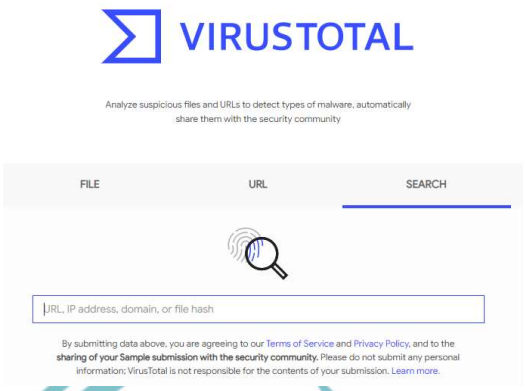
Gambar 3. 8 tampilan tools exeinfope

3.2.2.2 Proses Identifikasi *Malware*



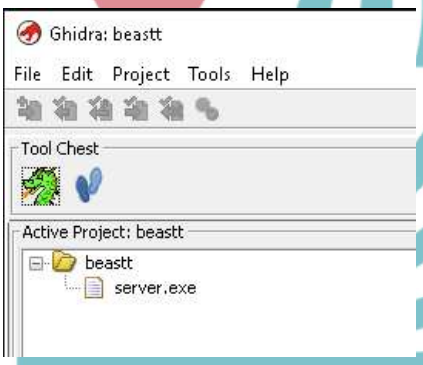
Gambar 3. 9 tools 'pestudio'

Selanjutnya akan digunakan *tools* "pestudio" untuk melakukan melihat informasi-informasi tentang identitas sang *file* tersebut, hal ini dilakukan untuk menunjukkan *hashing* milik *malware*, dimana *hashing* ini sifatnya unik seperti sidik jari sebuah *malware* sehingga identitas *malware* dapat diketahui dengan melakukan pencarian pada *tools* "VirusTotal", *tools* ini akan membantu untuk mengidentifikasi apakah *file* yang dianalisis merupakan sebuah *file* jahat (*malware)* atau tidak. "pestudio" dapat diunduh di internet dan diinstal pada mesin, sedangkan "VirusTotal" merupakan *tools* bersifat *multi-platform* yang tersedia dalam bentuk aplikasi juga dapat diakses di web internet: https://www.virustotal.com/gui/. Tampilan dari kedua *tools* ini dapat dilihat pada Gambar 3.9 dan Gambar 3.10.



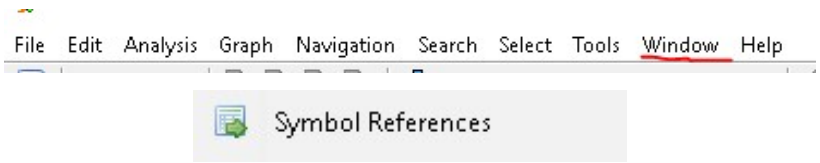
Gambar 3. 10 tools virustotal

3.2.2.3 Analisa String



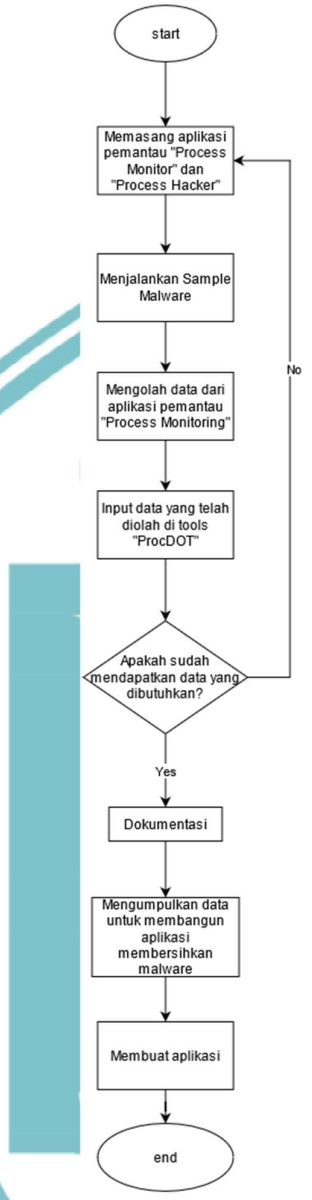
Gambar 3. 11 malware sudah diimport ke ghidra

Tahap terakhir pada analisis statis dilakukan menggunakan aplikasi "Ghidra", *tools* ini juga tersedia untuk diunduh secara gratis di internet. Pada proses ini dilakukan analisis string pada *file malware* untuk mencari tahu hal-hal apa saja yang dapat dilakukan oleh *malware*. "Ghidra" memberikan fitur yang sangat banyak, untuk memulai analisis *file* yang akan dianalisis akan di *import* ke dalam aplikasi sehingga menjadi proyek aktif pada "Ghidra" seperti pada Gambar 3.11. Analisis bisa dilakukan menggunakan beberapa cara dalam *tools* ini, salah satunya adalah melihat *symbol references*, yang dapat dilihat pada tab "windows" seperti pada Gambar 3.12:



Gambar 3. 12 symbol references

3.2.4 Analisis Dinamis

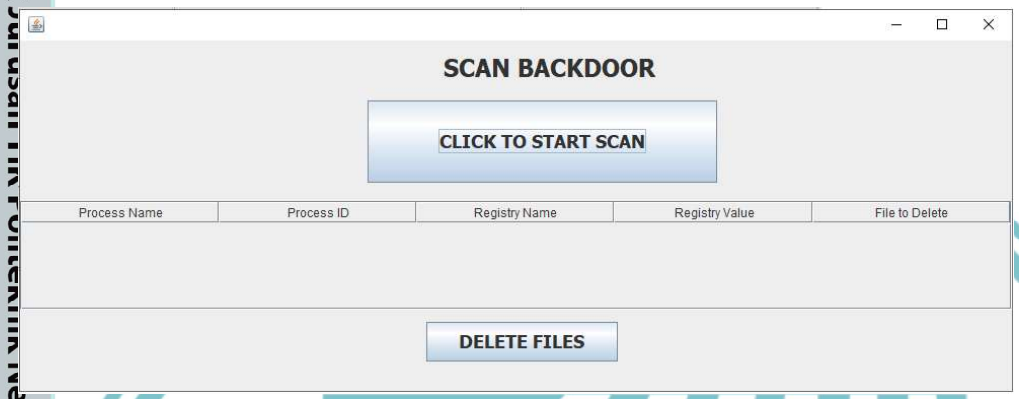


Gambar 3. 13 flowchart analisis dinamis

Seperti terlihat pada *flowchart* di Gambar 3.13, analisa ini dilakuakan dengan cara memonitoring perubahan yang terjadi pada komputer saat *malware* dijalankan. Perubahan yang dimaksud adalah perubahan pada *file* dan *registry* pada komputer yang terinfeksi. Untuk memonitoring digunakan *tools* bernama "*Process* Monitor" dan "*Process* Hacker" sedangkan untuk analisanya digunakan *tools* bernama "ProcDOT" yang dapat mengolah data dari "*Process* Monitor" menjadi grafik untuk proses penganalisaan yang lebih mudah.

3.2.3 Implementasi Tampilan Aplikasi

Aplikasi dibuat menggunakan bahasa pemrograman java dan menggunakan GUI untuk memudahkan penggunaan. Berikut tampilan utama aplikasi terlihat pada Gambar 3.14:



Gambar 3. 14 GUI aplikasi pembersih malware

Aplikasi ini memiliki beberapa fungsi, bisa dilihat pada Tabel 2 menjelaskan fungsi dan cara kerja fungsi pada aplikasi ini:

Table 2 tabel fungsi pada aplikasi serta penjelasan tentang fungsi tersebut

changeVariable() changeVariable2()	Berikut adalah function yang terdapat pada aplikasi. Langkah pertama yang akan dilakukan oleh aplikasi adalah mencari matrik (PID, Registry, files) yang dimiliki oleh *malware* yang sedang berjalan, kemudian hasil pencarian akan disimpan pada sebuah *file txt*.
compareFiles() compareProcess() compareRegistry()	Data yang sudah disimpan pada *file* akan dibandingan dengan matriks *malware* yang sudah diketahui *value*-nya dari hasil analisa. Ketiga fungsi ini memiliki cara kerja yang sama yaitu membandingkan data dengan variable yang di-*input* pada fungsi

	'changeVariable()' yang membedakan ketiga fungsi ini adalah data yang digiunakan yaitu *file*, *process* dan registry.
tulisKeTable() tulisKeTxt()	Jika ditemukan hasil yang cocok maka hasil akan ditulis ke *file* txt lain dan ditampilkan pada table pada tampilan awal GUI
DeleteThings()	Fungsi terakhir adalah fungsi untuk menghapus matriks yang telah ditemukan, dimana dengan melakukan hal ini *malware* juga ikut terhapus dari mesin.

Pada fungsi *changeVariable()* inilah data dari hasil analisis statis dan dinamis diperlukan, karena isi (*value)* dari variabel di aplikasi ini berbeda setiap *malware*, sehingga dibutuhkan data yang tepat, berikut perbedaan isi dari variable pada setiap *malware*, bisa dilihat perbedaan isi dari variable pada setiap *malware* seperti pada Gambar 3.15:

```
public void changeVariable2() throws IOException, InterruptedException{

        res="tnnbtib.exe";
        Process ab= Runtime.getRuntime().exec("cmd /c taskkill /IM \"tnnbtib.exe\" /F");
        Process bb= Runtime.getRuntime().exec("cmd /c reg query hkey_local_machine\\software\\wow6432node\\microsoft\\windows\\currentversion\\run  > regs.txt");
        Process cb= Runtime.getRuntime().exec("cmd /c dir /S -a \"c:\\tnnbtib.exe\" > files.txt");

        //compareProcess();
        compareRegistry();
        compareFiles();



}

public void changeVariable() throws IOException, InterruptedException{
    //String combox = jComboBox1.getSelectedItem().toString();


        Process a= Runtime.getRuntime().exec("cmd /c tasklist /svc | find \"svchost.exe\" > proc.txt");
        Process b= Runtime.getRuntime().exec("cmd /c reg query hkey_current_user\\software\\microsoft\\windows\\currentversion\\run > regs.txt");
        Process c= Runtime.getRuntime().exec("cmd /c dir /S /A \"c:\\svchost.exe\" > files.txt");

        compareProcess();
        compareRegistry();
        compareFiles();


}
```

Gambar 3. 15 kode fungsi 'changeVariable()' variable pada malware

```
String s;
String rep="";
String filePath = "proc.txt";
File fl=new File(filePath); //Creation of File Descriptor for input file
String[] words=null;  //Intialize the word Array
FileReader fr = new FileReader(fl);  //Creation of File Reader object
BufferedReader br = new BufferedReader(fr); //Creation of BufferedReader object
int count=0;   //Intialize the word to zero
while((s=br.readLine())!= null)   //Reading Content from the file
{

   rep = s.replaceAll("\\s{2,}", " ").trim();
   words=s.split(" ");  //Split the word using space
    for (String word : words)
    {
        if (word.equals(input_nama_proc)){

            count++;
            //System.out.println(word);
            //System.out.println(rep);
            res=rep.split(" ")[0];
            res2=rep.split(" ")[1];
        }
    }
}
```

Gambar 3. 16 kode fungsi 'compareProcess()'

```
String s;
String rep="";
String filePath = "files.txt";
```

Gambar 3. 17 teks file yang digunakan dalam fungsi 'comapreFiles()'

```
Thread.sleep(60000);
File reg=new File("regs.txt"); //Creation of
String[] words=null;  //Intialize the word Ar
FileReader fr2 = new FileReader(reg);  //Crea
BufferedReader br = new BufferedReader(fr2);
```

Gambar 3. 18 teks file yang digunakan untuk fungsi 'compareRegs()'

3 (tiga) fungsi pada aplikasi ini memiliki cara kerja yang sama yaitu membandingkan variable *input malware* dengan *text file* yang didapat dari hasil *scanning process*, registry dan *file.* Perbedaan *text file* dapat diliat pada Gambar 3.16, Gambar 3.17 dan Gambar 3.18. Seperti terlihat di kode pada Gambar 3.16 Fungsi ini menggunakan *for loop* untuk mencari kata dan baris yang dibutuhkan, dimana baris teks tersebut akan di print ke *file* baru untuk ditampilkan pada table ditampilan awal aplikasi.

```
public void tulisKeTxt()throws IOException {


  FileWriter fw = new FileWriter("test.txt", true);
      try (BufferedWriter bw = new BufferedWriter(fw)) {


          if(file==false){
              res_file="";
              res2_file="";
          }
          else {
              bw.write(res + "," + res2 + "," + reg_res + "," + reg_res2 + "," + res_file + "\\" + res2_file);
              bw.newLine();


          }

      }

  }
```

Gambar 3. 19 kode fungsi 'tulisKeTxt()'

Gambar 3.19 menunjukkan susunan kode bagaimana aplikasi menyimpan data yang didapat dari pencarian *malware* ke dalam *text file* agar dapat diakses oleh fungsi "tulisKeTable()" untuk ditampilkan pada tabel di halaman utama aplikasi. Berikut adalah kode dari fungsi yang mengatur tampilan pada table di halaman utama aplikasi dapat dilihat pada Gambar 3.20:

```
public void tulisKeTable() throws FileNotFoundException, IOException{
    String filemalware = "test.txt";
        File malware = new File(filemalware);

        BufferedReader rmal = new BufferedReader(new FileReader(malware));
          // get the first line
          // get the columns name from the first line
          // set columns name to the jtable model
          String firstLine = rmal.readLine().trim();
          String[] columnsName = firstLine.split(",");
          DefaultTableModel model = (DefaultTableModel)jTable1.getModel();
          model.setColumnIdentifiers(columnsName);

          // get lines from txt file
          String[] a = {res + "," + res2 + "," + reg_res + "," + reg_res2 + "," + res_file + "\\" + res2_file};

          // extratct data from lines
          // set data to jtable model
          for(int i = 0; i < a.length; i++)
          {
              String line = a[i].trim();
              String[] dataRow = line.split(",");
              model.addRow(dataRow);
          }


}
```

Gambar 3. 20 kode fungsi 'tulisKeTable'

BAB IV
PEMBAHASAN

4.1 Pengujian

Pengujian ini dilakukan untuk mengetahui efek yang disebabkan oleh *malware* berjenis *Backdoor* bernama *Beast Trojan* dan *Slackbot* pada sebuah komputer yang menjalankan Windows 10 OS. Hasil akhir dari penelitian ini adalah informasi-informasi tentang *malware* yang cukup untuk mengetahui identitas dan perilaku *malware* dan juga informasi yang cukup untuk membangun sebuah aplikasi yang akan digunakan untuk membersihkan *malware* tersebut dari komputer yang telah terinfeksi.

4.2 Deskripsi Pengujian

Malware yang digunakan yaitu *Beast Trojan* yang telah dikonfigurasi dimana server dari *malware* tersebut adalah komputer sang penyerang dengan.

Pada pengujian ini dilakukan menggunakan analisis statis dan dinamis, dimana:

1. Pada analisis statis akan dilakukan analisa tanpa pengeksekusian *malware*, menggunakan "exeinfope", "pestudio", "upx", dan analisa String menggunakan "Ghidra".
2. Analisis dinamis akan dilakukan pengeksekusian *malware* untuk dilakukan *monitoring* perubahan yang terjadi pada mesin yang terinfeksi *malware* dari segi *registry,* dan *file* menggunakan "*Process* Monitor", "*Process* Hacker", dan "ProcDOT".
3. Kemudian akan dilakukan pengujian untuk memastikan aplikasi mampu menghapus *malware* yang menginfeksi mesin, pada proses ini aplikasi yang dibangun akan dijalankan, kemudian akan dilihat koneksi yang sedang berjalan pada mesin menggunakan *netstat* cmd. Bila sudah tidak ada alamat IP asing yang tertanam dalam mesin maka akan dinyatakan aplikasi berhasil menghapus *malware* yang ada.

4.3 Prosedur Pengujian

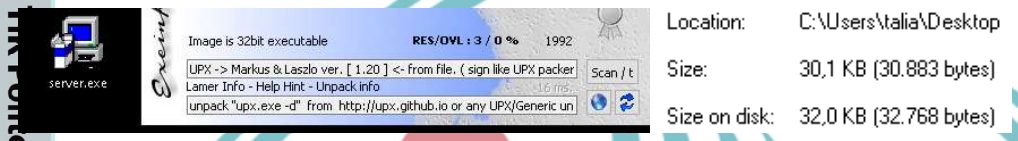
Pengujian memiliki beberapa tahap dan dilakukan setelah semua sistem yaitu sistem penyerang dan sistem target yang juga berperan sebagai *malware* analisis

lab terpasang. Semua analisa dilakukan pada mesin virtual *malware analysis lab.* Berikut adalah proses analisa *malware* yang dilakukan pada penelitian ini:

4.3.1 Proses Analisis Statis

Proses yang dimulai dari **analisa statis**, pada tahap ini dibagi menjadi beberapa bagian, diantaranya:

4.3.1.1 Proses *Unpacking*



Gambar 4. 1 cek file sampel *1 masih packed sehingga tools menyarankan untuk melakukan proses unpack*

a. Bagian pertama dari tahap statis ini adalah mengecek kondisi apakah *file* ini butuh melewati proses *unpacking* atau tidak dan mengidentifikasi *packer* yang digunakan, proses ini dilakukan menggunakan "exeinfope" seperti pada Gambar 4.1 diatas.
b. Bila *malware* yang dimiliki adalah *file packed* (*file* yang telah dikompres kontennya) maka informasi yang terdapat didalamnya juga terkompres sehingga informasi tersebut tidaklah lengkap untuk analisis, sehingga *file* harus melewati proses ***unpack.***
c. Cara menggunakan aplikasi ini hanyalah dengan melakukan *drag and drop file* yang ingin dianalisa ke *icon* exeinfope, kemudian aplikasi akan dengan sendirinya memproses PE *Header* dari *file* yang di-*input,* dan memberikan *output* informasi tentang *packer* yang digunakan.
d. Setelah informasi tentang *packer* didapatkan maka proses *unpacking* dilakukan menggunakan *tools* yang berbeda-beda sesuai dengan *packer* yang digunakan oleh *malware*. Untuk penelitian kali ini karena dapat diketahui pembuat *malware* menggunakan aplikasi *pakcer* "UPX" untuk membungkus *malware*-nya, maka akan digunakan aplikasi "UPX" untuk membongkar *malware* tersebut. Cara menggunakan aplikasi UPX adalah dengan mengaksesnya melalui *Windows PowerShell* dengan mengetik *command* "upx -d [diamna *file* disimpan]"

4.3.1.2 Proses Identifikasi *Malware*

a. Untuk mengidentifikasi *malware,* akan digunakan *tools* "pestudio" pada *file* yang sudah tidak lagi dibungkus (*packed*) untuk melihat informasi-informasi tentang identitas sang *file* tersebut, hal ini dilakukan untuk menentukan apakah *file* yang dipunya benar sebuah *malware* atau tidak.
b. *Tools* ini akan menunjukkan *hashing* milik malware, dimana hashing ini sifatnya unik seperti sidik jari sebuah *malware* sehingga identitas *malware* dapat diketahui dengan melakukan pencarian pada *tools* "VirusTotal".
c. *Hash* yang didapatkan dari *tools* "pestudio" akan di-*input* ke GUI pencariaan "VirusTotal" dan *tools* ini akan mencari memindai *hash* yang diinput di database milik "VirusTotal" yang merupakan database tentang *malware-malware* yang dikenali oleh kurang lebih 70 perusahaan *cyber-security* di seluruh dunia. Hasil pemindaian akan menunjukkan identitas tentang *malware*.

4.3.1.3 Analisis String

a. Setelah mengetahui dengan pasti bahwa *file* yang dimiliki adalah sebuah *malware*, maka langkah selanjutnya pada analisis ini adalah menganalisis kode dari *file* tersebut.
b. *Tools* yang digunakan untuk melakukan analisis ini adalah "Ghidra".
c. *File* yang akan dianalisis akan di-*import* kedalam "Ghidra" lalu biarkan ghidra melakukan analisis pada *file* tersebut, kunci dari analisis ini adalah melihat *symbol reference* yang merupakan *function,* atau *variable* pada *malware* yang ditandai penting untuk dianalisis oleh aplikasi "Ghidra". Kemudian hasil dari analisis bisa dirangkum agar menjadi masuk akal.

4.3.2 Analisa Dinamis

Pada tahap analisis dinamis hal paling penting yang harus dilakukan sebelum pengeksekusian adalah pemantauan (*monitoring)* pada mesin, karena hasil data yang diharapkan dari analisis ini adalah untuk mengetahui perubahan apa saja yang dilakukan oleh sang *malware* di mesin, maka dari itu, proses analisis dinamis adalah sebagai berikut:

a. Sebelum menjalankan/mengeksekusi *malware*, pastikan aplikasi "*Process* Monitor" berjalan karena aplikasi ini adalah aplikasi yang akan memantau dan mencatat proses yang berjalan pada komputer
b. Digunakan juga aplikasi "*Process* Hacker" untuk melihat proses-proses yang sedang berjalan pada komputer
c. Setelah *tools* monitoring telah berjalan, maka *malware* akan dijalankan untuk beberapa saat agar perubahan yang disebabkan oleh *malware* tersebut dapat tercatat oleh aplikasi
d. Setelah perubahan telah tercatat oleh "*Process* Monitor" maka data dari *tools* ini akan dikonversi menurut format "ProcDOT" dan data ini akan diolah oleh "ProcDOT" menjadi grafik untuk proses penganalisaan yang lebih mudah.

4.3.3 Pengujian Aplikasi Untuk Menghapus *Malware*

Proses pengujian ini dilakukan pada saat komputer sudah terinfeksi oleh *malware*, parameter yang diinput pada program adalah *input* yang sudah ditargetkan untuk membunuh dan menghapus matrik-matrik yang ditinggalkan oleh *malware* pada komputer yang terinfeksi. Untuk menguji apakah aplikasi sudah bisa menghapus *malware* atau tidak program akan dijalankan 1 kali, kemudian akan digunakan *tools* "RegShot" untuk membandingan apakah *windows registry* dan *file* yang ditanam oleh *malware* sudah dihapus oleh aplikasi sesudah aplikasi dieksekusi. Untuk memastikan apakah mesin masih terhubung dengan penyerang menggunakan *netstat* di cmd.

4.4 Data Hasil Pengujian

Data hasil pengujian terhadap *Backdoor Beast* dan *Slackbot* dilakukan pada mesin *Malware Analysis Lab*, dimana penguslackbojian dilakukan sesuai dengan prosedur pengujian yang telah dijelaskan pada sub bab sebelumnya.

4.4.1 Analisis Statis

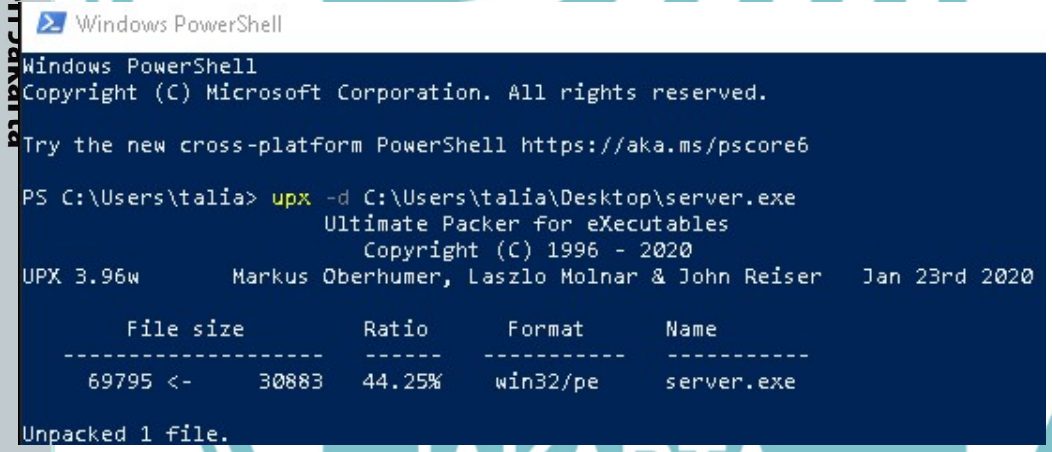
Analisis yang dibagi menjadi 3 (tiga) bagian ini menghasilkan data sebagai berikut:

4.4.1.1 Proses *Unpacking*

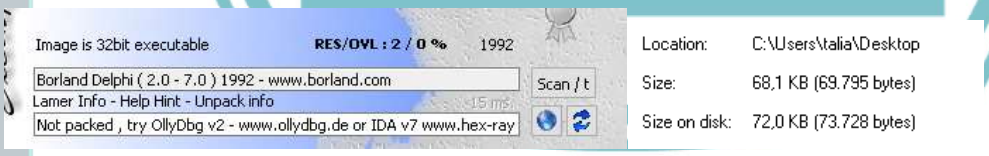
Pada Sampel 1 *Beast Malware* (server.exe):

Pada tahap ini penganalisa memastikan bahwa kode dari *file* exe yang dimiliki tidaklah dikompres menggunakan *file packer*, *file packer* adalah aplikasi yang digunakan untuk mengkompres konten-konten sebuah *file*, hal ini dilakukan untuk mempersulit seorang analis untuk menganalisa *malware* secara statis terutama penganalisaan *string malware*. Cara mengecek hal ini digunakan *tools* "exeinfope", hasil output dapat dilihat pada Gambar 4.1 diatas.

Hasil eksaminasi dari aplikasi "exeinfope" menunjukkan bahwa *file* exe yang dimiliki masih dikompress, *tools* ini memberi info tentang *packer* yang digunakan yaitu *packer* bernama "UPX Markus & Laszlo ver. [1.20]" dan pada kolom selanjutnya *tools* ini memberikan saran aplikasi apa yang digunakan untuk melakukan *unpack* berikut adalah proses *unpacking* seperti dilihat pada Gambar 4.2 dibawah:



Gambar 4. 2 command unpacking pada sampel *1*



Gambar 4. 3 cek file sampel *1 menunjukkan sudah tidak lagi packed sehingga ukuran file bertambah*

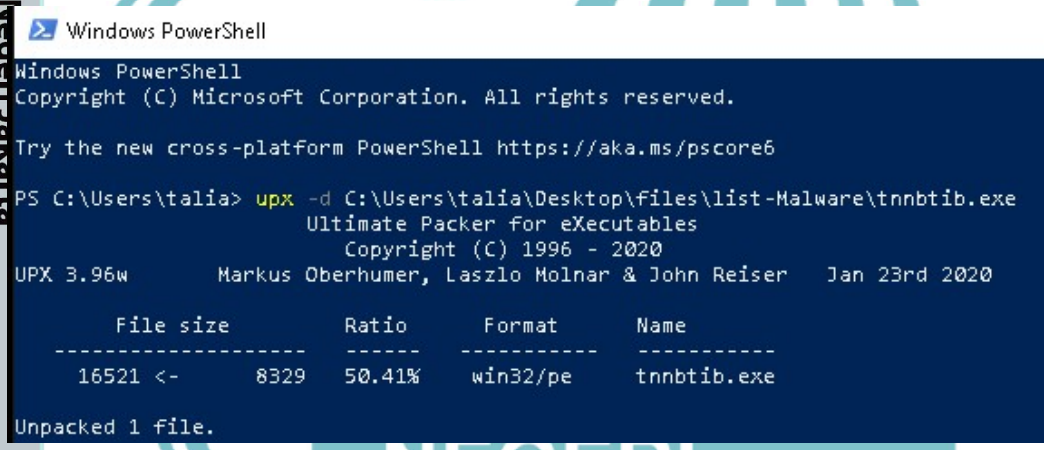
Setelah menjalankan command "*upx -d C:\Users\talia\Desktop\server.exe"* yang mana "C:\Users\talia\Desktop\server.exe" adalah *path* dimana *file* exe disimpan, maka proses *unpacking* pun selesai, dapat dilihat pada Gambar 4.3 perbedaan ukuan *file* exe sebelum dan sesudah proses decompressing, sebelum proses *file* hanya

memiliki ukuran 32,0 kb, tetapi setelah proses decompressing ukuran *file* menjadi lebih besar yaitu 72,0kb.

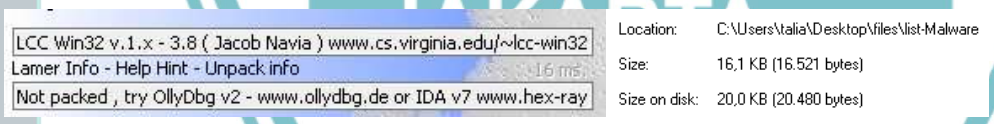
Pada Sampel 2 *Slackbot* (tnnbtib.exe):

Gambar 4. 4 cek file sampel 2 masih packed sehingga tools menyarankan untuk melakukan proses unpack

Melihat Gambar 4.4 diatas, sampel 2 juga menggunakan *packer* yang sama sehingga *tools* yang digunakan untuk melakukan proses *unpack* juga sama, terlampir pada Gambar 4.5 di bawah:



Gambar 4. 5 command unpacking pada sampel *2*



Gambar 4. 6 cek file sampel *1 menunjukkan sudah tidak lagi packed sehingga ukuran file bertambah*

Dapat dilihat pada Gambar 4.6 perubahan yang terjadi pada proses ini menunjukkan proses *unpacking* berhasil, sehingga dapat berlanjut ke tahap selanjutnya yaitu proses identifikasi *malware*.

4.4.1.2 Proses Identifikasi *Malware*

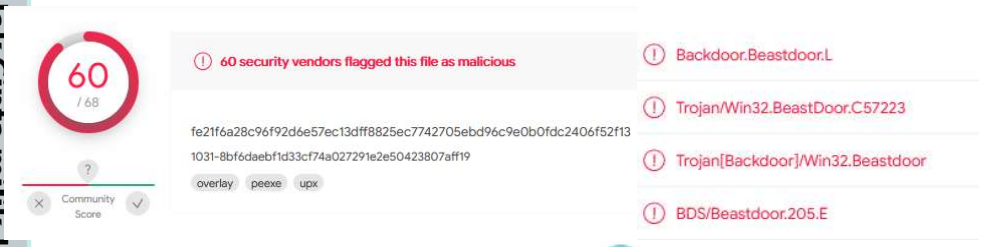
Pada sampel 1, *Beast Malware* (server.exe):

Kemudian *tools* kedua yang digunakan akan "pestudio". Saat *file* sudah melewati proses *unpacking* terdapat banyak informasi yang bisa diambil menggunakan *tools* ini, berikut adalah informasi yang didapatkan lebih detilnya terlihat pada tabel 3:

Table 3 Informasi file 'server.exe' didapatkan dari 'pestudio'

Property	Value
Md5	7D48B515331FECED4786626B9DA7E42B
Sha1	87F7E889D59ADD954DB794EF5DA6A14F71074B93
Sha256	FE21F6A28C96F92D6E57EC13DFF8825EC7742705EBD96C9E0B0FDC2406F52F13
Imphash	DEDF17FFA2A579E084585D51B9FD09BA
first-bytes-text	M Z P .. .. @ .. ..
File-type	Executable, dibuat menggunakan Bahasa programming Delphi
CPU	32-bit
Bahasa	Romania

Tools memberikan 4 (empat) jenis *hash* yang dapat digunakan untuk mengidentifikasi *malware*, pada *malware* informasi muncul ketika dilakukan pencarian pada *import hash (imphash),* import *hash* digunakan saat 3 (tiga) *hash* utama dari sebuah *malware* telah diubah atau tidak lagi bisa digunakan untuk mengidentifikasi *malware.* Digunakanlah *import hash* yaitu *hash* dari *import-import* yang dipakai di *malware*. Dari hasil pencarian pada *tools VirusTotal* seperti pada Gambar 4.7 di bawah, didapatkan hasil bahwa 60 dari 68 anti-virus menyatakan bahwa *file* ini adalah *file* yang mengandung konten *malicious* (jahat), dan dapat diidentifikasi nama *malware* adalah *Beast Backdoor*. Dari *tools* ini juga didapatkan first-bytes-text yang mengindikasikan tipe *file* dari *malware* ini adalah *executeable* yang dibuat mengunakan Delphi, dan persyaratan untuk menjalankan *malware* ini adalah pada CPU 32-bit. Juga terdeteksi bahasa yang digunakan mesih komputer saat compile program dilakukan adalah Romania, sehingga bisa diasumsikan bahwa penyerang atau setidaknya pembuat *malware* berasal dari Romania.



Gambar 4. 7 hasil dari identifikasi malware sampel *1 pada 'Virustotal'*

Sumber:

https://www.virustotal.com/gui/file/fe21f6a28c96f92d6e57ec13dff8825ec7742705ebd96c9e0b0fdc2406f52f13/detection

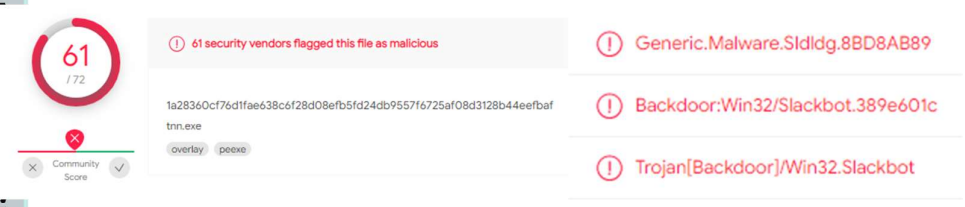
Pada samle 2 *Slackbot* (tnnbtib.exe):

Proses yang sama dilakukan pada sampel 2, untuk mengindentifikasi *malware* ini juga digunakan *tools* "pestudio", berikut adalah *output* data dari analisa yang dilakukan oleh *tools* secara ringkas dijelaskan pada tabel 4:

Table 4 Informasi file 'tnnbtib.exe' didapatkan dari 'pestudio'

Property	Value
Md5	CA7C8D66A45AEC18305FC59035F85FD6
Sha1	624AADCC86A824B9A64457026498FAE5413D66E6
Sha256	1A28360CF76D1FAE638C6F28D08EFB5FD24DB9557F6725AF08D3128B44EEFBAF
Imphash	7B7E8665F226BBAB519D75C7E411307B
first-bytes-text	M Z .. .. @ .. ..
File-type	Executable
CPU	32-bit
Bahasa	-

Data yang didapatkan dari "pestudio" menunjukkan 4 jenis *hash* yang dimiliki oleh *malware*. Hasil identifikasi pada *hash* tersebut muncul pada pencarian *hash* SHA256, dimana terdapat nama dari *malware* yaitu *Backdoor* bernama *Slackbot*, dan dari hasil pencarian pada *tools VirusTotal* seperti pada Gambar 4.8 di bawah, 61 dari 72 antivirus mendeteksi *file* ini sebagai *malware*. *Malware* mempunyai tipe executable dan berjalan pada CPU 32-bit, namun tidak terdeteksi bahasa apa yang digunakan oleh mesin komputer saat mengkompile *malware* ini.



Gambar 4. 8 hasil dari identifikasi malware sampel *2 pada 'Virustotal'*

Sumber:
https://www.virustotal.com/gui/file/1a28360cf76d1fae638c6f28d08efb5fd24db9557f6725af08d3128b44eefbaf/detection

4.4.1.3 Analisis String

Analisa pada string memberikan banyak informasi-informasi tersembunyi dari sebuah aplikasi, pertama analisa menggunakan *tools* "Ghidra", berikut adalah string-string yang muncul sebagai *symbol* pada aplikasi "Ghidra". Pada aplikasi "Ghidra" *symbol* adalah string terbaca yang dianggap "Ghidra" merupakan hal yang memiliki relevansi untuk dicek, baik symbol itu berupa data, parameter ataupun function.

Pada sampel 1 *Beast Malware* (server.exe):

Pada sampel terdapat beberapa data dan function yang muncul pada aplikasi, berikut dijabarkan pada table 5:

Table 5 Penemuan Analisis String malware Beast

Symbol Reference pada Ghidra			Penjelasan
RegCloseKey	Global	0040339c	Terlihat function dan data untuk mengendalikan registry-registry pada komputer target mulai dari membaca, membuat, mengganti dan menghapus tidak hanya Key yang ada namun juga value dari registry tersebut. Menurut string yang muncul pada saat
RegCloseKey	ADVAPI32.DLL	External[000139f4]	
RegCreateKeyExA	ADVAPI32.DLL	External[000139e2]	
RegCreateKeyExA	Global	004033a4	
RegDeleteKeyA	Global	004033ac	
RegDeleteKeyA	ADVAPI32.DLL	External[000139d2]	
RegDeleteValueA	ADVAPI32.DLL	External[000139c0]	
RegDeleteValueA	Global	004033b4	
RegEnumKeyA	ADVAPI32.DLL	External[000139b2]	
RegEnumKeyA	Global	004033bc	
RegEnumValueA	ADVAPI32.DLL	External[000139a2]	
RegEnumValueA	Global	004033c4	
RegisterClassA	USER32.DLL	External[00013c76]	
RegisterClassA	Global	004036c4	
RegOpenKeyExA	Global	004033cc	
RegOpenKeyExA	ADVAPI32.DLL	External[00013992]	
RegQueryValueExA	ADVAPI32.DLL	External[00013980]	
RegQueryValueExA	Global	004033d4	
RegSetValueExA	ADVAPI32.DLL	External[00013970]	
RegSetValueExA	Global	004033dc	

004056e8 " NT\\CurrentVersion\\SystemRestore" 00405714 "DisableSR" 0040f96c "SOFTWARE\\Microsoft\\Windows" 0040f990 "\\CurrentVersion"	analisis *malware* mampu mengakses registry yang ada pada komputer target yaitu registry: "Computer\HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows NT\CurrentVersion\SystemRestore" dan menanam pengaturan "DisableSR" untuk mematikan kemampuan komputer untuk melakukan recovery system. System juga pada registry CurrentVersion. Registry ini adalah registry yang mengatur agar *malware* dapat berjalan kembali bahkan setelah mesin di reboot.
SetClipboardData USER32.DLL External[00013c56] SetClipboardData Global 004036d4 SetCursorPos Global 004036dc SetCursorPos USER32.DLL External[00013c48] GetClipboardData Global GetClipboardData USER32.DLL	Terlihat function dan data untuk mengendalikan *cursor* dan juga *clipboard* pada komputer target.
WriteFile Global 0040356c WriteFile KERNEL32.DLL External[000134a8] ReadFile Global 0040109c ReadFile KERNEL32.DLL External[0001385e] DeleteFileA KERNEL32.DLL External[00013778] DeleteFileA Global 0040340c	Function menunjukkan *malware* mempunyai akses untuk membaca, membuat, mengubah,

CreateFileA Global 004033fc CreateFileA KERNEL32.DLL External[00013796] 0040d29c " /c del \"%s\"" 0040f8dc ":\\Documents and Settings\\" 0040f900 "\\Local Settings\\Application Data\\Microsoft\\"	dan menghapus *file* pada komputer target. Salah satunya adalah menghapus *file* sampel saat terjadi pengeksekusian, *malware* juga melakuakn sesuatu pada *file* di path seperti pada digambar.
StartServiceA Global 00403da8 StartServiceA ADVAPI32.DLL External[00013a7e] OpenServiceA ADVAPI32.DLL External[00013a8e] OpenServiceA Global 00403da0 DeleteService ADVAPI32.DLL External[00013ad8] DeleteService Global 00403d80 ControlService Global 00403d78 ControlService ADVAPI32.DLL External[00013ae8]	Functions untuk mengendalikan service pada komputer target.
CreateProcessA Global 00403404 CreateProcessA KERNEL32.DLL External[00013786] ExitProcess KERNEL32.DLL External[00013758] ExitProcess KERNEL32.DLL External[000137f2] ExitProcess Global 0040341c ExitProcess Global 004010dc OpenProcess Global 004034f4 OpenProcess KERNEL32.DLL External[0001359a] TerminateProcess KERNEL32.DLL External[000134dc] TerminateProcess Global 00403554 00004A44 GetScreen 00004A50 GetWebCam 00004A9C \shell\open\command 00004AB8 Explorer.exe	Functions untuk mengendalikan proses (aplikasi) yang sedang berjalan dalam komputer, dapat dilihat *malware* memulai proses untuk memanggil explorer.exe dan mengakses layar dan webcam komputer korban

Pada sampel 2 *Slackbot* (tnnbtib.exe) hasil Analisa *string* dijabarkan pada tabel 6:

Table 6 Penemuan Analisis String malware Slackbot

Symbols Reference pada Ghidra	Penjelasan
RegCloseKey ADVAPI32.DLL External[0000665c] RegCloseKey Global 00403530 RegCreateKeyExA ADVAPI32.DLL External[0000666c] RegCreateKeyExA Global 0040353c RegDeleteValueA Global 00403548 RegDeleteValueA ADVAPI32.DLL External[00006680] RegOpenKeyExA ADVAPI32.DLL External[00006694] RegOpenKeyExA Global 00403554 RegSetValueExA ADVAPI32.DLL External[000066a4] RegSetValueExA Global 00403560 0040568a Update 00405691 SOFTWARE\Microsoft\Windows\CurrentVersion\Run	Function-function untuk mengendalikan registry pada komputer korban. Salah satunya adalah membuat registry yang membuat *backdoor* agar *malware* dapat berjalan kembali setelah komputer melakukan reboot
InternetCanonicalizeUrlA WININET.DLL External[000063e4] InternetCanonicalizeUrlA Global 00403314 InternetCloseHandle Global 00403320 InternetCloseHandle WININET.DLL External[00006400] InternetOpenA Global 0040332c InternetOpenA WININET.DLL External[00006418] InternetOpenUrlA Global 00403338 InternetOpenUrlA WININET.DLL External[00006428] InternetReadFile Global 00403344 InternetReadFile WININET.DLL External[0000643c]	Function untuk membuka sumber lengkap dari FTP dan HTTP URL dan membaca *file* tersebut
ExitWindowsEx USER32.DLL External[00006608] ExitWindowsEx Global 00403500	Function ini berguna untuk

	log off user yang sedang berjalan, atau bahkan meng-restart serta mematikan mesin secara menyeluruh.
ReadFile KERNEL32.DLL External[0000656c] ReadFile Global 00403488	Function untuk membaca *file* di mesin korban

4.4.2 Analisis Dinamis

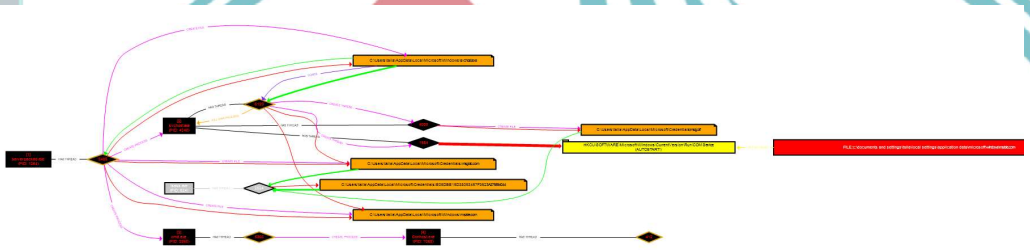
Dalam skenario analisa ini sebelum menjalankan *malware* yang sudah dikirim oleh PC Server, PC Target akan menjalankan aplikasi "ProcMon", "Procces Hakcer" untuk menangkap aktifitas *malware* dalam PC serta "cmd" untuk menangkap aktifitas TCP dan UDP pada jaringan yang digunakan, proses ini adalah proses ***dynamic analysis.***

Pada sampel 1 *malware Beast,* proses malware bisa dilihat pada Gambar 4.9:

Gambar 4. 9 process malware 1 terekam pada tools Process Hacker

Saat *malware* dijalankan di komputer target muncul proses baru bernama "svchost.exe" dengan PID 5896 pada aplikasi *Process* Hacker, nomor PID tidaklah mutlak dan bisa berubah disetiap eksekusinya, namun hal yang pasti adalah nama dari proses dan warna yang diberikan aplikasi kepada proses, warna ungu menunjukkan bahwa proses memiliki konten yang terbungkus (*packed)* dimana hal ini adalah salah satu ciri-ciri *malware*, maka dengan informasi ini target/korban harus meneliti lebih lanjut setiap aktifitas dari aplikasi ini.

Kemudian beralih pada aplikasi ProcMon, aktifitas aplikasi bisa dimonitor menggunakan aplikasi ProcMon (*Process* Monitor), data pada ProcMon masih berantakan dan banyak karena bercampur dengan proses-prosees lainnya yang sedang berjalan pada mesin, sehingga untuk membaca data-data berikut dengan lebih mudah digunakan satu *tools* pembantu lagi bernama "ProcDOT". Untuk menggunakan aplikasi ini, data dari ProcMon harus dikonversi ke format yang bisa dibaca oleh ProcDOT. Berikut adalah tampilan grafik dari data yang sudah dikonversi, ditampilkan di aplikasi ProcDOT bisa dilihat pada Gambar 4.10 di bawah:



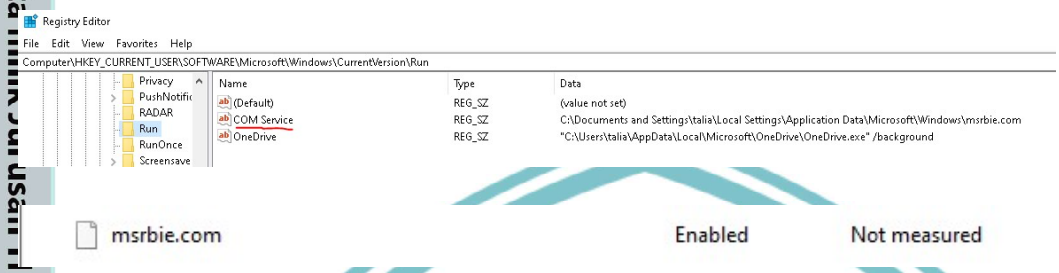
Gambar 4. 10 diagram kegiatan yang dilakukan malware Beast secara keseluruhan

Berikut adalah seluruh diagram yang dari analisis dinamis, menunjukkan proses yang berjalan di belakang maupun pada layar, *malware* mengakses banyak registry pada komputer target, salah satu dari registry yang diakses oleh *malware* adalah, pembuatan registry baru pada komputer target, yaitu registry untuk mempertahankan koneksi dan akses walaupun komputer telah di-*reboot.* Berikut tampilan aktifitas tersebut pada grafik yang dibuat oleh ProcDOT pada Gambar 4.11 di bawah:

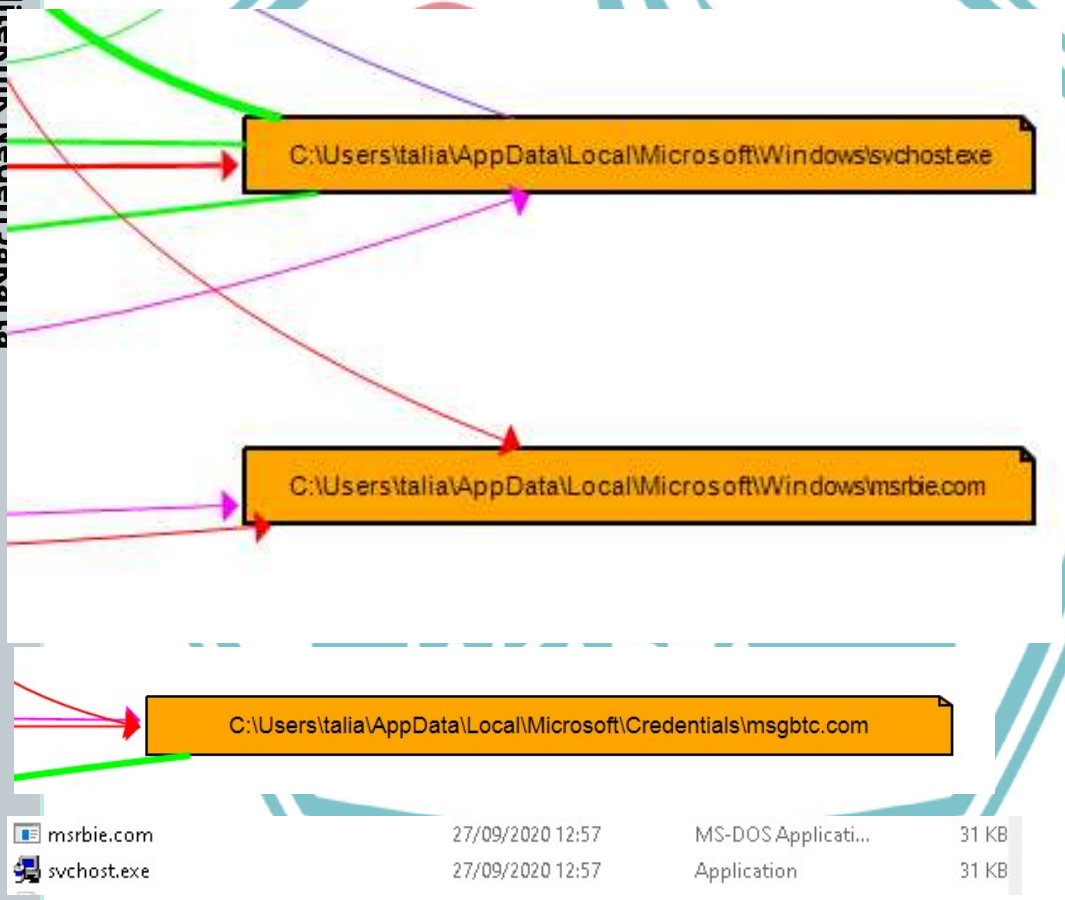
Gambar 4. 11 malware membuat registry persistance berserta value

Registry yang terletak di "HKCU/SOFTWARE/Microsoft/Windows/CurrentVersion/Run/COM Service" dengan isi (*value) file* path "c:\documents and settings\application data\microsoft\windows\mrsbie.com" ini mengkonfigurasi komputer untuk menjalankan *malware* setiap kali komputer dinyalakan, hal ini dapat dilihat pada

"Task Manager" dimana pada tab Startup terdapat aplikasi yang memiliki nama seperti value pada registry si *malware* dijalankan setiap kali komputer dinyalakan:



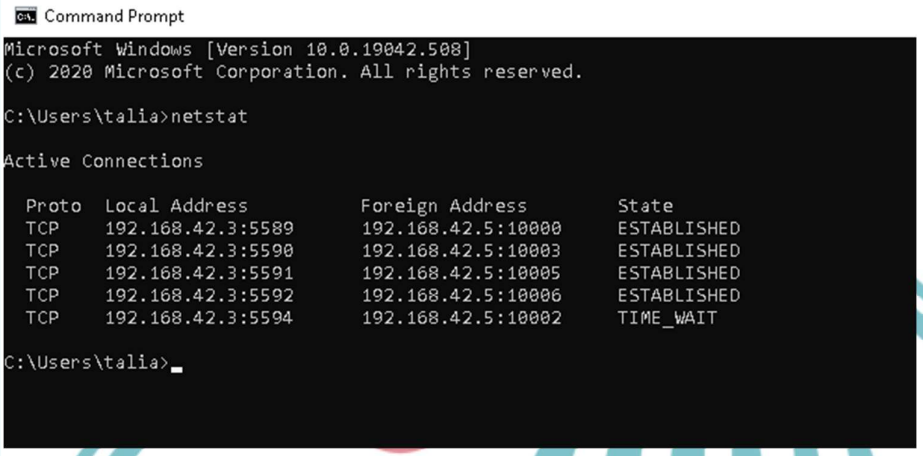
Gambar 4. 12 registry persistance dilihat dari 'registry editor' dan 'task namager'



Gambar 4. 13 file persistance yang ditanam oleh malware tercatat oleh 'ProcDOT' dan dilihat di 'windows file explorer'

Terlihat pada Gambar 4.12 dan Gambar 4.13 aktifitas lain dari *malware* adalah menanam *file* di mesin target, menunjukkan bahwa *malware* memiliki akses terhadap *file-file* di mesin target. *File* svchost.exe adalah *file* utama sang *malware*,

sedangkan *file* msrbie.com adalah *file* yang membantu penyerang mempertahankan akses dan koneksi jaringan saat mesin target pertama kali dinyalakan.

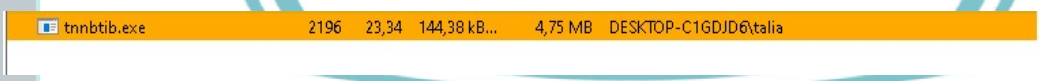


Gambar 4. 14 netstat menunjukkan sambungan pada komputer target

Gambar 4.14 adalah hasil dari analisa jaringan saat dijalankan *command* "netstat" pada "cmd" dapat terlihat *local address* adalah *address* dari mesin milik mesin target (sendiri) dimana command dijalankan dan ip dari foreign address menyatakan ip address perangkat yang tersambung dengan komputer target. Alamat IP 192.168.42.5 adalah alamat IP milik komputer penyerang.

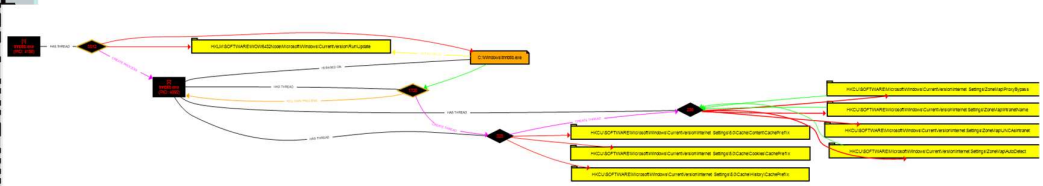
Pada sampel 2 *Slackbot* (tnnbtib.exe) bisa dilihat pada Gambar 4.15:

Pada saat sampel dijalankan, prosesnya tercatat oleh *tools* "*Process* Hacker" dengan PID 4772 dan berwarna oranye yang menunjukkan bahwa proses ini adalah proses yang berjalan menggunakan menggunakan UAC (user account control) dan hak penuh terhadap sistem.



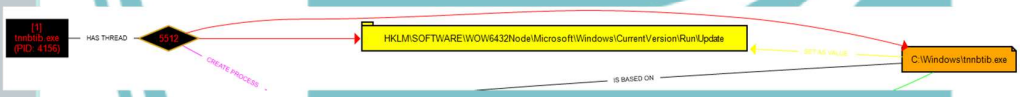
Gambar 4. 15 proses malware berjalan pada mesin tercatat oleh 'process hacker'

Aktifitas yang dilakukan oleh sang *malware* juga dicatat oleh "ProcMon" dan data dari *tools* ini diolah oleh "ProcDOT" sehingga didapatkan grafik aktifitas-aktifitas yang dilakukan oleh *malware*, sebagai berikut:



Gambar 4. 16 gambar keseluruhan grafik aktifitas malware di mesin target

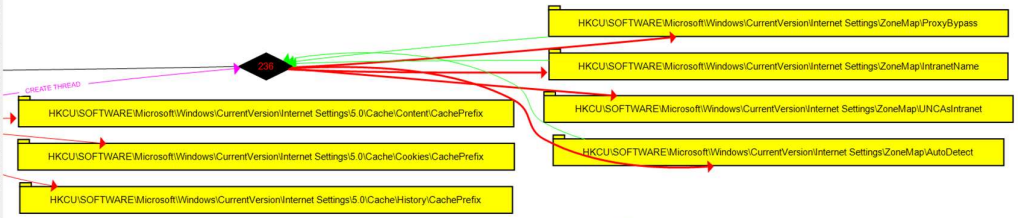
Terlihat pada Gambar 4.16 di atas adalah aktifitas yang dilakukan *malware* secara keseluruhan. Langkah pertama yang dilakukan oleh *malware* adalah menanam registry baru pada lokasi "HKLM\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Update" dengan *value* "C:\Windows\tnnbtib.exe" arti dari registry ini adalah untuk memberikan instruksi kepada windows agar menjalankan *file malware* tnnbtib.exe setiap kali komputer menayala agar penyerang dapat mempertahankan koneksi, lebih jelasnya pada Gambar 4.17 adalah gambar grafik dari *tools* pemantau ProcDOT, dan Gambar 4.18 adalah aplikasi *malware* yang muncul pada *tab Startup* di *Windos Task Manager*:



Gambar 4. 17 membuat registry dengan untuk mempertahankan koneksi

Gambar 4. 18 hasil konfigurasi registry dapat diliat pada menu startup di task manager

Aktifitas lain yang dilakukan oleh *malware* sampel 2 terpantau oleh *tools* "ProcMon" dan diolah oleh *tools* "ProcDOT" terlihat seperti grafik di Gambar 4.19 di bawah, adalah membaca registry-registry yang bersangkutan dengan internet dimana sang *malware* mengakses informasi yang tersimpan di mesin *(cache)* dan membaca informasi dari *content, cookies,* dan *history* di Internet Explorer untuk mencari informasi autentikasi penting yang bisa dicuri seperti *password,* id, identitas atau nomor kartu kredit.



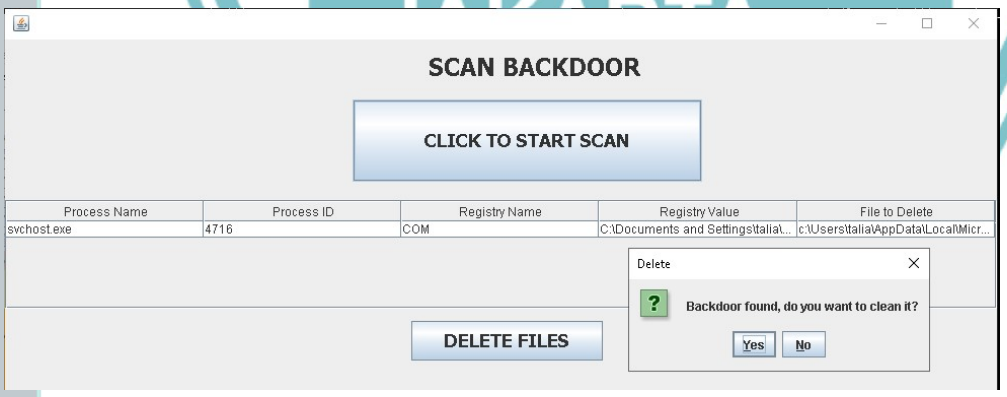
Gambar 4. 19 registry-registry internet yang diakses oleh sang malware

Pada *malware Slackbot* netstat menunjukkan hasil kosong karena virtualbox tidak tersambung dengan jaringan internet asli.

4.4.3 Pengujian Aplikasi

Untuk memastikan aplikasi telah sukses mencapai tujuan pembuatannya, digunakan *tools* "Regshot" untuk membandingkan keadaan mesin sebelum dan sesudah aplikasi ini digunakan. Berikut adalah tampilan saat aplikasi sudah menemukan matriks yang ditinggalkan oleh sampel 1 *malware Beast* (svchost.exe), pada kolom pertama adalah nama proses dari malware yang sedang berjalan, di kolom ke-2 adalah *Process ID* (PID) dari proses *malware* yang sedang berjalan, kemudian pada kolom ke-3 adalah nama registry yang ditanam dan kolom ke-4 adalah value dari registry tersebut, terakhir pada kolom ke-5 adalah *filepath* dari *file* yang ditanam oleh *malware*.

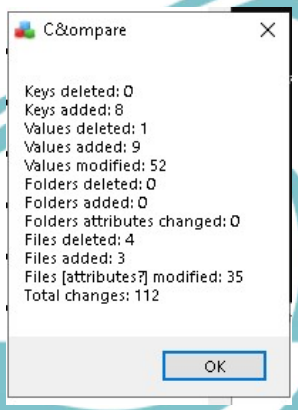
Hasil yang didapatkan pada *malware* sampel *1 Beast* dapat dilihat di Gambar 4.20 di bawah:



Gambar 4. 20 penampilan aplikasi menemukan malware Beast

Malware dapat menemukan *malware Beast* yang sedang berjalan mesin korban dengan nama proses svchost.exe dan PID 5896, kemudian aplikasi menemukan

matriks selanjutnya yaitu registry yang dibuat oleh *Beast* untuk mempertahankan koneksi dengan nama registry COM dan *filepath* aplikasi yang dijalankan setiap mesin menyala di kolom selanjutnya. Ke 5 (lima) matriks tersebut dihapus oleh aplikasi. Dibawah adalah perbandingan keadaan mesin sebelum dan sesudah aplikasi dijalankan dapat dilihat pada Gambar 4.21 di bawah:

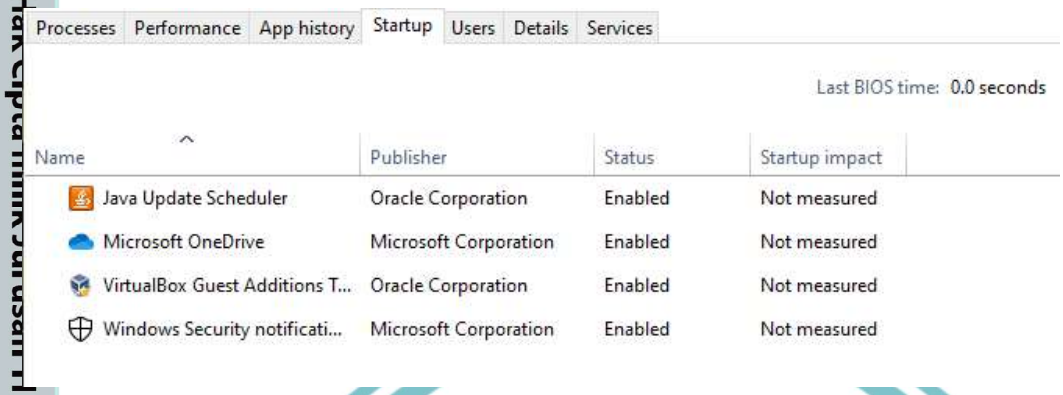


Gambar 4. 21 hasil komparasi sebelum dan sesudah aplikasi pembersih malware Beast dieksekusi oleh 'regshot'

Dari semua perubahan yang terjadi pada mesin, salah satu values dan *file* yang dihapus oleh aplikasi adalah values dan *file* yang ditanam oleh *malware*, 3 dari 4 *file* yang terhapus adalah *file* yang ditanam oleh *malware,* secara lebih rincinya dapat dilihat pada Gambar 4.22 di bawah:

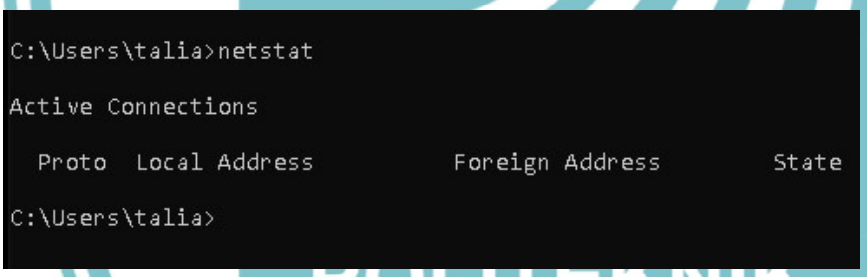
```
----------------------------------
Files deleted: 4
----------------------------------
C:\Users\talia\AppData\Local\Microsoft\Credentials\msgbtc.com
2020-09-27 19:57:19, 0x00000004, 30883
C:\Users\talia\AppData\Local\Microsoft\Windows\msrbie.com
2020-09-27 19:57:19, 0x00000004, 30883
C:\Users\talia\AppData\Local\Microsoft\Windows\svchost.exe
2020-09-27 19:57:19, 0x00000004, 30883

Values deleted: 1
----------------------------------
HKU\S-1-5-21-570714837-2217981920-2020154653-1001\SOFTWARE\Microsoft\Windows\CurrentVersion\Run\COM Service:
"C:\Documents and Settings\talia\Local Settings\Application Data\Microsoft\Windows\msrbie.com"
```

Gambar 4. 22 detail catatan regshot, salah satu value registry dan file yang terhapus adalah value yang ditanam oleh malware Beast



Gambar 4. 23 aplikasi sudah tidak lagi muncul pada tab Startup di 'windows task manager'

Gambar 4.23 menunjukkan *tab Startup* pada *Windows Task Manager* sudah tidak ada lagi aplikasi yang ditanam oleh *malware.* Dan pada Gambar 4.24 *command 'netstat'* menunjukkan bahwa komputer sudah tidak lagi terhubung dengan penyerang.

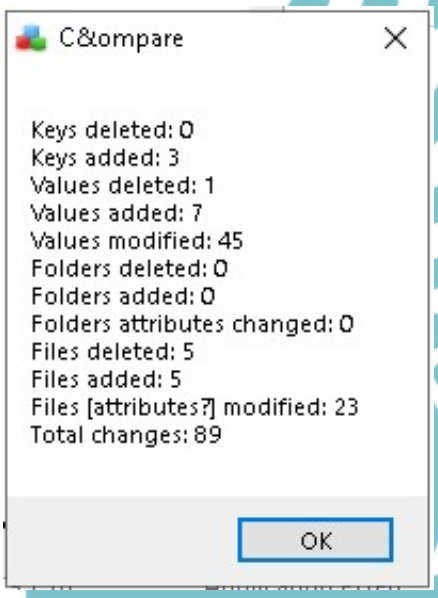


Gambar 4. 24 netstat pada cmd menunjukkan mesin tidak terhubung lagi dengan penyerang

Hasil yang didapatkan pada *malware* sampel *2 Slackbot* dapat dilihat pada Gambar 4.25*:*

Gambar 4. 25 penamilpilan aplikasi menemukan malware Slackbot

Pada tampilan ini aplikasi tidak bisa menemukan *process* ID dari *malware* karena proses tersebut sudah dihapus terlebih dahulu oleh aplikasi tanpa aplikasi mencari secara *real-time,* hal ini dilakukan karena keterbatasan sumber mesin di *virtualBox*, jika aplikasi mencoba untuk mencari *process malware* seperti pada metode pertama, maka mesin *virtualBox* akan mengalami eror dan pencarian data tidak akan selesai, untuk menangani masalah ini maka metode pembunuhan proses yang digunakan pada sampel *malware* ke-2 sedikit berbeda, aplikasi membunuh *process* dengan menggunakan nama *process* yang unik, parameter ini sudah di-*input* pada saat pembuatan aplikasi sehingga aplikasi tidak perlu melakukan proses pencarian lagi. Dengan begini barulah aplikasi dapat mencari data *registry* dan *file* yang ditanam oleh *malware* dengan lancar.

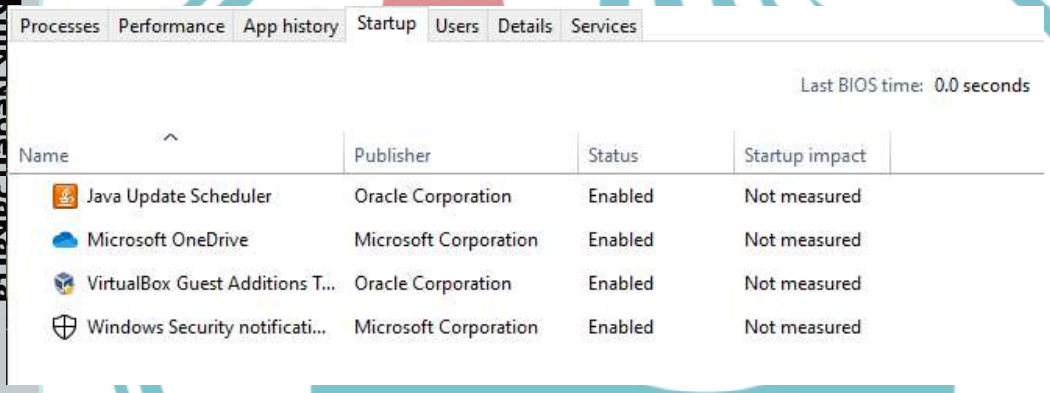


Gambar 4. 26 hasil komparasi sebelum dan sesudah aplikasi pembersih malware Slackbot *dieksekusi oleh 'regshot'*

Gambar 4.26 adalah hasil perbandingan kondisi komputer sebelum dan sesudah aplikasi penghapus *malware* dijalankan. Menunjukkan 1 dari 5 *file* yang terhapus pada jangka waktu sebelum dan sesudah aplikasi penghapus *malware* dieksekusi adalah *file* yang ditanam oleh malware *Slackbot.* Secara lebih rinci dapat dilihat pada Gambar 4.27 di bawah:

```
---------------------------------
Files deleted: 5
---------------------------------
C:\Windows\tnnbtib.exe
2002-05-22 06:59:28, 0x00000020, 8329

---------------------------------
Values deleted: 1
---------------------------------
HKLM\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Run
\Update: "C:\Windows\tnnbtib.exe"
```

Gambar 4. 27 detail catatan regshot, salah satu value registry dan file yang terhapus adalah value yang ditanam oleh malware Slackbot



Gambar 4. 28 aplikasi sudah tidak lagi muncul pada tab Startup di 'windows task manager'

Keberhasilan aplikasi menghapus *malware* sampel 2 ini juga dapat dilihat dari *tab Startup* di *Windows Task Manager*, terlampir pada Gambar 4.28 di atas bahwa aplikasi yang ditanam oleh *malware* untuk berjalan saat komputer pertama dihidupkan sudah tidak ada, sehingga dapat disimpulkan penyerang sudah tidak lagi memiliki *backdoor* untuk terhubung ke komputer korban secara diam-diam.

4.5 Analisis Data/Evaluasi

4.5.1 Perbandingan Data Hasil Analisis Statis dan Dinamis

Perbandingan data dari hasil analisis statis dan dinamis pada sampel *malware* 1 *Beast* (server.exe) dapat dilihat pada tabel 7 di bawah:

Table 7 table kumpulan penemuan analisis statis dan dinamis pada malware 1 Beast

No	Temuan	Analisis Statis	Analisis Dinamis
1.	Menemukan identitas *malware*	✓	-
2.	Penambahan Registry	✓	✓
3.	Penambahan Registry: "HKCU/SOFTWARE/Microsoft/Windows/CurrentVersion/Run/COM Service" dengan isi *(value) file* path "c:\documents and settings\application data\microsoft\windows\mrsbie.com"	-	✓
4.	Membaca Registry	✓	✓
5.	Penghapusan Registry	✓	-
6.	Mengubah isi Registry	✓	-
7.	Membuat Proses Baru	✓	✓
8.	Membunuh Proses	✓	
9.	Membuat *File* Baru: svchost.exe; mrsbie.com	-	✓
10	Membaca *file*	✓	-
11.	Menghapus *File*	✓	-
12.	Mengatur Service	✓	-
13.	Mengatur Clipboard	✓	-
14.	Mengatur posisi cursor	✓	-
15.	Mencari tahu IP penyerang	-	✓

Selanjutnya, pada sampel *malware* 2 *Slackbot* (tnnbtib.exe) dapat dilihat pada tabel 8 di bawah:

Table 8 table kumpulan penemuan analisis statis dan dinamis pada malware 2 Slackbot

No	Temuan	Analisis Statis	Analisis Dinamis
1.	Mengetahui identitas *malware*	✓	-
2.	Penambahan Registry: "HKLM\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Update" dengan *value* "C:\Windows\tnnbtib.exe"	✓	✓
3.	Penghapusan Registry	✓	-
4.	Mengubah isi registry	✓	-
5.	Membaca registry	✓	✓
6.	Membaca *File*	✓	-
7.	Mengakses pengaturan Internet	✓	✓
8.	Membuat *file*: c:\windows\tnnbtib.exe	-	✓
9.	Mencari tahu IP penyerang	-	-

Dapat dilihat perbandingan informasi yang didapatkan dari metode analisis statis dan dinamis. Metode analisis statis bagus digunakan untuk mengetahui identitas dan kemampuan sebuah *malware* namun terkadang informasi yang didapat dari analisis ini tidak cukup untuk melakukan pembersihan secara langsung karena terkadang informasi pada String analisis statis hanya terbaca setengah.

Dengan menggabungkan kedua metode didapatkan informasi yang lebih lengkap tentang identitas, kemampuan, perilaku dan efek yang disebabkan oleh *malware* pada mesin yang terinfeksi.

4.5.2 Analisis hasil kerja aplikasi pembersih *malware*

Pengujian ini dilakukan untuk membuktikan bahwa aplikasi yang dibangun sudah berhasil menghapus malware dari mesin yang terinfeksi, hasilnya dapat dilihat pada table berikut:

Table 9 hasil pengujian aplikasi penghapus backdoor

Malware	Matriks yang harus dihapus	Berhasil dihapus dari mesin
Beast (server.exe)	***Process*: *process* svchost.exe	berhasil
	Registry: "HKCU/SOFTWARE/Microsoft/Windows/CurrentVersion/Run/COM Service" dengan isi (*value) file* path "c:\documents and settings\application data\microsoft\windows\mrsbie.com"	berhasil
	***File*: svchost.exe; msribe.exe	berhasil
Slackbot (tnnbtib.exe)	***Process*: *process* tnnbtib.exe	berhasil
	Registry: "HKLM\SOFTWARE\WOW6432Node\Microsoft\Windows\CurrentVersion\Update" dengan *value* "C:\Windows\tnnbtib.exe"	berhasil
	***File*: tnnbtib.exe	berhasil

Berdasarkan table 9, hasil dari pengujian aplikasi penghapus *backdoor*, aplikasi berhasil menghapus *backdoor* dari mesin yang terinfeksi dan memastikan bahwa *backdoor* tersebut tidak bisa lagi tersambung (mengakses) mesin karena pintu belakang celah penyerang mengambil akses sudah dihapus.