 doi.org/10.56706/ik.v19i3.132

Automasi Forensik Memori berbasis *Volatility* dan YARA untuk Deteksi *Ransomware*

Syaiful Andy
PT Telkom Indonesia, syaifulandy@gmail.com

Riwayat Artikel

Dikirim 16 Aug 2025
Diterima 6 Nov 2025
Diterbitkan 9 Des 2025

Kata kunci:

Forensik memori
malware
Volatility
YARA
Automasi

Abstrak

Banyaknya insiden serangan *malware* yang terjadi saat ini, menuntut adanya proses digital forensik yang cepat dan akurat. Penelitian ini mengusulkan teknik automasi forensik *file* memori untuk triase proses *malware*, dengan studi kasus sampel memori yang terinfeksi *ransomware WannaCry* dan *Teslacrypt*.

Metodologi dimulai dengan akuisisi memori dari sistem yang terinfeksi *WannaCry* dan *Teslacrypt*. Volatility 3 digunakan untuk mengekstraksi daftar proses dan mengidentifikasi proses mencurigakan atau anomali berdasarkan indikator seperti hubungan *parent-child* yang tidak wajar, *path* eksekusi yang tidak semestinya, serta nama proses yang tidak dikenal. Proses mencurigakan kemudian diekstraksi menggunakan kemampuan *file dumping* Volatility 3. Hasil *dump* dipindai dengan YARA *rule* yang tersedia di publik (*open source*) untuk mendeteksi pola dan tanda *malware* yang sudah dikenal. Untuk memperkuat hasil analisis, berkas yang terdeteksi di-*hash* dan diverifikasi lebih lanjut melalui VirusTotal, sehingga dapat dibandingkan dengan basis data *antivirus* serta intelijen ancaman publik. Semua proses di atas dikemas dalam sebuah *shell* skrip yang siap dijalankan.

Kontribusi utama penelitian ini adalah proses automasi triase proses *malicious* dengan menjalankan satu skrip. Hasil pengujian pada sampel *file* memori yang digunakan, skrip ini dapat mendeteksi proses *malware* dengan cepat (<10 menit) dengan hasil akurasi rata-rata lebih dari 97%, dan efisiensi triase rata-rata kurang dari 3%.

Keywords:

Forensics
memory
malware
Volatility
YARA
Automation

Abstract

The increasing frequency of malware attacks demands faster and more accurate digital forensic processes. This research proposes an automated forensic technique for memory file analysis to triage malware processes, using memory samples infected with WannaCry and Teslacrypt ransomware as case studies.

The methodology begins with memory acquisition from systems infected with WannaCry and Teslacrypt. Volatility 3 is used to extract process lists and identify suspicious or anomalous processes based on indicators such as abnormal parent–child relationships, unexpected execution paths, and unknown process names. The suspicious processes are then extracted using Volatility's file dumping capabilities. The resulting dumps are scanned using publicly available YARA rules to detect known malware patterns and signatures. To strengthen the analysis results, detected files are hashed and further verified through VirusTotal for cross-comparison with antivirus databases and public threat intelligence. All processes are encapsulated into a single executable shell script.

The main contribution of this study is the automation of malicious process triage through a single-script workflow. Based on the testing results on memory samples, the script can detect malware processes quickly (in less than

10 minutes) with an average accuracy of over 97% and an average triage efficiency of under 3%.

1. PENDAHULUAN

Banyaknya insiden serangan *malware* yang terjadi saat ini, menuntut adanya proses digital forensik yang cepat dan akurat. Riset volGPT menunjukkan potensi LLM dalam mengotomatisasi forensik memori. Namun, riset tersebut [1] menjelaskan di bagian analisis *false positive*, penggunaan LLM memiliki kelemahan berupa halusinasi, yaitu proses sistem operasi Windows yang sah seperti svchost.exe, explorer.exe dapat diklasifikasikan sebagai mencurigakan. Kesalahan analisis ini sering berdampak ke proses turunannya sehingga menghasilkan tingkat *false positive* yang tinggi. Hal ini menegaskan keterbatasan pendekatan berbasis LLM dalam mendeteksi proses berbahaya.

Untuk mengatasi masalah tersebut, penelitian ini mengadopsi pendekatan non-AI berbasis aturan (*rule-based heuristics*) pada tahap awal analisis. Dengan cara ini, proses sistem yang sudah diketahui aman dapat dieliminasi sejak awal sehingga mengurangi potensi halusinasi dan meminimalkan *false positive*. Selain itu, berbeda dengan volGPT, metode ini tidak bergantung pada API LLM publik yang berbiaya tinggi maupun infrastruktur besar untuk *fine-tuning*. volGPT juga sangat bergantung pada nama proses *malware*, yang dapat menurunkan akurasi ketika *malware* menyamar menggunakan nama yang tampak sah.

Untuk meningkatkan akurasi deteksi, alur kerja di riset ini dimulai dari mengekstrak proses yang tidak dikenal atau mencurigakan yang teridentifikasi dalam tahap analisis. *File* hasil ekstraksi kemudian dipindai dengan aturan YARA untuk mendeteksi *malware* yang sudah diketahui. Walaupun kehandalan metode ini bergantung pada kualitas aturan YARA, pendekatan ini tetap lebih dapat dipercaya dibanding hanya mengandalkan nama proses seperti pada volGPT. Untuk memperkuat hasil pemindaian, *file* yang terdeteksi sebagai *malware* oleh YARA akan dihitung nilai *hash* MD5-nya dan diverifikasi melalui VirusTotal menggunakan API VirusTotal untuk memberikan validasi tambahan sekaligus memperkaya hasil analisis.

Kontribusi utama penelitian ini adalah proses automasi triase proses malicious dengan menjalankan satu skrip. Hasil pengujian pada sampel *file* memori yang digunakan, skrip ini dapat mendeteksi proses *malware* dengan cepat (<10 menit) dengan hasil akurasi rata-rata lebih dari 97%, dan efisiensi triase rata-rata kurang dari 3%.

2. LANDASAN TEORI

Forensik memori merupakan cabang analisis digital yang fokus pada data volatil dalam RAM. Penelitian terbaru menunjukkan peningkatan pemanfaatan analisis memori volatil untuk mendeteksi aktivitas berbahaya, karena setiap insiden meninggalkan jejak di memori. Terdapat banyak *tools* yang dapat digunakan untuk membantu proses forensik memori. Dari *review* yang dilakukan pada jurnal ACM, *tools* alternatif selain Volatility 3 sudah usang secara teknologi atau dijual komersial. Volatility 3 dipilih untuk penelitian ini karena bersifat *open source*, masih aktif dikembangkan, dapat diprogram melalui skrip, serta mampu menganalisis *memory dump* Windows dan Linux terbaru [2]. Hal senada juga disampaikan dalam penelitian lain [3], yang menyatakan bahwa dibanding *tools* lain seperti LibVMI dan Rekall, Volatility 3 dapat melakukan analisis proses, koneksi jaringan, aktivitas *registry*, injeksi kode dengan waktu eksekusi tercepat.

Dalam tahap analisis forensik, langkah awal yang penting adalah melakukan *imaging* terhadap barang bukti untuk menjaga keaslian data. *Memory image* ini kemudian dianalisis menggunakan *tools* seperti Volatility 3. Akshay et al. [4] mengusulkan teknik analisis cepat dengan menggunakan *memory image* sistem operasi bersih sebagai *baseline* pembanding untuk mendeteksi proses asing pada mesin tersangka. Pendekatan ini terbukti efektif, terutama dalam kasus serangan *malware*, karena membantu mengidentifikasi proses *malicious* secara lebih cepat.

YARA merupakan alat *open source* yang menggunakan pendekatan berbasis aturan untuk mendeteksi *malware* melalui pencocokan pola teks maupun biner. Aturan YARA disusun dari

kombinasi string dan logika untuk mengidentifikasi ciri khas suatu keluarga atau varian *malware* secara cepat dan efisien [5].

Menurut penelitian Hamid dan Rahman [6] pada bagian rekomendasi, mereka menekankan perlunya pengembangan lebih lanjut terhadap *tools* analisis memori, dan integrasi *workflow* antar *tools* forensik sehingga proses analisis forensik dapat berlangsung lebih efektif dan efisien. Oleh karena itu pada penelitian ini nanti kita akan membuat suatu *workflow* yang dikemas dalam bentuk skrip untuk mengotomasi sebagian proses analisis forensik memori.

Penelitian ini menggunakan dataset *benchmark* yang tersedia secara publik berupa *dump* memori sistem yang terinfeksi *ransomware* untuk mengevaluasi alur kerja forensik otomatis yang diusulkan. Secara khusus, digunakan dataset berjudul "*Virtual Machine (VM) Memory Dumps for Ransomware Forensics*" yang dipublikasikan oleh Arfeen et al. (2020) pada *Harvard Dataverse* [7], dan juga digunakan dalam penelitian volGPT. Dataset ini dibuat dengan cara menangkap *snapshot* mesin virtual setiap lima menit selama eksekusi *ransomware*.

Namun, karena beberapa keluarga *ransomware* melakukan *self-delete* setelah proses enkripsi, penelitian volGPT memperkenalkan prosedur praproses yang terdiri dari empat langkah yang mencakup pemantauan proses saat *runtime*, analisis memori menggunakan *plugin pslist* dari Volatility, serta pembuatan label dengan menginterseksi kedua daftar proses tersebut untuk mengidentifikasi proses *malware* yang valid.

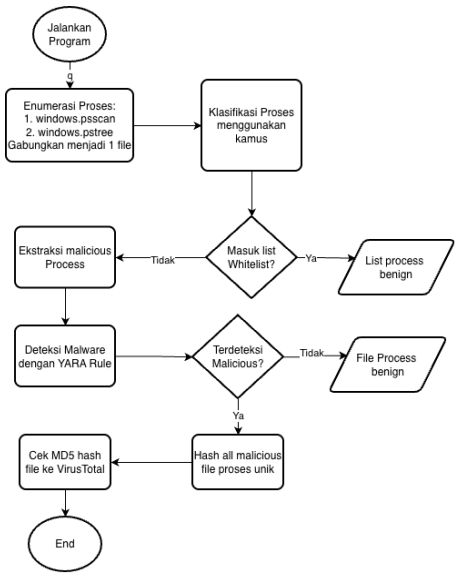
Dalam penelitian ini, dipilih dua *snapshot* memori untuk masing-masing dari dua keluarga *ransomware*, sehingga total terdapat empat sampel dengan ukuran sekitar 1,21 GB per *file*, yaitu:

1. *WannaCry*: mengambil *snapshot* nomor 109 dan 149
2. *Teslacrypt*: mengambil *snapshot* nomor 403 dan 452

Pemilihan sampel dilakukan secara acak untuk mewakili eksekusi berbeda dari masing-masing *ransomware*, sehingga memungkinkan evaluasi yang ringan namun tetap fokus terhadap alur kerja forensik yang diusulkan. Tidak diperlukan praproses manual atau validasi tambahan, karena *dump* memori tersebut disediakan dalam format *raw* yang sepenuhnya kompatibel dengan Volatility sehingga dapat langsung diintegrasikan ke dalam *pipeline* analisis otomatis yang dibuat.

3. METODE PENELITIAN

Penelitian ini mengusulkan alur kerja forensik memori otomatis yang mengombinasikan analisis proses dengan deteksi *malware* berbasis YARA *rule*. Alur kerja tersebut terdiri dari lima tahap utama: (1) enumerasi proses, (2) klasifikasi proses, (3) ekstraksi *dump* memori, (4) pemindaian berbasis YARA, dan (5) pengayaan intelijen ancaman seperti yang ditampilkan pada Gambar 1.



Gambar 1. Alur Kerja Automasi Forensik Memori

3.1. Enumerasi Proses

Framework Volatility 3 versi 2.26.2 digunakan untuk mengekstraksi informasi terkait proses dari *dump* memori melalui dua *plugin* pelengkap: *windows.psscan* dan *windows.pstree*. *Plugin psscan* merekonstruksi struktur proses dari *memory pool*, sehingga mampu mendeteksi baik proses aktif maupun proses yang telah dihentikan. Sementara itu, *plugin pstree* menyusun kembali hubungan hierarki *parent-child* antar proses yang sedang berjalan.

Keluaran dari kedua *plugin* kemudian digabungkan untuk membentuk tampilan proses yang komprehensif, mencakup atribut seperti *Process* ID (PID), *Parent Process* ID (PPID), *path file* eksekusi, nama proses induk, waktu keluar (exit_time), dan status.

3.2. Algoritma Klasifikasi Proses

Setiap proses dievaluasi dan dikategorikan ke dalam salah satu kelas berdasarkan aturan kamus *whitelist* menjadi empat kelas:

a) OK: Nama proses, *path file*, dan induk proses sesuai dengan *whitelist*.
b) UNKNOWN_PROCESS: Nama proses tidak terdapat dalam kamus *whitelist*, atau nama ada dalam *whitelist* tetapi path/induk proses mencurigakan.
c) MALICIOUS_PATH: Nama proses valid, namun *path* eksekusi dianggap berbahaya.
d) MALICIOUS_PARENT: Nama dan jalur proses valid, tetapi memiliki induk yang tidak wajar.

Whitelist diadaptasi dari *SANS Hunt Evil* [8] yang mendefinisikan hubungan *parent-child* proses sah dalam lingkungan Windows bersih. *Whitelist* tersebut disimpan dalam sebuah berkas teks yang berfungsi sebagai kamus referensi, seperti ditunjukkan pada Gambar 2. Kerangka klasifikasi ini memfasilitasi penyaringan proses mencurigakan secara efisien dengan mendeteksi anomali berdasarkan hubungan proses. Pada Gambar 3 kita dapat melihat cuplikan hasil klasifikasi proses dari *snapshot memory Wannacry*.

```
└─# cat kamus.txt
smss.exe;\windows\system32\smss.exe;system
csrss.exe;\windows\system32\csrss.exe;smss.exe
wininit.exe;\windows\system32\wininit.exe;smss.exe
winlogon.exe;\windows\system32\winlogon.exe;smss.exe
services.exe;\windows\system32\services.exe;wininit.exe
lsass.exe;\windows\system32\lsass.exe;wininit.exe
lsaiso.exe;\windows\system32\lsass.exe;wininit.exe
lsm.exe;\windows\system32\lsm.exe;wininit.exe
svchost.exe;\windows\system32\svchost.exe;services.exe
runtimebroker.exe;\windows\system32\runtimebroker.exe;svchost.exe
taskhostw.exe;\windows\system32\taskhostw.exe;svchost.exe
userinit.exe;\windows\system32\userinit.exe;winlogon.exe
explorer.exe;\windows\explorer.exe;userinit.exe
fontdrvhost.exe;\windows\system32\fontdrvhost.exe;winlogon.exe
dwm.exe;\windows\system32\dwm.exe;winlogon.exe
spoolsv.exe;\windows\system32\spoolsv.exe;services.exe
```

Gambar 2. Kamus *Whitelist* Proses

```
└─# cat output_pstree_analysis_Snapshot_malicious_109_Wannacry.elf.csv
pid,ppid,image,path,parent_proc,exit_time,status
240,4,smss.exe,windows\system32\smss.exe,system,RUNNING,OK
384,240,smss.exe,windows\system32\smss.exe,smss.exe,2020-07-29 19:35:35.000000 UTC,MALICIOUS_
392,384,csrss.exe,windows\system32\csrss.exe,smss.exe,RUNNING,OK
428,384,winlogon.exe,windows\system32\winlogon.exe,smss.exe,RUNNING,OK
744,428,dwm.exe,windows\system32\dwm.exe,winlogon.exe,RUNNING,OK
732,428,logonui.exe,windows\system32\logonui.exe,winlogon.exe,2020-07-29 19:38:59.000000 UTC,
328,320,csrss.exe,windows\system32\csrss.exe,UNKNOWN,RUNNING,MALICIOUS_PARENT
400,320,wininit.exe,windows\system32\wininit.exe,UNKNOWN,RUNNING,MALICIOUS_PARENT
488,400,services.exe,windows\system32\services.exe,wininit.exe,RUNNING,OK
704,488,svchost.exe,windows\system32\svchost.exe,services.exe,RUNNING,OK
256,488,spoolsv.exe,windows\system32\spoolsv.exe,services.exe,RUNNING,OK
1632,488,svchost.exe,windows\system32\svchost.exe,services.exe,RUNNING,OK
2496,488,taskhostex.exe,windows\system32\taskhostex.exe,services.exe,RUNNING,UNKNOWN_PROCESS
2976,488,searchindexer.,windows\system32\searchindexer.exe,services.exe,RUNNING,UNKNOWN_PROCE
```

Gambar 3. Cuplikan Hasil Analisis Proses

3.3. Ekstraksi Proses Mencurigakan

Untuk mempercepat *dumping file* pada memori dari proses mencurigakan, prioritas diberikan pada proses yang tidak masuk kategori "OK" atau dieksekusi dari direktori selain "System32". Namun, demi kelengkapan, *dumping* juga dilakukan terhadap seluruh proses yang terdeteksi *plugin* ***windows.malfind***, meskipun berasal dari direktori "System32", karena proses tersebut mungkin menunjukkan indikasi *code injection* atau perilaku berbahaya lainnya.

Hasil *dumping* disusun dalam direktori terpisah berdasarkan *process ID* (PID) masing-masing, sehingga memudahkan manajemen, korelasi, dan analisis *file* yang diekstrak. Berkas hasil *dump* dapat diteruskan ke tim analisis *malware* khusus untuk pemeriksaan lebih lanjut seperti analisis dinamis, *reverse engineering*, atau perbandingan dengan *malware* lain yang sudah dikenal. Gambar 4 menunjukkan cuplikan hasil *dump* proses yang terorganisasi berdasarkan PIDnya.

```
└─# ls -l | grep pid_
drwxr-xr-x 2 root root  12288 Jul 12 11:42 pid_1100
drwxr-xr-x 2 root root  12288 Jul 12 11:42 pid_1192
drwxr-xr-x 2 root root   4096 Jul 12 11:42 pid_1488
drwxr-xr-x 2 root root  16384 Jul 12 11:42 pid_1824
drwxr-xr-x 2 root root   4096 Jul 12 11:42 pid_1924
drwxr-xr-x 2 root root  12288 Jul 12 11:42 pid_2160
drwxr-xr-x 2 root root   4096 Jul 12 11:42 pid_2272
drwxr-xr-x 2 root root  16384 Jul 12 11:42 pid_2388
drwxr-xr-x 2 root root  12288 Jul 12 11:42 pid_2444
drwxr-xr-x 2 root root  12288 Jul 12 11:42 pid_2496
drwxr-xr-x 2 root root   4096 Jul 12 11:42 pid_2620
drwxr-xr-x 2 root root  36864 Jul 12 11:43 pid_2712
drwxr-xr-x 2 root root  12288 Jul 12 11:42 pid_2828
drwxr-xr-x 2 root root  20480 Jul 12 11:43 pid_2960
drwxr-xr-x 2 root root  12288 Jul 12 11:43 pid_2976
```

Gambar 4. File Hasil Dump yang Terorganisasi Berdasarkan Folder PID

3.4. Deteksi *Malware* Berbasis YARA

Tahapan selanjutnya adalah proses triase *file* proses *malicious*. *File* hasil *dump* dipindai menggunakan kumpulan aturan YARA dari proyek **fsYara** [9]. Semua aturan dikompilasi ke dalam satu berkas .yar untuk meningkatkan efisiensi dan konsistensi eksekusi. Tahap ini bertujuan mengidentifikasi pola tanda *malware* yang sudah dikenal, termasuk varian WannaCry dan *Teslacrypt*, berdasarkan karakteristik *malicious file* yang telah dikenali oleh YARA *rules*. Berbeda dengan pendekatan berbasis *machine learning*/AI, *workflow* ini menghindari halusinasi sekaligus meningkatkan *explainability*, sehingga lebih cocok digunakan dalam skenario respons insiden yang membutuhkan bukti forensik yang dapat diverifikasi. Gambar 5 menunjukkan cuplikan hasil pemindaian YARA pada *snapshot memory Wannacry*.

```
└─# cat Report_scan_dump_memory_Snapshot_malicious_109_Wannacry.elf.txt
[FOUND] Rule: Compiled_rule_fsyara_PE-ELFs | Target: /opt/forensic_tools/volatility/dump_mem
ory_Snapshot_malicious_109_Wannacry.elf/pid_3328/file.0xfa80013b7300.0xfa8002dee860.ImageSec
tionObject.@WanaDecryptor@.exe.img
Win32_Ransomware_WannaCry /opt/forensic_tools/volatility/dump_memory_Snapshot_malicious_109_
Wannacry.elf/pid_3328/file.0xfa80013b7300.0xfa8002dee860.ImageSectionObject.@WanaDecryptor@.
exe.img

[FOUND] Rule: Compiled_rule_fsyara_PE-ELFs | Target: /opt/forensic_tools/volatility/dump_mem
ory_Snapshot_malicious_109_Wannacry.elf/pid_3468/file.0xfa80013b7300.0xfa8002dee860.ImageSec
tionObject.@WanaDecryptor@.exe.img
Win32_Ransomware_WannaCry /opt/forensic_tools/volatility/dump_memory_Snapshot_malicious_109_
Wannacry.elf/pid_3468/file.0xfa80013b7300.0xfa8002dee860.ImageSectionObject.@WanaDecryptor@.
exe.img
```

Gambar 5. Hasil Laporan Pemindaian YARA

3.5. Pengayaan Intelijen Ancaman Tambahan

Sebagai langkah tambahan, digunakan API gratis dari VirusTotal untuk memperkaya hasil analisis. *File* yang sebelumnya terdeteksi sebagai berbahaya oleh YARA dihitung *hash*-nya menggunakan algoritma MD5. Hanya *hash* unik yang dikirimkan ke VirusTotal untuk analisis.

Respons JSON dari VirusTotal kemudian diproses dan dikonversi menjadi format CSV terstruktur dengan *field* sebagai berikut: *target, md5, name, malicious, undetected, yara_rules,*

threat_names, dan *threat_categories*. Gambar 6 menunjukkan cuplikan laporan hasil verifikasi VirusTotal. Terlihat hanya satu *file* yang terdeteksi oleh VirusTotal sebagai *ransomware Wannacry*.

```
└─# cat hasil_virustotal_Snapshot_malicious_109_Wannacry.elf.csv
target,md5,name,malicious,undetected,yara_rules,threat_names,threat_categories
"/opt/forensic_tools/volatility/dump_memory_Snapshot_malicious_109_Wannacry.elf/pid_1056/fil
e.0xfa80013b7300.0xfa8002dee860.ImageSectionObject.@WanaDecryptor@.exe.img","dff310ecd71f8a7
36c6ea92b374de8c2","LODCTR.EXE",60,12,"win_wannacryptor_auto;Win32_Ransomware_WannaCry;INDIC
ATOR_SUSPICIOUS_GENRansomware;win_wannacryptor_auto;Win32_Ransomware_WannaCry;INDICATOR_SUSP
ICIOUS_GENRansomware","wanna;wannacryptor;wannacry","ransomware;trojan;dropper"
"/opt/forensic_tools/volatility/dump_memory_Snapshot_malicious_109_Wannacry.elf/pid_4532/fil
e.0xfa80013c7f20.0xfa80013c54c0.DataSectionObject.taskhsvc.exe.dat","8a622d22611c25adfd8f97f
4bf81b21a","-","-","-","-","-","-"
"/opt/forensic_tools/volatility/dump_memory_Snapshot_malicious_109_Wannacry.elf/pid_4532/fil
e.0xfa80013c7f20.0xfa8001ea0550.ImageSectionObject.taskhsvc.exe.img","f06d6700514dab7def34bd
1675a25e2b","-","-","-","-","-","-"
"/opt/forensic_tools/volatility/dump_memory_Snapshot_malicious_109_Wannacry.elf/pid_4532/fil
e.0xfa80013cf500.0xfa80019dcf20.DataSectionObject.libevent-2-0-5.dll.dat","aa5eb8fc4a9980554
86164ea0ffa3c6f","libevent-2-0-5.dll",0,64,"","",""
"/opt/forensic_tools/volatility/dump_memory_Snapshot_malicious_109_Wannacry.elf/pid_4532/fil
e.0xfa80016a2da0.0xfa80029c34a0.DataSectionObject.libeay32.dll.dat","49d0572793690b493f91e62
411e616bf","-","-","-","-","-","-"
```

Gambar 6. Laporan Hasil Verifikasi VirusTotal

4. HASIL DAN PEMBAHASAN

Seluruh alur kerja yang telah dijelaskan sebelumnya diintegrasikan ke dalam satu skrip, dengan waktu eksekusi diukur dari awal hingga akhir. Seluruh percobaan dijalankan pada perangkat MacBook Air dengan prosesor Apple M3 dan RAM 8 GB. Seluruh komponen perangkat lunak dieksekusi dalam lingkungan *Docker container* untuk memastikan konsistensi serta isolasi pada setiap percobaan.

Alur kerja forensik memori otomatis berhasil mendeteksi sampel *ransomware WannaCry* dan *Teslacrypt* melalui penerapan aturan YARA pada *dump* memori yang diperoleh dari proses mencurigakan. Bagian ini mengevaluasi hasil dari setiap tahapan metodologi yang telah dibuat.

4.1. Evaluasi Klasifikasi Proses

Evaluasi efektivitas klasifikasi dilakukan dengan mengukur dampaknya dalam mengurangi jumlah proses yang perlu diperiksa lebih lanjut.

Tabel 1. Hasil Klasifikasi Proses

Memory *Snapshot*	**Total PID**	**PID tersaring (% Retained)**
Wannacry - 109	124	107 (86,3%)
Wannacry - 149	49	32 (65,3%)
Teslacrypt - 403	65	47 (72,3%)
Teslacrypt - 452	46	28 (60,9%)

Pada Tabel 1, kolom **Total PID** menunjukkan jumlah proses yang diekstraksi dari *file* memori, sedangkan **PID** tersaring menunjukkan jumlah proses yang diklasifikasikan mencurigakan/tidak diketahui menggunakan algoritma klasifikasi proses. Hasil ini menunjukkan algoritma mampu mengurangi *noise* dengan mengeliminasi proses yang aman. Hasil tahap ini tersisa antara 60,9% hingga 86,3% proses yang disisakan untuk tahap analisis berikutnya, sehingga meningkatkan efisiensi dengan mempersempit ruang lingkup ke proses yang lebih relevan.

4.2. Evaluasi Pemindaian YARA

Langkah selanjutnya adalah triase *file* atau proses *malicious*. Dimulai dari proses *dump files* menggunakan Volatility 3 terhadap seluruh PID yang telah tersaring sebelumnya. *File* hasil *dump* tersebut kemudian di-*scan* menggunakan YARA *rule* untuk mendeteksi proses *malicious*. Tabel 2 menyajikan jumlah *file* seluruh PID tersaring yang di-*dump* dan hasil dari tahap pemindaian menggunakan YARA *rule*. Kolom ***Total Dumped files*** menunjukkan jumlah *file* yang diekstrak dan dipindai, sedangkan kolom ***File*** terdeteksi *malicious* menunjukkan jumlah *file* yang cocok dengan YARA *rules*.

Tabel 2. Hasil Deteksi YARA pada *File Dump*

Memory Snapshot	***Total Dumped Files***	***File* terdeteksi *Malicious***
Wannacry - 109	4681	71
Wannacry - 149	1868	1
Teslacrypt - 403	2084	1
Teslacrypt - 452	1437	4

4.3. Evaluasi Intelijen Ancaman

Setiap *file malicious* yang telah terdeteksi oleh YARA *rule* kemudian dihitung nilai MD5nya untuk melihat ada berapa *file* proses unik yang mencurigakan. Hasil MD5 ini kemudian kita cocokkan dengan *database* VirusTotal untuk memperkaya hasil analisis YARA *rule*. Tabel 3 merangkum hasil dari tahap pengayaan intelijen ancaman menggunakan VirusTotal.

Tabel 3. Hasil Verifikasi VirusTotal

Memory *Snapshot*	**MD5 unik *file* proses mencurigakan**	**Terdeteksi VirusTotal**
Wannacry - 109	5	1
Wannacry - 149	1	0
Teslacrypt - 403	1	0
Teslacrypt - 452	1	0

Kolom **MD5 unik *file* proses mencurigakan** menunjukkan jumlah *hash file* berbeda yang teridentifikasi mencurigakan oleh YARA. Disini kita dapat mengamati bahwa untuk *Wannacry* proses mencurigakan berkurang 4 proses dari *snapshot* nomor 109 ke *snapshot* nomor 149. Hal ini mempertegas fakta bahwa waktu pengambilan *sample memory* dapat mempengaruhi hasil analisis.

Kolom **Terdeteksi VirusTotal** menunjukkan jumlah *hash* yang juga dikenali sebagai *malware* oleh VirusTotal. Hanya satu *file* dari *snapshot WannaCry* - 109 yang terdapat di *database* VirusTotal, sementara sisanya tidak terdeteksi. Disini kita dapat melihat bahwa deteksi *malware* berbasis MD5 memiliki potensi *false negative* yang cukup tinggi.

Selanjutnya kita akan membandingkan *labeling malicious process* dari paper VolGPT [1] dengan hasil deteksi YARA *rule* di Tabel 4. Kita hanya membandingkan *list process malicious* dari paper VolGPT [1] yang ada di keluaran *plugin pslist* terhadap *memory snapshot* yang kita gunakan.

Tabel 4. Perbandingan Label Proses VolGPT dengan Deteksi YARA

Memory *Snapshot*	**VolGPT Malicious Process Label [1]**	**Detected by YARA**
Wannacry	*ed01ebfbc9eb5b*	Tidak
Wannacry	*wanadecryptor*	Iya
Wannacry	*taskhsvc*	Iya
Wannacry	*conhost*	Tidak
Teslacrypt	*fjsauxu*	Iya
Teslacrypt	*prvudcr*	Tidak
Teslacrypt	*vssadmin*	Tidak

Paper VolGPT menandai suatu proses sebagai mencurigakan dari pengamatan proses tambahan yang muncul sejak eksekusi *ransomware* di *virtual machine*. Analisis lebih lanjut terhadap Tabel 4 untuk proses yang tidak dideteksi oleh YARA menunjukkan hal berikut:

1. ed01ebfbc9eb5b adalah proses induk dari proses *wanadecryptor* yang terdeteksi oleh YARA.
2. conhost muncul sebagai proses turunan dari taskhsvc yang terdeteksi YARA.
3. vssadmin muncul sebagai proses turunan dari fjsauxu yang terdeteksi YARA.
4. prvudcr tidak diketahui *parent* prosesnya dari hasil *psscan*/*pstree*, tetapi diketahui berjalan di direktori yang sama dengan proses fjsauxu (appdata\appdata\roaming\).

4.4. Evaluasi Akurasi dan Efisiensi Triase

Disini kita akan menggunakan metrik evaluasi akurasi dan efisiensi triase seperti paper VolGPT [1]. Efisiensi triase merupakan indikator yang dapat digunakan untuk menilai seberapa besar peningkatan efektivitas kerja yang diperoleh oleh investigator. Metrik ini menunjukkan perbandingan antara jumlah proses mencurigakan yang berhasil diidentifikasi dengan total keseluruhan proses, yaitu menghitung seberapa banyak proses mencurigakan yang dapat ditemukan dari seluruh *list* proses keluaran *plugin pslist* pada Volatility 3. Rumus yang digunakan sebagai berikut:

1. Akurasi =(TP+TN)/(TP+TN+FP+FN)
2. Efisiensi triase =(TP+FP)/(TP+TN+FP+FN)

Tabel 5. Evaluasi Akurasi dan Efisiensi Triase

Snapshot	Total PID	TP	TN	FP	FN	Akurasi	Efisiensi Triase
Wannacry – 109	124	2	118	2	2	96.77%	3.23%
Wannacry – 149	49	1	47	0	1	97.96%	2.04%
Rata-rata Wannacry	173	3	165	2	3	97.11%	2.89%
Teslacrypt – 403	65	1	62	0	2	96.92%	1.54%
Teslacrypt - 452	46	1	44	1	0	97.82%	4.35%
Rata-rata *Teslacrypt*	111	2	106	1	2	97.29%	2.70%

Keterangan:
TP = True Positive
TN = True Negative
FP = False Positive
FN = False Negative

Tabel 6. Perbandingan Akurasi dan Efisiensi Metode YARA dengan Paper VolGPT

Memory *Snapshot*	Akurasi YARA	Akurasi VolGPT	Efisiensi YARA	Efisiensi VolGPT
Wannacry	97.11%	95.74%	2.89%	11.06%
Teslacrypt	97.29%	88.14%	2.70%	8.073%

Tabel 5 menampilkan hasil evaluasi dari metode yang digunakan dalam penelitian ini. Sedangkan Tabel 6 menampilkan perbandingan nilai akurasi dan efisiensi rata-rata antara metode YARA dengan VolGPT. Dalam kasus ini metode YARA terlihat lebih akurat dan efektif dibandingkan metode VolGPT. Sebagai catatan, pada penelitian ini hanya digunakan dua sampel *snapshot memory* untuk tiap *malware*. Sebagai pembanding jumlah proses yang dianalisis pada penelitian ini untuk *Wannacry* total proses yang ada sebanyak 173 proses, sedangkan pada paper VolGPT [1] sebanyak 5826 proses. Untuk *Teslacrypt*, total proses pada penelitian ini sebanyak 111 proses, sedangkan pada paper VolGPT [1] sebanyak 767 proses.

4.5. Evaluasi Kecepatan

Evaluasi kecepatan analisis dan ekstraksi data ditampilkan pada tabel 6. Hal ini menunjukkan waktu analisis sangat dipengaruhi oleh jumlah proses mencurigakan yang dipilih untuk proses *dumping* dan pemindaian. Dari sample *memory* yang dianalisis waktu eksekusi kurang dari 10 menit untuk setiap *snapshot* yang diuji.

Tabel 6. Evaluasi kecepatan analisis dan ekstraksi data

Memory *Snapshot*	Execution time
Wannacry - 109	9 min 53 sec
Wannacry - 149	3 min 56 sec
Teslacrypt - 403	3 min 29 sec
Teslacrypt - 452	3 min 14 sec

5. KESIMPULAN

Penelitian ini mengusulkan alur kerja forensik memori yang efisien dengan mengombinasikan penyaringan perilaku proses, *dumping* spesifik proses, pemindaian YARA, serta verifikasi intelijen ancaman untuk mendeteksi *ransomware*. *Workflow* ini berhasil mengidentifikasi ancaman seperti *WannaCry* dan *Teslacrypt* dengan penggunaan sumber daya yang minimal, serta mendukung analisis lanjutan melalui penyediaan *dump file* dari memori yang dapat diteruskan ke tim analisis *malware* untuk pemeriksaan lebih mendalam. Dari pengujian pada sampel *file* memori yang digunakan, skrip pada penelitian ini dapat mendeteksi proses *malware* dengan cepat (<10 menit) dengan hasil akurasi rata-rata lebih dari 97%, dan efisiensi triase rata-rata kurang dari 3%.

Arah penelitian selanjutnya dapat mengeksplorasi integrasi *workflow* ini dengan pendekatan berbasis *machine learning* atau LLM seperti VolGPT, guna meningkatkan kemampuan identifikasi proses mencurigakan di luar metode berbasis aturan. Kombinasi antara *whitelisting* dengan model pembelajaran dapat memperbaiki deteksi terhadap ancaman baru yang belum dikenal.

Selain itu, perluasan aturan YARA yang digunakan pada tahap pemindaian akan secara signifikan meningkatkan cakupan deteksi. Pengayaan aturan yang lebih komprehensif dan terkini—dengan menargetkan lebih banyak *signature malware* serta pola perilaku—akan memperkuat kemampuan sistem dalam mendeteksi ancaman tingkat lanjut maupun varian langka yang tersembunyi di dalam *dump* memori.

REFERENSI

[1] D. B. Oh, D. Kim, D. Kim, and H. K. Kim, "volGPT: Evaluation on triaging ransomware process in memory forensics with Large Language Model," *Forensic Science International: Digital Investigation*, vol. 49, p. 301756, July 2024, doi: 10.1016/j.fsidi.2024.301756. Available: https://linkinghub.elsevier.com/retrieve/pii/S2666281724000751.

[2] Y. Dehfouli and A. H. Lashkari, "Memory Analysis for Malware Detection: A Comprehensive Survey Using the OSCAR Methodology," *ACM Computing Surveys*, vol. 58, no. 4, art. 86, pp. 1–58, Oct. 2025, doi: 10.1145/3764580.

[3] A. A. S. Akshay, V. Pavithran, and S. R. Syam, "APT Detection Using Memory Forensics: An Empirical Study," in *Proc. 2024 15th Int. Conf. on Computing, Communication and Networking Technologies* (ICCCNT), Kamand, India, Jun. 2024, doi: 10.1109/ICCCNT61001.2024.10724662.

[4] A. A. Thakar, K. Kumar, and B. Patel, "Next Generation Digital Forensic Investigation Model (NGDFIM) – Enhanced, Time Reducing and Comprehensive Framework," *Journal of Physics: Conference Series*, vol. 1767, no. 1, p. 012054, 2021, doi: 10.1088/1742-6596/1767/1/012054.

[5] K. Yildirim, M. E. Demir, T. Keles, A. M. Yildiz, S. Dogan, and T. Tuncer, "A YARA-based approach for detecting cyber security attack types," *Firat Univ. J. Exp. Comput. Eng.*, vol. 2, no. 2, pp. 55–68, 2023, doi: 10.5505/fujece.2023.09709.

[6] I. Hamid and M. M. H. Rahman, "A Comprehensive Literature Review on Volatile Memory Forensics," *Electronics*, vol. 13, no. 15, art. 3026, Jul. 2024, doi: 10.3390/electronics13153026.

[7] A. Arfeen, M. A. Khan, O. Zafar, and U. Ahsan, "Virtual Machine (VM) Memory Dumps for Ransomware Forensics." Harvard Dataverse, 2020. doi: 10.7910/DVN/YVL3CW..

[8] R. Lee, M. Pilkington, Hunt Evil: Your Practical Guide to Threat Hunting, Poster, SANS DFIR Faculty, 10 Juni 2024. Online: https://www.sans.org/posters/hunt-evil.

[9] Filescan.io, fsYara: YARA *rule*s used as part of the *File*scan.io service, GitHub Repository, 2025. Online: https://github.com/*file*scanio/fsYara/.