JURNAL REKAYASA SISTEM DAN INDUSTRI
e-ISSN: 2579-9142 p-ISSN: 2356-0843
http://jrsi.sie.telkomuniversity.ac.id

Analisis *Malware* Berdasarkan *API CALL MEMORY* Dengan Metode Deteksi *Signature-Based*

Analysis Malware Based on Call Memory API withh Signature-Based Detection Method

Julian Dwi Nugraha, *[1], Avon Budiono [2] , Ahmad Almaarif [3]

[1, 2, 3]Program Studi Sistem Informasi, Fakultas Rekayasa Industri, Telkom University

ARTICLE INFO

Article history:
Diterima 10-11-2019
Diperbaiki 13-12-2019
Disetujui 29-12-2019

Kata Kunci:
malware, analisis malware, static analysis, dynamic analysis, signature-based.

ABSTRAK

Malware merupakan sebuah perangkat lunak atau program komputer yang digunakan untuk melakukan tindakan kejahatan. *Malware* pada dasarnya dirancang untuk menginfeksi sistem komputer penguna tanpa persetujuan pemiliknya. *Trojan*, *Worms*, *Virus*, *Spyware*, dan *Keylogger* adalah kategori *malware* yang dapat merugikan pengguna yang telah terinfeksi. Berdasarkan hal tersebut maka dari itu diperlukan *analisis malware* menggunakan API *call memory* dengan metode *signature-based detection*. *Signature based detection* adalah teknik deteksi yang berdasarkan *pattern matching*, *string*, *mask*, atau teknik *fingerprinting*. *Signature* adalah teknik persamaan *bit* yang disuntikkan dalam program aplikasi oleh *attacker*, yang secara unik mengidentifikasi jenis *malware* tertentu. Hal ini digunakan dengan tujuan untuk mengidentifikasi *malware* tersebut menggandung program yang dapat mengambil data pengguna tanpa sepengetahuan pengguna itu sendiri. Maka dari itu di dalam penelitian ini dilakukan *analisis malware* menggunakan sebanyak 30 *malware* untuk melihat jenis API *call* yang digunakan oleh *malware* tersebut. Pada penelitian ini berfokus untuk melakukan analisis pada API *Memory* yang telah didapatkan. Dari semua *malware* menjalankan satu API *memory* yang sama ketika dijalankan pertama kali. Hasil pada penelitian ini melihat API *call memory* dan hasil *signature* yang telah dilakukan menggunakan metode *signature based detection* dan melihat keterkaitan antara API *call memory* dengan hasil *signature* pada setiap *malware.*

Keywords:
malware, analisis malware, static analysis, dynamic analysis, signature-based.

ABSTRACT

Malware is a software or computer program that is used to commit a crime. *Malware* is basically designed to infect user computer systems without the owner's consent. Trojans, Worms, Viruses, Spyware, and Keyloggers are categories of *malware* that can harm infected users. Based on this, the analysis *malware* is used using the API call memory with the signature-based detection method. Signature-based detection is a detection technique based on pattern matching, strings, masks, or fingerprinting techniques. Signature is a bit equation technique that is injected into an application program by an attacker, which uniquely identifies certain types of *malware*. It is used with the aim of identifying the *malware* containing a program that can retrieve user data without the user's own knowledge. Therefore, in this research, analisis *malware* uses as many as 30 *malware* to see the type of API call used by the *malware*. This research focuses on analyzing the obtained API Memory. Of all *malware* will run the same API memory when running the first time. The results of this study are to see the call memory API and the results of the signatures that have been done using the signature-based detection method and see the relationship between the call memory API and the results of the signatures on each *malware*.

1. Pendahuluan

Saat ini perkembangan teknologi semakin pesat. Dengan semakin meningkatnya pengetahuan masyarakat mengenai teknologi informasi dan komunikasi, serta adanya sifat murni manusia yang selalu tidak pernah merasa puas, tentu saja hal ini lama kelamaan, membawa banyak dampak positif maupun negatif. [1]. *Cyber-crime* adalah istilah yang mengacu kepada aktivitas kejahatan dengan komputer atau jaringan komputer menjadi alat, sasaran atau tempat terjadinya kejahatan. *Cyber-*

Email: juldwinug@gmail.com (Julian Dwi Nugraha)*
DOI: https://doi.org/10.25124/jrsi.v6i02.351

*Penulis korespondensi

crime sebagai tindak kejahatan dimana dalam hal ini penggunaan komputer secara illegal [2]. *Malware* adalah salah satu cara menginfeksi pengguna untuk mendapatkan hak akses sistem komputer pengguna tanpa persetujuan miliknya. Hal ini dimaksudkan untuk mengumpulkan informasi pribadi tanpa mengambil persetujuan dari pemilik sistem. Sehingga menciptakan ancaman bagi privasi penggunanya [3].

Salah satu cara untuk megantisipasinya kita dapat melakukan analisis *malware.* Analisis *malware* adalah kumpulan dari proses penentuan tujuan dan fungsionalitas dari *sample malware* yang diberikan seperti *virus, worm, trojan,* dan sebagainya untuk melakukan deteksi *malicious code.* Baik dengan mengeksekusi *malware* tersebut (*dynamic analysis)* ataupun dengan menginspeksi kode program saat sebelum dieksekusi (*static analysis*) [4]. Teknik analisis *malware* dapat membantu penelitian dalam memahami risiko dan bahaya dari *malware* tersebut. salah satu cara metode yang dapat digunakan ada dengan *signature-based detection. Signature based detection technique* juga dikenal sebagai pencocokan pola atau string atau teknik sidik jari. *Signature* adalah urutan bit yang disuntikkan dalam program aplikasi oleh penulis *malware*, yang secara unik mengidentifikasi jenis *malware* tertentu.

Untuk mendeteksi *malware* dalam kode, detektor *malware* akan melakukannya mencari *signature* yang ditentukan sebelumnya dalam kode [5]. *Application Programming interface* (API) adalah bagian yang tak terelakkan sistem operasi Windows. Semua program di Windows dapat berinteraksi dengan Windows API dan mengakses yang telah ditentukan fungsi dengan membuat panggilan API untuk memanfaatkan layanan disediakan olehnya seperti layanan dasar untuk mengakses sumber daya seperti *file* sistem, proses, dan perangkat [6]. API *call* ialah suatu prosedur, protokol dan alat untuk membangun suatu aplikasi. Informasi dari API *call* akan digunakan untuk mengetahui aktivitas dari *malware*, sehingga informasi aktivitas *malware* akan diklasifikasikan menggunakan teknik *signature based detection* [7].

Layanan lanjutan untuk mengakses *registry* Windows, layanan jendela *shell*, jaringan layanan dll. Setelah dilakukan kategorisasi, dapat disimpulkan *impact* sampel *malware* terhadap performansi suatu komputer yang terinfeksi. Keuntungan dari penerapan metode *signature based* adalah deteksi suatu *signature* akan lebih mudah dan lebih cepat ditemukan [6]. Oleh karena itu, untuk melakukan klasifikasi jenis *malware* penulis melakukan *analysis malware* dengan menggunakan metode *signature-based detection*. Berdasarkan hal tersebut maka hasil akhir dari penelitian ini berupa *analysis malware* berdasarkan hasil API *call memory* dengan hasil *signature*.

2. Studi Literatur

2.1 Malicious Software (Malware)

Malware adalah sebuah *software* yang dirancang dengan tujuan untuk membahayakan, menyusup, atau merusak sebuah komputer. *Malware* juga biasa didefinisikan sebagai kode berbahaya [8]. Sebagian besar *malware* memasuki komputer saat mengunduh *file* dari *Internet*. Setelah perangkat lunak berbahaya masuk ke dalam sistem, ia memindai kerentanan yang ada di sistem operasi dan melakukan tindakan yang tidak diinginkan dan tidak sah pada sistem *file* sehingga memperlambat kinerja sistem [9]. Semua *attacker* yang terlibat dapat menghasilkan uang dari pengguna yang telah terinfeksi oleh *malware*. Dengan naiknya pengguna *internet* dan jumlah *host* yang terhubung, saat ini lah yang ditunggu oleh para *attacker* atau penjahat *cyber* untuk menginfeksi banyak host setelah merilis sebuah *malware*. Dengan demikian *malware* dapat menjadi ancaman paling berbahaya bagi pengguna sistem komputer [10]**.**

2.2 Jenis-Jenis Malware

Berbagai macam *malware* dibuat untuk menginfeksi sistem komputer pengguna. Berikut beberapa jenis-jenisnya [10].

1. *Virus*

   Virus adalah sejenis *malware* yang menanamkan salinan programnya sendiri ke dalam perangkat pengguna yang terkena dampaknya dan dapat mempengaruhi kinerja dari sistem yang telah terinfeksi. Virus tidak lain hanyalah kode atau program menular yang menempel pada *software* lain dan memerlukan interaksi penggunanya agar virus tersebut dapat menjalankan kode atau program yang telah ditanamkan didalamnya.

2. *Worm*

   Worm adalah sebuah *software* mandiri yang bereplikasi tanpa menargetkan dan menginfeksi *file* tertentu yang sudah ada di komputer. Dan juga dapat menghancurkan data-data yang ada di dalamnya.

3. *Trojan*

   Trojan adalah satu jenis *malware* yang menyamar atau berperilaku sebagai program yang bermanfaat dengan tujuan memberikan akses yang tidak sah kepada *attacker* ke sistem komputer pengguna. Trojan sekarang dianggap sebagai *malware* paling berbahaya, terutama trojan yang dirancang untuk mencuri informasi finansial pemilik komputer tersebut.

4. *Adware*

   Adware adalah semacam perangkat lunak berbahaya yang berusaha menawarkan sesuatu kepada pengguna yang akibatnya muncul sebagai jendela *pop-up* bahkan jika pengguna buka biasanya *adware* masuk

ke sistem dalam bentuk perjudian, iklan, atau jendela *pop-up* lainnya.

5. *Ransomware*

   Ransomware adalah salah satu *malware* yang paling *advance* dan paling berbahaya. *Ransomware* memblokir akses ke data korban, agar bisa mengancam untuk mempublikasikannya atau menghapusnya sampai uang tebusan yang tidak sedikit. Jenis *malware* ini pada dasarnya menginfeksi sistem dari dalam, mengunci komputer dan membuatnya tidak berguna.

6. *Spyware*

   Spyware adalah *software* yang selalu memata-matai penggunanya. Tujuannya untuk melacak aktivitas *internet* anda agar bisa dikirim sebuah *adware* yang digunakan untuk mengumpulkan informasi tentang data-data kita dan mengirimkan informasi tersebut ke entitas lain, tanpa persetujuan dari pemilik data tersebut.

7. *Backdoor*

   Backdoor adalah perangkat lunak yang memungkinkan *hacker* atau *attacker* untuk mengakses sistem tanpa menggunakan nama pengguna, kata sandi, atau teknik lainnya yang masuk ke dalam sistem yang digunakan *hacker* atau *attacker* untuk mendapatkan akses ke sistem atau data.

8. *Rootkits*

   Rootkits adalah sebuah koleksi *software* yang dirancang khusus untuk memperbolehkan *malware* untuk mengumpulkan informasi. Jenis *malware* ini bekerja dibalik layar sehingga pengguna komputer tidak akan curiga.

9. *Keyloggers*

   Keyloggers adalah *software* yang menyimpan semua informasi yang diketik dengan menggunakan *keyboard*. Keyloggers mengumpulkan informasi dan kemudian mengirimnya kepada *attackers* atau *hacker.*

10. *Browser Hijacker*

    Browser hijacker adalah sebuah *software* yang tidak diinginkan yang mengubah pengaturan *browser web* tanpa ijin penggunanya. *Malware* ini menempatkan iklan yang tidak diinginkan ke *browser* dan mungkin perubahan *homepage* atau *searchpage* menjadi halaman hijacker, *browser hijacker* juga mungkin mengandung *spyware* untuk mendapatkan informasi bank dan data sensitif lainnya.

11. *Rogue Security Software*

    Rogue security software atau juga sering disebut sebagai *rogue antimalware* adalah sebuah *software* yang terlihat seperti *software* bermanfaat dari sisi keamanan, padahal tidak. Tujuannya adalah untuk menipu pengguna agar percaya bahwa komputer anda terinfeksi dengan beberapa ancaman serius, dan kemudian menipu penggunanya untuk meng*install* / membeli *software* keamanan palsu.

2.3 Static Analysis

Static analysis 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. *Static analysis* memiliki teknik analisis yaitu *signature based* dan *heuristic* [11].

1. *Signature Based Detection*

   Signature based detection adalah teknik deteksi yang berdasarkan *pattern matching*, *string*, *mask*, atau teknik *fingerprinting*. *Signature* adalah teknik persamaan *bit* yang disuntikkan dalam program aplikasi oleh *attacker*, yang secara unik mengidentifikasi jenis *malware* tertentu. Untuk mendeteksi *malware* dalam kode, detektor *malware* akan melakukan *signature* yang telah ditentukan di dalam kode tersebut [5]. Meskipun *signature based detection* sangat efisien untuk *malware* yang telah dikenali, akan tetapi mempunyai kelemahan yaitu tidak dapat mendeteksi jenis *malware* yang tidak dikenal atau baru. Karena basis data *signature* yang terbatas, sebagian besar *malware* tetap tidak akan terdeteksi. Jadi varian *malware* harus diperbahurui saat akan dideteksi untuk bisa mendapatkan *signature* dari basis datanya [12]. Berikut merupakan kelebihan dan kekurangan dari *signature-based detection* [13-17]:

   - Kelebihan teknik *signature-based detection* adalah:
     a. Waktu yang diperlukan untuk pemindaian lebih sedikit.
     b. Gampang untuk diimplementasikan, karena tinggal menggunakan *database signature* yang tersedia.
     c. Sedikit menghasilkan *false-positive.*
     d. Cepat untuk mendeteksi serangan yang sama dikemudian hari dan menghasilkan *true-positive* yang lebih besar.
   - Kekurangan teknik *signature-based detection* adalah:
     a. *Malware* yang tidak dikenal dapat dengan mudah menghindarinya contohnya serangan yang bersifat *zero-day attack*. Karena tidak dapat mendeteksi intruksi yang tidak ada di *database signature.*
     b. Beberapa *false-positive* tidak dapat menangani *obfuscation*.
     c. Melakukan *packet analysis* dengan *size* yang besar dan banyak memerlukan *resource* yang besar.

2.4 API Calls

API *call* ialah suatu prosedur, protokol dan alat untuk membangun suatu aplikasi. Informasi dari API *call* akan digunakan untuk mengetahui aktivitas dari *malware*, sehingga informasi aktivitas *malware* akan diklasifikasikan menggunakan teknik *signature based detection* [7].

API *Calls* itu sendiri ada beberapa macam antara lain:

- API *Network*
- API *Memory*
- API *Pipe*
- API *Cryptography*
- API *Registry*
- API *File*

2.5 API Memory

API *memory* merupakan API yang terdapat di sistem operasi Windows dan Linux, yang berguna untuk tempat proses penyimpanan sementara dari tiap aktivitas. API *memory* biasa digunakan *malware* untuk melakukan perulangan dengan tujuan memenuhi sumber daya yang ada pada *memory* tersebut dan membuat *memory* tersebut bekerja menjadi tidak optimal.

3. Metode Penelitian

Metode yang digunakan dalam penelitian ini menggunakan model konseptual. Model konseptual sangat erat hubungannya dengan teori referensi/literatur yang akan digunakan. Model konseptual memberikan koneksi yang membuatnya lebih mudah untuk memetakan sebuah masalah. Selanjutnya model konseptual akan membantu menyederhanakan masalah dengan mengurangi jumlah *resource* yang akan digunakan. Sehingga peneliti dapat menjelaskan model konseptual pada penelitian tugas akhir ini yang memiliki tujuan untuk bagaimana sebuah *malware* dianalisis dengan menggunakan teknik *signature based detection* untuk mendapatkan sebuah perintah program yang diduga sebagai *malware*.

Permasalahan dalam penelitian ini berada pada semakin pesatnya jenis-jenis *malware* yang menyerang komputer pengguna tiap tahunnya. Dari permasalahan tersebut didapatkan cara bagaimana cara untuk mengurangi kerentanan sistem pada setiap komputer pengguna. Cara yang dapat digunakan adalah dengan cara melakukan analisis *malware* pada file yang telah dicurigai sebagai *malware*.

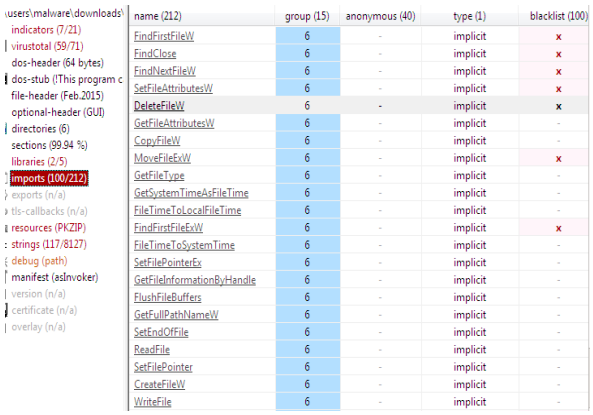
Dengan permasalahan yang ada dan peluang yang terdapat pada penelitian ini, dihasilkan sebuah artefak berupa analisis *malware* menggunakan teknik *signature-based detection*. Untuk menghasilkan analisis *malware* tersebut, diperlukan sebuah teknik untuk melakukannya yang dapat membatu dalam analisis *malware* tersebut. Adapun teori yang digunakan adalah dengan menggunakan teori mengenai *static analysis*, teori *signature-based detection*, dan teori mengenai API *call*.

Analisis yang akan dihasilkan yaitu berupa hasil API *call memory* yang akan dihubungkan dengan hasil *signature* yang telah didapat dalam pengujian ini dengan menggunakan metode *signature-based detection*.

4. Hasil dan Pembahasan

1. Pengujian Menggunakan PEStudio

   PEStudio adalah *tools* untuk analisis *malware* yang dapat memberikan informasi secara detail. Fitur yang digunakan dalam penelitian ini yaitu menggunakan fitur *import* untuk melihat API *call* apa saja yang telah di *import* oleh *malware*.

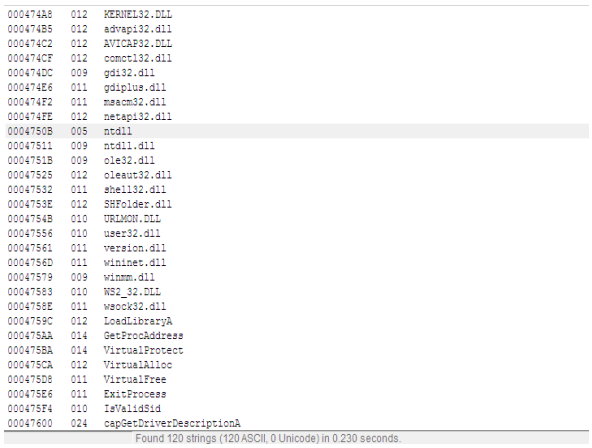


Gambar 1 Hasil PEStudio

Gambar 1 merupakan hasil informasi API *call* yang didapat dari pengujian *malware* menggunakan *sample malware* samp_22.exe. informasi yang didapattkan dari pengujian tersebut ialah mendapatkan mengenai hasil *import* API *call memory* yang dilakukan oleh *malware* tersebut. Contoh API *call memory* yang dipanggil oleh *malware* tersebut yaitu *CopyFileW*, yang bergungsi untuk menyalin *file* yang dienkripsi. Jika fungsi ini tidak jalan, maka *malware* tersebut akan mencoba mengenkripsi *file* tujuan dengan program bawaan yang dibuat oleh *attacker*,

2. Pengujian Menggunakan ShowString

   ShowString adalah *tools* yang digunakan untuk mendapatkan informasi mengenai *plain text* dan menunjukkan semua *string* ASCII dan UNICODE dalam sebuah *file*/*application*. Pada penelitian ini *ShowString* digunakan utuk mencari *string* secara lengkap dari API *call* yang terdapat dari *sample malware*.



Gambar 2 Hasil ShowString

Gambar 2 merupakan hasil *strings* yang didapat dari *sample malware* yang bernama samp_22.exe. Dimana terlihat *strings* dari API *call* yang digunakan seperti *GetProcAddress*, Fungsi *strings* ini yaitu untuk memanggil fungsi DLL dan juga *GetProcAddress* dapat memanggil nama fungsi yang diinginkan dan mengekspor *file* yang diinginkan oleh yang membuat *file* tersebut.

3. Pengujian Menggunakan Cuckoo Sandbox

Cuckoo Sandbox adalah *tools* yang digunakan untuk melakukan analisis *malware* dan memberi *report* secara otomatis apa yang akan dilakukan *malware* terhadap komputer . Data yang diambil dalam pengujian ini dari cuckoo sandbox yaitu API *call memory*.

Gambar 3 Hasil Cuckoo Sandbox 1

Pada Gambar 3 merupakan informasi dari API *memory* yang berhasil didapatkan dengan menggunakan Cuckoo Sandbox. Informasi yang didapatkan dengan menggunakan *Cuckoo Sandbox* ini sangat detail. Dapat dilihat informasi yang didapatkan seperti: TID, *caller*, API *memory*, *arguments*, status, dan *return*. Sebagai contoh adalah API memory *NtOpenProcess* yang memiliki fungsi sebagai membuka jalan untuk *process* lain yang ingin berjalan pada sistem. Dengan membukakan jalan proses ini digunakan untuk membaca dan menulis ke memori proses lain atau untuk menyuntikkan kode ke dalam proses lainnya.

4. Hasil Analisis dengan Metode *Signature Based Detection*

Analisis dengan menggunakan metode *signature-based detection* ini berdasarkan hasil pengujian yang telah dilakukan. Berikut merupakan hasil hubungan antara API *call memory* dengan hasil *signature*.

Table 1 Hasil API *Call Memory*

No	API *Memory*	Jumlah
1	NtCreateSection	28
2	NtMapViewOfSection	23
3	NtOpenSection	20
4	NtUnmapViewOfSection	18
5	NtAllocateVirtualMemory	17
6	VirtualProtectEx	16
7	NtTerminateProcess	14
8	CreateToolhelp32Snapshot	9
9	Process32FirstW	9
10	Process32NextW	9

Table 2 Hasil *Signature*

No	Hasil *Signature*	Jumlah
1	*The Binary likely contains encrypted or compressed data*	24
2	*Creates RWX memory*	21
3	*Creates a hidden or system file*	10
4	*Attempts to modify proxy settings*	9
5	*Installs itself for autorun at windows startup*	8
6	*Reads data out of its own binary image*	8
7	*Deletes its original binary from disk*	7
8	*A process created a hidden window*	7
9	*Expresses interest in spesific running processes*	6
10	*Drops a binary and executes it*	6

Pada Table 1 terdapat 10 API *call memory* terbanyak yang terdeteksi dengan menggunakan Cuckoo Sandbox. API *memory* tersebut yang digunakan oleh *malware* untuk melakukan aksinya ke komputer pengguna yang telah terinfeksi oleh *malware* tersebut. API *memory* saling berkaitan dengan API *memory* lainnya. Dari beberapa *sample malware* terdapat kesamaan karakteristik API *memory* yang dijalankan

untuk menyerang pengguna yang sedang menjalankan *malware* tersebut.

Pada Table 2 terdapat 10 hasil *signature* terbanyak yang didapatkan dengan menggunakan cuckoo sandbox. Hasil *signature* tersebut yang telah dilakukan *malware* ketika menjalankan aksinya ke komputer pengguna yang telah terinfeksi oleh *malware* tersebut. Hasil *signature* tersebut saling berkaitan dengan API *call memory* yang telah didapatkan sebelumnya.

Table 3 Hasil Analisis API *Memory* dengan Hasi *Signature*

No	API *Call Memory*	Hasil *Signature*
1.	-NtCreateSection -NtMapViewOfSection -NtTerminateProcess	*The Binary likely contains encrypted or compressed data*
2.	-NtAllocateVirtualMemory -VirtualProtectEx -NtCreateSection -NtMapViewOfSection -NtOpenSection	*Creates RWX memory*
3.	-NtCreateSection -NtMapViewOfSection -NtUnmapViewOfSection -NtOpenProcess -NtOpenSection	*Creates a hidden or system file*
4.	-NtMapViewOfSection -NtUnmapViewOfSection - NtCreateSection	*Reads data out of its own binary image*
5	-NtCreateSection -NtAllocataVirtualMemory -VirtualProtectEx -NtTerminateProcess	*Deletes its original binary from disk*
6	-CreateToolhelp32Snapshot -Process32FirstW -Process32NextW -NtCreateSection	*Expresses interest in spesific running processes*
7	- NtCreateSection -VirtualProtectEx -NtOpenProcess -NtMapViewOfSection	*Drops a binary and executes it*

Dari beberapa *sample malware* pengujian terdapat kesamaan karakteristik API *memory* yang dijalankan untuk menyerang pengguna yang sedang menjalankan *malware* tersebut. API *memory* yang telah didapatkan akan menghasilkan sebuah *signature* yang berbeda-beda dari setiap *malware*. Hasil *signature* didapatkan ketika API *memory* telah selesai dijalankan pada Cuckoo Sandbox. Karena satu hasil *signature* ini didapatkan dari beberapa proses dari API *memory* yang telah dijalankan oleh *malware* tersebut.

5. Rekomendasi Hasil Analisis

Berdasarkan hasil analisis *malware* dengan menggunakan metode *signature-based detection*, terdapat beberapa rekomendasi yang dibuat berdasarkan hasil *signature* yang telah didapatkan pada pengujian *malware*. Berikut tabel rekomendasi yang dibuat oleh peneliti berdasarkan hasil pengujian dan hasil analisis.

Table 4 Rekomendasi Hasil Analisis *Malware*

No	**Hasil *Signature***	**Rekomendasi**
1	*The Binary likely contains encrypted or compressed data*	Memiliki indikasi sebagai *malware* karena melakukan *drop biner* yang berisi data terenkripsi yang membahayakan *user*. Tindakan pencegahan dilakukan dengan *delete file* dengan nama *file* yang mencurigakan seperti xzkjler.exe dan dapat menggunakan fitur pada *website* virustotal untuk pengecekan *file* yang di *drop* tersebut berbahaya atau tidak.
2	*Creates RWX memory*	Memiliki indikasi sebagai *malware* karena mengubah hak akses pada *file* atau sistem komputer tanpa sepengetahuan *user*. Tindakan pencegahan yang dilakukan dapat meng*install antivirus* atau *antimalware* untuk dilakukan *scanning* secara berkala supaya tidak terjadi hal yang merugikan.
3	*Creates a hidden or system file*	Memiliki indikasi sebagai *malware* karena membuat *file* tersembuyi dan membuat sistem komputer disembunyikan tanpa izin dari *user*. Tindakan pencegahan yang dilakukan dengan tidak *install software* secara sembarangan. Karena beberapa *software* dapat mengandung program berbahaya untuk keamanan sistem komputer.
4	*Reads data out of its own binary image*	Memiliki indikasi sebagai *malware* karena menjalankan program yang dapat membaca *biner* pada sistem komputer. Tindakan pencegahan yang dilakukan dengan meng*install antivirus* untuk dilakukan pengecekkan secara menyeluruh pada

		sistem komputer.
5	*Deletes its original binary from disk*	Memiliki indikasi sebagai *malware* karena mengandung program yang menghapus *biner* aslinya dari *disk* komputer ketika program ini dijalankan tidak akan ketahuan *user*. Tindakan pencegahan yang dilakukan dengan memonitor proses komputer dengan *software* seperti *task manager* dan ProcessMonitor untuk melihat program-program yang mencurigakan tersebut.
6	*Expresses interest in spesific running processes*	Memiliki indikasi sebagai *malware* karena berjalan secara otormatis mencari proses memori sesuai perintah program *malware* tersebut. Tindakan pencegahan yang dilakukan dengan melihat proses komputer melalui *task manager* atau ProcessMonitor untuk melihat dan mematikan proses mencurigakan secara paksa agar tidak membahayakan sistem komputer.
7	*Drops a binary and executes it*	Memiliki indikasi sebagai *malware* karena *import* biner dari program *malware* dan menjalankannya secara otomatis untuk menyerang proses memori. Tindakan pencegahan yang dilakukan dengan meng*install antivirus* atau *antimalware* untuk dilakukan *scanning* secara berkala supaya tidak terjadi hal yang merugikan pengguna komputer.

5. Kesimpulan

Berdasarkan hasil analisis dari pengujian *malware* dengan parameter API call memory menggunakan metode *signature-based detection* yang telah dilakukan dapat diambil kesimpulan sebagai berikut:

1. Peneliti hanya berfokus untuk mencari API *call memory* yang terkandung dari setiap *malware*. API *memory* didapatkan dari pengujian menggunakan Cuckoo Sandbox. Data yang didapat dari Cuckoo Sandbox akan dibandingkan dengan data yang ada dari virustotal. Jika mempunyai kesamaan dengan *database* dari virustotal maka *sample malware* tersebut dapat diindikasikan sebagai *malware* yang berbahaya bagi *user*.
2. Penentuan identifikasi *malware* dilakukan dengan mencari API *call memory* dan hasil *signature*. Berdasarkan data yang diperoleh, hasil API *call memory* dengan hasil *signature* saling terkait. Hasil *signature* dihasilkan dari beberapa API *call memory* yang dijalankan oleh *malware*. Hasil API *call memory* dan hasil *signature* didapatkan dengan menggunakan *tools static analysis* yang digunakan selama pengujian berlangsung.
3. Rekomendasi yang diberikan adalah melakukan *install antivirus* atau *antimalware* untuk dilakukan *scanning* secara berkala agar *file* yang terindikiasi sebagai *malware* akan terlacak dan *user* dapat menghapus *file* tersebut.

Referensi

[1] Zeltser. (2016). *What is malware?* diambil dari https://securingthehuman.sans.org/ouch/2015#january2015. Diakses pada tanggal 10 November 2018.

[2] Gandotra, E., Bansal, D., & Sofat, S. (2014). Analisis *malware* and Classification: A Survey. *Journal of Information Security*, *05*(02), 56–64. https://doi.org/10.4236/jis.2014.52006

[3] Chandra, S., Hutauruk, Y., Yulianto, F. A., & Satrya, G. B. (2016). Analisis *malware* Pada Windows Operating System Untuk Mendeteksi Trojan Analisis *malware* on Windows Operating System To Detect Trojan, *3*(2), 3590–3595.

[4] Zalavadiya, N., & Sharma, P. (2017). A Methodology of Analisis *malware*, Tools and Technique for windows platform – RAT Analysis. *International Journal of Innovative Research in Computer and Communication Engineering*, *5*(2), 1302–1309. https://doi.org/10.15680/IJIRCCE.2017.

[5] Uppal, D., Mehra, V., & Verma, V. (2014). Basic survey on Analisis *malware*, Tools and Techniques. *International Journal on Computational Science & Applications*, *4*(1), 103–112. https://doi.org/10.5121/ijcsa.2014.4110

[6] Chandra, S., Hutauruk, Y., Yulianto, F. A., & Satrya, G. B. (2016). Analisis *malware* Pada Windows Operating System Untuk Mendeteksi Trojan Analisis *malware* on Windows Operating System To Detect Trojan, *3*(2), 3590–3595.

[7] Mohamed, G. A. N., & Ithnin, N. B. (2017). Survey on Representation Techniques for *Malware* Detection System. *American Journal of Applied Sciences*, *14*(11), 1049–1069. https://doi.org/10.3844/ajassp.2017.1049.1069

[8] Ravi, C., & Manoharan, R. (2012). *Malware* Detection using Windows Api Sequence and Machine Learning, *43*(1), 12–16.

[9] Agrawal, M., Singh, H., Gour, N., & Kumar, A. (2014). Evaluation on Analisis *malware*. *International Journal of Computer Science and Information Technologies*, *5*(3), 3381–3383.

[10] Cahyanto, T. A., Wahanggara, V., & Ramadana, D. (2017). Analisis dan Deteksi *Malware* Menggunakan Metode *Malware* Analisis Dinamis dan *Malware* Analisis Statis. *Justindo*, *2*(1), 19–30.

[11] Rao, P. V., & Hande, K. (2017). A comparative study of static , dynamic and hybrid analysis techniques for android *malware* detection, *5*(2), 1433–1436.

[12] Iwamoto, K., & Wasaki, K. (2012). *Malware* classification based on extracted API sequences using static analysis.

Proceedings of the Asian Internet Engineeering Conference on - AINTEC '12, (June), 31–38. https://doi.org/10.1145/2402599.2402604

[13] Jul Ismail. (2019). Teknik Penyebaran *Malware* | Jul Ismail. [online] Available at: https://julismail.staff.telkomuniversity.ac.id/teknik-penyebaran-*malware*/ [Accessed 12 November. 2018].

[14] Souri, A., & Hosseini, R. (2018). A state-of-the-art survey of *malware* detection approaches using data mining techniques. *Human-Centric Computing and Information Sciences*, *8*(1). https://doi.org/10.1186/s13673-018-0125-x

[15] O. T. . Suryati and A. . Budiono, "Impact Analysis of Malware Based on Call Network API With Heuristic Detection Method", *Int. J. Adv. Data Inf. Syst.*, vol. 1, no. 1, pp. 1-8, Apr. 2020.

[16] A. F. . Muhtadi and A. Almaarif, "Analysis of Malware Impact on Network Traffic using Behavior-based Detection Technique", *Int. J. Adv. Data Inf. Syst.*, vol. 1, no. 1, pp. 17-25, Apr. 2020.

[17] Docs.microsoft.com. (2018). Windows API Index – Windows applications. Diakses pada 17 November 2018, dari https://docs.microsoft.com/en-us/windows/desktop/apiindex/windows-api-list.