JUDUL

Analisis Malware Menggunakan *Static Analysis, Dynamic Analysis,* dan *Reverse Engineering* pada *Honeypot*

SKRIPSI

DINA YULIASTI 2207421002

PROGRAM STUDI TEKNIK MULTIMEDIA DAN JARINGAN JURUSAN TEKNIK INFORMATIKA DAN KOMPUTER POLITEKNIK NEGERI JAKARTA

2026

JUDUL

Analisis Malware Menggunakan *Static Analysis, Dynamic Analysis,* dan *Reverse Engineering* pada *Honeypot*

SKRIPSI

DINA YULIASTI / 2207421002

PROGRAM STUDI TEKNIK MULTIMEDIA DAN JARINGAN JURUSAN TEKNIK INFORMATIKA DAN KOMPUTER POLITEKNIK NEGERI JAKARTA

2026

SURAT PERNYATAAN BEBAS PLAGIARISME

SURAT PERNYATAAN BEBAS PLAGIARISME

Saya yang bertanda tangan di bawah ini:

Nama	: Dina Yuliasti
NIM	: 2207421002
Jurusan/Program Studi	: Teknik Informatika dan Komputer/Teknik Multimedia dan Jaringan
Judul Skripsi	: Analisis Malware Menggunakan *static analysis*, *dynamic analysis*, dan *reverse engineering* pada *Honeypot*

Menyatakan dengan sebenarnya bahwa skripsi ini benar – benar merupakan hasil karya saya sendiri, bebas dari peniruan terhadap karya dari orang lain. Kutipan pendapat dan tulisan orang lain ditunjuk sesuai dengan cara-cara penulisan karya ilmiah yang berlaku.
Apabila di kemudian hari terbukti atau dapat dibuktikan bahwa dalam skripsi ini terkandung ciri-ciri plagiat dan bentuk-bentuk peniruan lain yang dianggap melanggar peraturan, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Depok, 2026

Yang membuat pernyataan

Dina Yuliasti
NIM. 2207421002

LEMBAR PENGESAHAN

© Hak Cipta milik Politeknik Negeri Jakarta

Hak Cipta :
1. Dilarang mengutip sebagian atau seluruh karya tulis ini tanpa mencantumkan dan menyebutkan sumber :
a. Pengutipan hanya untuk kepentingan pendidikan, penelitian, penulisan karya ilmiah, penulisan laporan, penulisan kritik atau tinjauan suatu masalah.
b. Pengutipan tidak merugikan kepentingan yang wajar Politeknik Negeri Jakarta
2. Dilarang mengumumkan dan memperbanyak sebagian atau seluruh karya tulis ini dalam bentuk apapun tanpa izin Politeknik Negeri Jakarta

LEMBAR PENGESAHAN

Skripsi diajukan oleh:

Nama	: Dina Yuliasti
NIM	: 2207421002
Program Studi	: Teknik Multimedia dan Jaringan
Judul Skripsi	: Analisis Malware menggunakan Static Analysis, Dynamic Analysis, dan Reverse Engineering pada Honeypot

Telah diuji oleh tim penguji dalam Sidang Skripsi pada hari Selasa, Tanggal 13, Bulan Januari, Tahun 2026 dan dinyatakan LULUS.

Disahkan oleh:

Pembimbing I	Ik Muhamad Malik Matin, S.Kom., M.T.	( )
Penguji I	Asep Kurniawan, S.Pd., M.Kom.	( )
Penguji II	Fachroni Arbi Murad, S.Kom., M.Kom.	( )
Penguji III	Andika Riyadi Jasril, M.Pd.T	( )

Mengetahui:
Jurusan Teknik Informatika dan Komputer
Ketua

Dr. Anita Hidayati, S.Kom., M.Kom.
NIP. 197908032003122003

Dipindai dengan CamScanner

KATA PENGANTAR

Assalamualaikum Wr., Wb.

Puji dan syukur selalu kepada Tuhan Yang Maha Esa atas rahmat dan karunia-Nya sehingga penulis dapat menyelesaikan skripsi yang berjudul "Analisis Malware Menggunakan *Static Analysis*, *Dynamic Analysis*, dan *Reverse Engineering* pada *Honeypot*" dengan baik atas bantuan, motivasi, dan dukungan dari berbagai pihak. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. kepada Ibu Dr. Anita Hidayati, S.Kom., M.Kom., selaku Ketua Jurusan, penulis mengucapkan terima kasih.
2. Bapak Iik Muhammad Malik Matin, S.Kom., M.T. selaku pembimbing penulis yang telah banyak membantu, mendukung, dan memberikan masukan serta saran kepada penulis selama mengerjakan skripsi ini hingga selesai.
3. Teristimewa penulis mengucapkan terima kasih yang sebesar-besarnya kepada kedua orang tua tercinta yang telah memberikan banyak dukungan, doa dan cinta tanpa syarat yang diberikan kepada penulis selama proses penyelesaian skripsi.
4. Sahabat dan teman-teman yang telah banyak membantu dan memberikan dukungan dalam pengerjaan skripsi.
5. Seluruh pihak yang telah terlibat, memberikan doa serta dukungan dalam proses penyusunan skripsi ini yang tidak dapat penulis sebutkan satu per satu.
6. Teruntuk Dina Yuliasti, anak perempuan pertama dan harapan orang tua. Terima kasih telah bertahan, tidak menyerah, dan terus berjalan melewati semua tantangan. Perjalanan ini belum selesai, tapi kamu mampu melewatinya. Ingat, kamu berhak bahagia dan bermimpi.

Akhir kata penulis mengucapkan banyak terima kasih kepada semuanya. Penulis menyadari bahwa dalam penyusunan skripsi ini masih banyak terdapat kekurangan dan keterbatasan, oleh karena itu penulis memohon maaf atas ketidaksempurnaan ini. Semoga skripsi yang ditulis ini bermanfaat dan menjadi motivasi untuk penelitian selanjutnya dan bagi pembaca.

Wassalamualaikum Wr., Wb.

Depok, Juni 2026

Penulis

SURAT PERNYATAAN PERSETUJUAN PUBLIKASI SKRIPSI UNTUK KEPENTINGAN AKADEMIS

SURAT PERNYATAAN PERSETUJUAN PUBLIKASI SKRIPSI UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademik Politeknik Negeri Jakarta, saya bertanda tangan dibawah ini:

Nama	: Dina Yuliasti
NIM	: 2207421002
Jurusan/Program Studi	: Teknik Informatika dan Komputer/Teknik Multimedia dan Jaringan

Demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada Politeknik Negeri Jakarta Hak Bebas Royalti Non-Eksklusif atas karya ilmiah saya yang berjudul:

Analisis Malware menggunakan *Static Analysis*, *Dynamic Analysis*, dan *Reverse Engineering* pada *Honeypot*

Dengan Hak Bebas Royalti NonEksklusif ini Politeknik Negeri Jakarta Berhak menyimpan, mengalihmediakan/formatkan, mengelola dalam bentuk pangkalan data (*database*), merawat, dan mempublikasikan skripsi saya tanpa meminta izin dari saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.
Demikian pernyataan ini saya buat dengan sebenarnya.

Depok, 2026

Yang menyatakan

Dina Yuliasti
NIM.2207421002

Analisis Malware Menggunakan Static Analysis, Dynamic Analysis, dan Reverse Engineering pada Honeypot

ABSTRAK

Malware Windows merupakan ancaman keamanan siber yang terus berkembang dengan varian baru bermunculan setiap harinya, sementara penelitian terdahulu masih terbatas pada satu atau dua metode analisis serta bergantung pada dataset publik yang bersifat statis dan kurang mencerminkan pola serangan terkini. Penelitian ini bertujuan untuk mengembangkan honeypot sebagai sarana pengumpulan sampel malware Windows secara real-time, serta melakukan analisis mendalam terhadap sampel tersebut menggunakan tiga metode terintegrasi, yaitu Static Analysis, Dynamic Analysis, dan Reverse Engineering, guna menghasilkan kompilasi Indicators of Compromise (IoC) yang komprehensif. Studi kasus dilakukan terhadap lima sampel malware yang ditangkap oleh honeypot Dionaea pada platform T-Pot, terdiri atas dua sampel bertipe Launcher dan tiga varian WannaCry, yang sebelumnya diverifikasi menggunakan VirusTotal. Static analysis menggunakan PEStudio, Detect-It-Easy (DIE), dan FLOSS berhasil mengidentifikasi struktur PE, toolchain kompilasi yang identik pada seluruh sampel, evolusi tingkat obfuscation (entropi rendah pada Sampel 1–3 dan tinggi pada Sampel 4–5), serta domain C2 hardcoded yang menyerupai mekanisme kill-switch WannaCry tanpa risiko eksekusi. Dynamic analysis melalui Process Monitor, Process Explorer, Wireshark, dan FakeNet mengungkap perilaku runtime aktual, termasuk network scanning agresif menuju port 445 (SMB), upaya eksploitasi SMBv1/EternalBlue oleh proses mssecsvr.exe, serta pola registry enumeration dan masquerading pada berbagai proses. Reverse Engineering menggunakan Ghidra berhasil membongkar logika dropper dan mekanisme konstruksi path dinamis melalui fungsi ekspor Play Game, yang tidak terlihat pada kedua metode sebelumnya. Hasil penelitian menunjukkan bahwa ketiga metode bersifat saling melengkapi, di mana kombinasi ketiganya menghasilkan cakupan IoC yang jauh lebih komprehensif dibandingkan penggunaan satu atau dua metode saja, sehingga dapat dijadikan acuan deteksi dan respons insiden siber terhadap malware Windows canggih seperti WannaCry.

Kata kunci: *Malware Analysis, Static Analysis, Dynamic Analysis, Reverse Engineering, Honeypot, Indicators of Compromise, Windows Malware, WannaCry.*

DAFTAR ISI

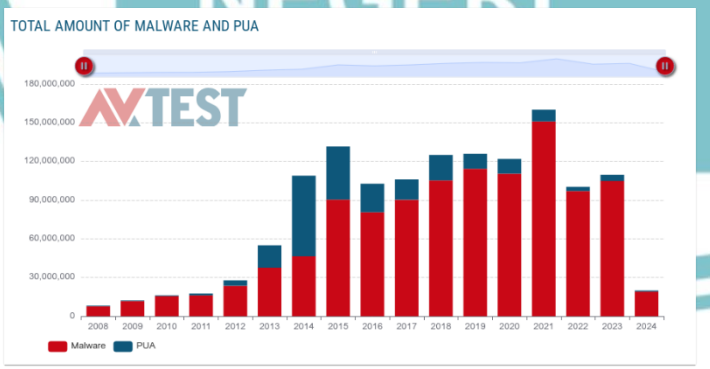
DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Malware merupakan ancaman keamanan siber yang jahat dan paling kritis, yang secara sistematis menargetkan sistem operasi Windows, *platform* yang mendominasi sekitar 73,38% pasar perangkat komputasi global di lingkungan personal, *enterprise*, dan infrastruktur kritis (StatCounter, 2024). Jenis serangan ini mencakup ransomware yang mengenkripsi data berharga secara massal untuk memeras korban (Kaspersky, 2024), trojan downloader yang mengunduh *payload* tambahan secara diam-diam, serta *infostealer* canggih seperti Lumma Stealer yang mencuri kredensial perbankan dan akun korporat, mengakibatkan kerugian ekonomi mencapai ratusan miliar rupiah pada tahun 2023-2024 (Asosiasi Penyelenggara Jasa Internet Indonesia, 2024). Menurut AV-TEST Institute, sebuah lembaga independen yang memantau ancaman siber secara global, lebih dari 450.000 malware dan program tidak diinginkan (Potentially Unwanted Application/PUA) baru terdaftar setiap hari di basis data mereka, dengan jumlah ini terus bertambah karena aktivitas penyerang yang konstan (AV-TEST Institute, 2024). *Platform* Windows tercatat sebagai salah satu target utama malware karena dominasi penggunaannya di lingkungan desktop/laptop serta luasnya serangan yang sering dieksploitasi dalam berbagai teknik malware.



Gambar 1. 1 Statistik Total Malware dan Potentially Unwanted Applications

Pada Gambar menunjukkan perkembangan jumlah total malware dan *Potentially Unwanted Applications* (PUA) secara global yang tercatat oleh AV-TEST Institute sejak tahun 2008 hingga 2024. Grafik disajikan dalam bentuk batang bertumpuk (stacked bar chart), di mana warna merah merepresentasikan jumlah malware dan warna biru menunjukkan jumlah PUA.

Berdasarkan data yang dipublikasikan oleh analisis antivirus independen di tahun 2025 sekitaran 87,4% dari deteksi malware terjadi pada windows, sedangkan pada mac OS mencatat sekitar 12,6% dari total kasus malware yang terdeteksi.Penelitian sebelumnya menghadapi beberapa tantangan dan keterbatasan yang signifikan. Pertama, berkaitan dengan sumber data, sebagaimana yang telah dilakukan oleh Yusirwan et al. (2020) dan Guo et al. (2023), banyak penelitian sebelumnya bergantung pada dataset publik seperti Virus Share, Malware Bazaar, atau Contagiodump yang bersifat statis dan sudah tersedia secara terbuka, sehingga menimbulkan permasalahan terhadap tingkat kebaruan sampel yang dianalisis. Di sisi lain, penelitian yang dilakukan oleh R.M. Muhammad et al. (2021) dan Holbel et al. (2024) memanfaatkan honeypot, namun fokus penelitian mereka lebih mengarah pada perancangan sistem keamanan jaringan dan proses *threat hunting*, bukan pada analisis teknis mendalam terhadap sampel malware itu sendiri. Kedua, keterbatasan metodologi analisis yang dipakai sebagaimana dilakukan oleh Eliando et al. (2022), Guo et al. (2023), dan Damaševičius et al. (2025). Penelitian terdahulu cenderung melakukan satu jenis analisis saja, baik itu *static analysis*, *dynamic analysis*, maupun *reverse engineering*, tanpa melakukan kombinasi komprehensif dari ketiga metode tersebut untuk membongkar logika internal malware secara mendalam.

Static analysis menjadi langkah penting pertama yang memungkinkan ekstraksi informasi struktural dari *Portable Executable* (PE) header tanpa eksekusi berisiko, termasuk analisis *Import Address Table (IAT)*, *section permissions*, overlay data, serta *suspicious strings* yang mengindikasikan C2 domains atau *registry run keys.* Tools seperti PEiD untuk packer identification dan *Strings* untuk *hardcoded IOC extraction* memberikan *baseline analysis* yang cepat namun terbatas pada *surfacelevel artifacts.*

Dynamic analysis melengkapi keterbatasannya dengan monitoring perilaku *runtime* dalam lingkungan sandbox terisolasi, mengamati API calls mencurigakan seperti *CreateRemoteThread, registry* modifications di *HKLM\Software\Microsoft\Windows\CurrentVersion\Run, network beaconing ke external IPs, serta file drops di %AppData% atau %Temp% directories. Platform* tools Process Monitoring dan Process Explorer secara spesifik mencatat *behavioral indicators* yang tak terlihat pada *static examination*, seperti *process injection* dan *privilege escalation attempts.*

Reverse engineering melalui *disassembly* dan *decompilation* kode *assembly* merupakan tahap lanjutan analisis untuk mengungkap logika tersembunyi seperti *anti-analysis techniques (timing checks, red pill), custom crypters*, dan *hardcoded C2 infrastructure* yang luput dari dua metode sebelumnya, memerlukan tools profesional seperti Ghidra untuk *control flow reconstruction*.

Penelitian ini mengambil sampel malware terbaru yang dihasilkan langsung dari honeypot tpot, menangkap varian autentik dari penyerang nyata dalam real-time, berbeda dengan penggunaan dataset publik yang bersifat statis seperti VirusShare atau Malware Bazar yang sudah outdated dalam 24-48 jam. Sampel *Honeypot* merepresentasikan *attack patterns* terkini dengan geographical relevance ke Indonesia, memberikan ground truth data untuk validasi ketiga metode analisis secara komprehensif.

Penelitian ini menerapkan analisis sistematis terhadap sampel malware Windows terkini dari honeypot menggunakan tiga metode terintegrasi secara berurutan: *Static Analysis* dengan Detect-It-Easy (DIE), strings/Floss, PEStudio, dan Virustotal untuk *structural profiling* tanpa *execution risk*. *Dynamic Analysis* melalui Process monitor (Procmon), Process explorer, Wireshark, dan juga Fakenet. *Reverse Engineering* menggunakan Ghidra (free) untuk *full disassembly, function identification, string references,* dan *control flow graph reconstruction* guna ekstraksi IOC tingkat tinggi seperti *mutex names, registry persistence paths,* dan *obfuscated C2 communications.*

1.2. Perumusan Masalah

Melalui latar belakang tersebut, maka rumusan masalahnya adalah:

1. Bagaimana mengembangkan honeypot untuk mengumpulkan malware?
2. Bagaimana melakukan analisis menggunakan *static analysis, dynamic analysis*, dan *reverse engineering* terhadap sampel malware yang dihasilkan dari honeypot.

1.3. Batasan Masalah

Batasan masalah ini bertujuan untuk memperjelas ruang lingkup penelitian sehingga pembahasan tidak terlalu luas. Adapun batasan masalah dalam penelitian ini adalah sebagai berikut:

1. Sampel malware terbatas pada varian Windows yang ditangkap honeypot dan dipilih sebanyak 5 sampel
2. Analisis static menggunakan tools DIE, PEStudio, VirusTotal,dan strings/Floss
3. Analisis dinamik menggunakan Procmon, Process Explorer, Wireshark, dan Fakenet
4. *Reverse Analysis* menggunakan tools Ghidra untuk *disassemble* dan *decompilation.*
5. Penelitian ini Tidak membahas pengembangan tools baru atau integrasi dengan SIEM production systems
6. Lingkungan pengujian dibatasi hanya pada windows 11
7. Penelitian ini Tidak membahas *behavioral machine learning classification*
8. Penelitian ini tidak menggunakan YARA rules untuk mendeteksi malware.

1.4. Tujuan dan Manfaat

1.4.1. Tujuan

Tujuan dari penelitian ini adalah sebagai berikut:

1. Mengembangkan honeypot untuk mengumpulkan malware
2. Melakukan analisis menggunakan *static analysis, dynamic analysis*, dan *reverse engineering* terhadap sampel malware yang dihasilkan dari honeypot

1.4.2. Manfaat

Adapun manfaat dari penelitian ini adalah sebagai berikut.

1. Menyediakan referensi analisis malware berbasis honeypot dengan *dataset real-time* untuk penelitian *cybersecurity* akademik
2. Memperkaya literatur tentang sinergi *static-dynamic-reverse engineering* pada varian Windows malware terkini
3. Memberikan studi kasus praktis tools analisis yang direkomendasikan komunitas peneliti malware
4. Menghasilkan kompilasi IoC *actionable* (*registry keys, mutexes, C2 domains*)
5. Memberikan panduan analisis malware bagi praktisi *cyber security* Indonesia menghadapi ancaman Windows dominan
6. Mendukung validasi honeypot *deployment* untuk pengumpulan sampel malware berkualitas tinggi di organisasi lokal
7. Menjadi bahan evaluasi pada sampel malware yang dihasilkan dari honeypot sebagai sumber data analisis malware berbasis serangan nyata.

1.5. Sistematika Penulisan

Adapun sistematika penulisan dari laporan ini adalah sebagai berikut

A. BAB I PENDAHULUAN

Bab I ini menjelaskan latar belakang penelitian, perumusan masalah, batasan, tujuan, dan manfaat penelitian, serta sistematika penulisan.

B. BAB II TINJAUAN PUSTAKA

Bab II berisikan penjelasan mengenai landasan teori atau kajian ilmu yang berhubungan dengan berbagai pokok pikiran topik penyusunan skripsi ini yang relevan dari sumber yang valid.

C. BAB III METODE PENELITIAN

Bab III menjelaskan rancangan penelitian dan alur sistem analisis malware berbasis honeypot yang diusulkan, sumber data dan sampel yang digunakan, teknik pengumpulan data dari lingkungan honeypot dan lab analisis, teknik analisis data secara kualitatif dan kuantitatif terhadap hasil

static, *dynamic*, dan *reverse engineering*, serta jadwal pelaksanaan penelitian dan rincian biaya yang diperlukan.

D. BAB IV

Bab IV menyajikan hasil penelitian dan pembahasan dari analisis malware menggunakan tiga metode terintegrasi. Bab ini terdiri dari sub-sub bab mencakup analisis kebutuhan, perancangan sistem, implementasi sistem, dan pengujian dengan *static analysis*, *dynamic analysis*, dan *reverse engineering* serta evaluasi pengujian berupa kompilasi IoC dari hasil ketiga metode analisis.

E. BAB V PENUTUP

Bab V berisikan kesimpulan atau hasil akhir dari penelitian yang telah dilakukan, serta saran untuk penelitian berikutnya.

BAB II
TINJAUAN PUSTAKA

2.1. Penelitian Sejenis

No	Peneliti & Judul	Metode & Objek Utama	Keterbatasan Penelitian Sebelumnya	Kebaruan Penelitian
1	S. Yusirwan dkk., 2020	Menggunakan *static* dan *dynamic analysis* untuk menganalisis ransomware (LockBit 2.0) pada Windows; fokus pada pola infeksi dan *persistence*, tanpa honeypot sebagai sumber dataset.	Hanya dua metode (*static & dynamic*); sampel malware tidak dijelaskan sebagai hasil honeypot realtime; tidak ada *reverse engineering* mendalam terhadap kode, dataset terbatas dan kurang terukur kebaruannya.	Menggunakan tiga metode sekaligus (*static*, *dynamic*, *reverse engineering*) dan dataset sampel malware Windows terbaru dari honeypot, sehingga pola serangan yang dianalisis lebih mutakhir dan komprehensif.
2	R.M. Muhammad dkk., 2021	Membangun sistem keamanan jaringan dengan honeypot untuk	Fokus utama pada perancangan sistem honeypot & keamanan	Penelitian ini tidak merancang sistem baru, tetapi memanfaatkan honeypot hanya

		menjebak serangan dan mendapatkan informasi malware, fokus pada rancangan sistem dan integrasi keamanan jaringan	jaringan, bukan analisis teknis detail sampel malware, tidak membahas *static/dynamic/reverse analysis* secara terstruktur.	sebagai penghasil dataset sampel malware terbaru, lalu melakukan analisis teknis mendalam (*static*, *dynamic*, *reverse*) terhadap setiap sampel.
3	E. Eliando dkk., 2022	Analisis LockBit 2.0 ransomware pada Windows menggunakan kombinasi *static dan dynamic analysis* untuk melihat tahapan infeksi dan *mekanisme persistence*.	Objek terbatas pada satu keluarga ransomware, tidak ada honeypot untuk memperoleh sampel baru, *reverse engineering* hanya disentuh sebagai bagian kecil *static*, bukan sebagai pilar metode utama.	Objek penelitian ini adalah beragam sampel malware Windows terbaru yang ditangkap honeypot, bukan satu varian saja, *reverse engineering* dijadikan metode utama ketiga untuk membongkar logic tersembunyi.

4	Guo dkk., 2023	Mengembangkan sistem deteksi malware Windows	Fokus pada deteksi otomatis (ML), bukan analisis manual mendalam;	Penelitian ini tidak menggunakan ML dan tidak membangun sistem
		berbasis *static analysis (PE header,imported DLL, API calls)* dengan berbagai model machine learning; membuat dataset besar 27.920 sampel PE	hanya menggunakan *static analysis*, tanpa *dynamic* dan *reverse* sampel berasal dari koleksi umum, bukan honeypot realtime.	deteksi, fokus pada analisis mendalam per sampel menggunakan tiga metode dan dataset honeypot terkini, sehingga lebih kuat untuk studi forensik dan pola serangan aktual.
5	Holbel dkk., 2024	Menggunakan honeypot virtual untuk threat hunting dan pengumpulan malware, lalu dianalisis dan *direverse engineering*, terintegrasi dengan SIEM (*Elastic Stack*) untuk visualisasi *attacker behavior.*	Tujuan utama adalah *threat hunting* & SIEM, bukan komparasi terstruktur *static–dynamic–reverse* terhadap dataset Windows secara kuantitatif, tidak fokus spesifik pada keunggulan dataset terbaru sebagai variabel penelitian	Kontribusi ini memfokuskan pada Analisis perbandingan tiga metode terhadap dataset malware Windows terbaru dari honeypot sebagai poin utama kebaruan, tanpa membangun SIEM, sehingga metodologisnya lebih tajam ke sisi analisis.

6.	Bombardieri et al., 2015	Mengusulkan arsitektur *honeypot high-interaction* (HERESy) untuk	Fokus kuat pada arsitektur sistem honeypot dan *workflow logging, reverse engineering* lebih	Penelitian ini mengambil *spirit honeypot-powered analysis*, tetapi menggeser fokus ke studi komparatif tiga
7.	Damaševičius dkk., 2025	Mengembangkan framework otomatis yang menggabungkan static dan *dynamic analysis* untuk ekstraksi fitur (opcode, EMBER, APIbased) dan digunakan untuk klasifikasi malware dengan ML	Fokus pada otomatisasi dan fitur ML, tidak melakukan *reverse engineering* manual, tidak spesifik ke dataset honeypot Windows terkini, tujuan akhir adalah *classifier*, bukan pemetaan pola serangan per sampel.	Penelitian ini melengkapi lini ini di sisi forensik manual dan eksploratif, dengan tiga metode, tanpa ML – sehingga memberikan insight kualitatif dan IoC detail dari sampel terbaru honeypot.

2.1.1 Penelitian *Static Analysis* Malware

Penelitian yang dilakukan Guo beserta timnya (Guo et al., 2023). mengembangkan sistem deteksi malware Windows berbasis *static analysis* dengan mengekstraksi berbagai fitur dari berkas PE seperti *header*, *imported DLL*, dan API calls, kemudian melatih beberapa model *machine learning*. Tools yang digunakan meliputi PE parser dan framework ekstraksi fitur otomatis, sedangkan evaluasi dilakukan pada dataset 27.920 sampel PE (*benign dan malicious*). Hasilnya menunjukkan bahwa kombinasi beberapa fitur statis mampu memberikan akurasi

deteksi tinggi. Keterbatasan penelitian ini adalah fokus pada klasifikasi otomatis, tanpa analisis manual mendalam terhadap perilaku malware dan tanpa integrasi *dynamic analysis* maupun *reverse engineering*. Selain itu, dataset berasal dari koleksi publik yang tidak mencerminkan malware paling baru.

2.1.2 Penelitian *Dynamic Analysis* Malware

Penelitian yang dilakukan oleh Yusirwan beserta timnya menerapkan analisis malware dengan memadukan *static* dan *dynamic analysis* untuk meneliti ransomware pada sistem Windows. Analisis statis dipakai untuk memeriksa struktur berkas serta tanda tanda, sementara analisis dinamis memanfaatkan *sandbox* guna memantau proses, perubahan berkas, dan aktivitas registry selama ransomware dijalankan (Yusirwan et al., 2020). Temuan penelitian mengindikasikan bahwa perpaduan kedua metode tersebut dapat memetakan tahapan infeksi serta mekanisme persistence. Namun, penelitian ini tidak menggunakan honeypot sebagai sumber sampel, tidak mencakup *reverse engineering* secara mendalam, dan tidak memberi perhatian khusus pada kebaruan kumpulan data.

Penelitian yang dilakukan oleh Eliando beserta timnya. menganalisis LockBit 2.0 ransomware pada sistem Windows dengan pendekatan *static* dan *dynamic analysis* untuk mengidentifikasi pola infeksi, teknik enkripsi, dan dampak terhadap system (Eliando et al., 2022, Cogito Smart Journal). Tools yang digunakan mencakup PE *analyzer, process monitor,* dan *network monitor*. Penelitian ini memperlihatkan detail perilaku satu keluarga ransomware secara mendalam, tetapi objeknya terbatas pada satu varian, tidak memanfaatkan honeypot, dan belum membandingkan hasil analisis dengan *reverse engineering* di level kode.

2.1.3 Penelitian *Reverse Engineering* Malware

Penelitian yang dilakukan oleh Bombardieri dan Capretti memperkenalkan HERESy, sebuah arsitektur honeypot *high interaction* yang dirancang untuk mendukung malware *reverse engineering*. Sistem ini menangkap sampel malware dan melacak *lifecycle* serangan untuk dianalisis lebih lanjut di level kode (Bombardieri & Capretti, 2015, arXiv:1508.02113). Tools *reverse engineering* seperti *disassembler* dan

debugger digunakan untuk menelusuri perilaku internal malware. Fokus utama penelitian adalah desain arsitektur dan *workflow logging*, bukan komparasi terukur antara *static*, *dynamic*, dan *reverse analysis* pada dataset yang sama.

2.1.4 Penelitian tentang *Honeypot* untuk pengumpulan sampel Malware

Penelitian yang dilakukan oleh Holbel beserta timnya. memanfaatkan *virtualized* honeypots untuk *threat hunting*, pengumpulan malware, dan analisis, kemudian mengintegrasikan hasilnya dengan SIEM (Elastic Stack) untuk visualisasi aktivitas penyerang (Holbel et al., 2024, Issues in Information Systems). Penelitian ini menunjukkan bagaimana honeypot dapat menjadi sumber *threat intelligence* dan sarana *reverse engineering*, tetapi tidak berfokus pada komparasi tiga metode analisis (*static*, *dynamic*, *reverse*) terhadap satu set dataset Windows secara kuantitatif.

2.1.5 Penelitian Multi-Metode Analisis Malware

Penelitian yang dilakukan oleh Damaševičius beserta timnya mengembangkan framework FETA untuk ekstraksi fitur malware secara sistematis dengan menggabungkan *static* dan *dynamic analysis*, kemudian menggunakan fitur tersebut dalam model klasifikasi *machine learning* (Damaševičius et al., 2025). Penelitian ini menunjukkan bahwa kombinasi multi fitur meningkatkan performa deteksi, namun tidak melakukan *reverse engineering* manual dan tidak menggunakan dataset yang berasal dari honeypot Windows terkini. Fokus utama adalah performa *classifier*, bukan pemetaan detail pola serangan per sampel.

Selain aspek honeypot, penelitian Holbel beserta timnya. juga dapat dikategorikan sebagai multi metode karena memanfaatkan *logging*, analisis perilaku, dan *reverse engineering* untuk memahami serangan yang tertangkap honeypot virtual (Holbel et al., 2024). Namun, pendekatan ini lebih diarahkan pada *threat hunting* operasional dan integrasi dengan SIEM daripada analisis komparatif sistematis antara *static*, *dynamic*, dan *reverse analysis* terhadap dataset yang sama.

2.1.6 Posisi Penelitian (Research Gap)

Dari studi pustaka diatas, dapat disimpulkan beberapa Gap Utama yaitu:

1. Banyak penelitian hanya menggunakan 1–2 metode analisis (*static/dynamic*) atau hanya menjelaskan konsep *reverse engineering*, sementara kombinasi *static, dynamic*, dan *reverse engineering* secara sistematis pada dataset yang sama masih jarang dilakukan.
2. Sebagian besar penelitian yang menggunakan dataset besar memanfaatkan repository publik (Virus Share, Malware Bazaar, dan dataset PE GitHub) yang berisi sampel sejak 2015–2019, sehingga berpotensi tidak merepresentasikan malware Windows terbaru.
3. Penelitian honeypot seperti Muhammad (2021) dan Holbel (2024) menunjukkan potensi honeypot sebagai sumber data dan *threat intelligence*, tetapi tidak memfokuskan diri pada analisis tiga metode yang terukur terhadap sampel malware Windows dari honeypot.

Penelitian ini menempatkan posisi sebagai berikut:

1. Menggunakan dataset sample malware terbaru yang dikumpulkan dari honeypot, sehingga merepresentasikan varian nyata yang menyerang lingkungan jaringan saat ini.
2. Menerapkan tiga metode analisis yaitu *static analysis, dynamic analysis*, dan *reverse engineering* pada dataset yang sama. Kemudian membandingkan hasil dari masing-masing metode dalam menghasilkan *Indicator Of Compromise* (IoC).
3. Peneliti berfokus pada analisis mendalam dan IoC *extraction*, bukan pada pengembangan sistem deteksi otomatis (ML) atau arsitektur SIEM.

Dengan demikian, penelitian ini berkontribusi pada *state of the art* analisis malware Windows berbasis honeypot dengan memberikan studi kasus komprehensif tri metode terhadap dataset yang relevan secara temporal dan geografis.

2.2 Malware

Malware merupakan perangkat lunak yang berbahaya yang dibuat untuk merusak, mengganggu, mencuri, atau mendapatkan akses yang tidak sah kedalam sistem dan data. Ciri utama dari malware adalah dieksekusi tanpa izin sah, memanfaatkan

kelemahan sistem, dan sering menyamarkan diri agar tidak mudah terdeteksi. Evolusi malware bergerak dari virus sederhana menuju ransomware, infostealer, dan APT yang menggunakan teknik canggih seperti *fileless execution* dan enkripsi komunikasi (AV-TEST Institute, 2024; Kaspersky, 2024).

Windows menjadi target utama malware karena pangsa pasarnya yang dominan dan *attack surface* yang luas (layanan jaringan, browser, aplikasi pihak ketiga, dll) (StatCounter, 2024; SQ Magazine, 2025). Malware Windows umumnya didistribusikan sebagai berkas *Portable Executable* (PE). Struktur PE mencakup *PE Header*, *section table*, dan berbagai *section* (*.text*, *.data*, *.rdata*, *.rsrc)*. Import *Address Table* (IAT) berisi daftar fungsi API yang diimpor dari DLL *system*, analisis IAT membantu mengidentifikasi fungsi yang disalahgunakan malware misalnya *CreateRemoteThread, VirtualAllocEx* (Infosec, 2024; Microsoft, 2023). Setiap *section* memiliki *permissions (read/write/execute)* yang dapat mengindikasikan perilaku mencurigakan, dan overlay data di luar struktur PE dapat menunjukkan adanya *packer* atau data tambahan tersembunyi (GeeksforGeeks, 2023).

2.2.1 Teknik Evasion dan Obfuscation

Malware menggunakan packing dan encryption untuk menyembunyikan kode aslinya, meningkatkan *entropi file*, dan menyulitkan *static analysis* (Infosec, 2024). *Anti-analysis techniques* meliputi anti *debugging* (*timing checks*, pengecekan *debugger*), anti VM detection, dan anti sandbox yang menyebabkan malware tidak sepenuhnya mengeksekusi *payload* jika mendeteksi lingkungan analisis (Varonis, 2025; Kanj et al., 2026). Untuk *persistence*, malware memanfaatkan Windows *Registry* (*run keys* seperti *HKLM\Software\Microsoft\Windows\CurrentVersion\Run*), *scheduled tasks*, pembuatan service, dan teknik lain agar tetap aktif setelah reboot (Microsoft, 2024).

2.2.2 Indication of Compromise (IoC)

IoC adalah artefak teknis yang mengindikasikan adanya kompromi, seperti *hash file, path, IP/domain, registry keys, mutex names*, dan pola perilaku tertentu

(Fortinet, 2024; Microsoft, 2024). IoC dapat dibagi menjadi file based, network indicators, registry based, dan behavioral indicators.

2.2.3 Metode Deteksi Malware Konvensional

Metode konvensional mencakup *signature-based detection*, yang mencocokkan hash atau pola *byte file* dengan *signature database*; efektif untuk malware yang sudah dikenal, tetapi lemah terhadap *zeroday* dan *polymorphic malware* (SOC Prime, 2024)

Heuristic based detection menggunakan aturan dan *pattern matching* untuk menilai potensi berbahaya sebuah program, sedangkan *behavior based detection* memantau aktivitas *runtime* dan anomali perilaku, misalnya enkripsi massal atau koneksi mencurigakan ke domain tertentu (Fortinet, 2024)

2.3 *Static Analysis*

Static analysis adalah proses menganalisis malware tanpa mengeksekusi file tersebut, dengan cara memeriksa struktur, metadata, dan konten biner. Tujuannya untuk memahami karakteristik awal malware secara aman dan cepat, misalnya dengan melihat format PE, header, dan strings (Infosec, 2024; GeeksforGeeks, 2023).

Kelebihan *static analysis* antara lain adalah tidak membutuhkan eksekusi sehingga relatif aman, dapat dilakukan dengan cepat, dan memungkinkan inspeksi menyeluruh terhadap seluruh kode. Keterbatasannya, teknik ini mudah terhambat oleh *packing*, *enkripsi*, dan *obfuscation*, serta tidak dapat mengungkap perilaku *runtime* seperti *process injection* atau komunikasi pada jaringan (Infosec, 2024).

Teknik umum meliputi analisis file PE (PE *header*, *section table*, *import/export table*), *string extraction* untuk menemukan *URL*, *IP*, dan *path*, *entropy analysis* untuk mendeteksi packing, serta identifikasi *packer* yang digunakan. Pemeriksaan *Import Address Table (IAT)* memberikan gambaran fungsi API yang disalahgunakan oleh malware (*Infosec, 2024; Microsoft, 2023)*

2.4 *Dynamic Analysis*

Dynamic analysis adalah analisis malware dengan menjalankan sampel dalam lingkungan sandbox/VM terisolasi untuk memantau perilaku *runtime*. Pendekatan ini mengungkap aktivitas nyata malware seperti proses yang dibuat, file yang dimodifikasi, dan koneksi jaringan (Cuckoo Sandbox, 2024). Kelebihan dinamik analisis adalah mampu mengamati perilaku aktual, menembus *packing/unpacking runtime*, dan memonitor aktivitas jaringan serta perubahan sistem. Keterbatasannya: membutuhkan lingkungan sandbox yang aman, rentan terhadap sandbox evasion dan anti VM, hanya melihat code path yang dieksekusi selama observasi, serta relatif memakan waktu. Beberapa alat dinamik analisis diantaranya adalah Procmon.

2.4.1 *Behavioral Indicators*

Dynamic analysis memantau indikator perilaku seperti pembuatan proses baru, *process injection* misalnya melalui *CreateRemoteThread*, dan struktur *process tree*. Aktivitas file system (*file creation/modification, dropped files di %AppData%* dan *%Temp%)*, perubahan *registry* (terutama *run keys* untuk *persistence*), dan *network activity* (koneksi *IP/port, DNS queries, HTTP(S) traffic, data exfiltration*) juga menjadi fokus (Cuckoo Sandbox, 2024).

2.5 *Reverse Engineering*

Reverse engineering adalah proses mendekonstruksi binary untuk memahami struktur internal, logika, dan algoritma program. Dalam konteks malware, RE digunakan untuk mengungkap fungsi tersembunyi, protokol komunikasi C2, dan teknik anti-analisis yang tidak mudah terlihat pada *static/dynamic analysis* permukaan (Varonis, 2025). Kelebihan *Reverse Engineering* meliputi pemahaman mendalam terhadap logika malware, kemampuan mengekstrak IoC lanjutan misalnya algoritma enkripsi dan kunci, serta dokumentasi detail perilaku. Keterbatasannya adalah kebutuhan *skill tinggi* (assembly, arsitektur CPU), proses yang memakan waktu, dan kompleksitas tinggi terutama pada kode yang heavily obfuscated.

Reverse Engineering dimulai dengan *disassembly*, mengonversi *machine code* menjadi instruksi *assembly*, lalu dilanjutkan dengan *decompilation* ke *pseudocode* tingkat tinggi. Langkah ini membantu analis membaca alur program, memetakan fungsi, dan memahami kontrol aliran (loop, percabangan).

2.5.1 Tools *Reverse Engineering*

Tools yang digunakan dalam melakukan analisis *reverse engineering* Ghidra, suite RE open-source yang menyediakan *disassembler*, *decompiler*, *CFG*, dan *scripting.* (Varonis, 2025). Ghidra digunakan dalam berbagai skenario analisis keamanan, termasuk analisis malware, *vulnerability research*, *binary patching*, dan forensik digital. Tool ini telah menjadi standar industri dalam komunitas *reverse engineering* karena kombinasi fitur yang lengkap dan dukungan komunitas yang aktif.

Ghidra menyediakan kemampuan *disassembly*, *decompilation*, *debugging*, *emulation*, dan *scripting*, menjadikannya aset penting bagi profesional keamanan siber dengan dukungan terhadap berbagai *instruction set processor* dan *executable formats* yang memungkinkan pengguna melakukan analisis perangkat lunak secara mendalam (Cyber Security News, 2025).

2.6 Honeypot

Honeypot adalah sistem atau layanan yang sengaja dibuat menarik bagi penyerang untuk memantau, merekam, dan mempelajari serangan. Konsep ini bagian dari *deception technology*, dengan tujuan utama deteksi dini, pengalihan serangan, dan pengumpulan data ancaman (Holbel et al., 2024; Muhammad et al., 2021). Dionaea adalah honeypot *low interaction* yang dirancang untuk menangkap malware dengan meniru berbagai protokol seperti SMB, HTTP, FTP, dan MSSQL.

Cowrie adalah honeypot SSH/Telnet yang mencatat brute force login dan perintah penyerang, termasuk file malware yang diunduh (Oosterhof, 2026) Keduanya sangat cocok untuk pengumpulan sampel malware Windows secara otomatis dari serangan nyata. Sampel dari honeypot diperoleh secara real time dari penyerang asli, sehingga mencerminkan varian dan teknik serangan terkini serta memiliki relevansi geografis misalnya serangan yang menargetkan Indonesia. Hal ini

berbeda dengan dataset publik yang sering berisi sampel lama atau sudah diproses berkali kali (Holbel et al., 2024; ITS honeypot study).

Dataset publik seperti Virus Share dan Malware Bazaar memiliki kelebihan dari sisi volume dan label, tetapi dapat menjadi *outdated* dan kurang *kontekstual* jika tidak terus diperbarui. Dataset honeypot lebih terbatas jumlahnya, namun memberikan konteks serangan yang autentik, termasuk waktu, sumber IP, dan teknik *eksploit* yang digunakan (VirusTotal, 2024).

BAB 3
METODE PENELITIAN

3.1. Rancangan Penelitian

3.1.1 Desain Arsitektur Sistem

Gambar 3. 1 Topologi Sistem

Dilihat dari Gambar 3.1, penelitian ini menggunakan desain studi kasus eksploratif pada sejumlah sampel malware Windows yang diperoleh dari honeypot. Setiap sampel dianalisis menggunakan tiga metode secara berurutan, kemudian hasilnya dibandingkan dan dikorelasikan. Pendekatan studi kasus dipilih karena memungkinkan penggalian mendalam terhadap karakteristik dan pola serangan pada sampel yang relatif terbatas jumlahnya tetapi tinggi nilai informasinya. Selain itu, penelitian ini bersifat *eksperimental* analitik karena peneliti mengatur lingkungan uji (mesin virtual Windows), mengeksekusi malware dalam kondisi terkendali, serta mengamati dampak dan perilakunya secara sistematis. Lingkungan yang terisolasi memastikan bahwa perubahan yang terjadi pada sistem selama analisis merupakan akibat langsung dari eksekusi sampel malware, sehingga hubungan sebab-akibat dapat ditelusuri dengan lebih jelas.

3.1.2 Tahapan Penelitian

Alur penelitian meliputi langkah-langkah berikut:

1. Studi Literatur

   Dilakukan pengumpulan dan pemahaman landasan teori yang relevan misalnya.

2. Analisis Kebutuhan

   Berdasarkan studi literatur dan penelitian terdahulu, dilakukan analisis kebutuhan berupa arsitektur ataupun perangkat yang dapat digunakan untuk menyelesaikan permasalahan awal.

3. Perancangan dan Pembuatan Sistem

   Pembangunan topologi jaringan virtual, instalasi sistem operasi dan tools yang digunakan, serta konfigurasi integrasi antar-sistem.

4. Pengujian Sistem

   Sistem yang telah dibuat dan terintegrasi dilakukan kemudian dilakukan pengujian Pelaksanaan pengujian berupa serangan terhadap sistem target sesuai dengan skenario ancaman yang telah ditentukan.

5. Analisis dan Evaluasi

   Data log yang terekam selama tahap pengujian kemudian dianalisis. Evaluasi akan dilakukan untuk menilai apakah sistem yang sudah dibuat berhasil mencapai mendeteksi, merespon, dan menampilkan data.

6. Penyusunan Laporan

   Tahap terakhir yaitu mendokumentasikan seluruh proses penelitian, mulai dari perancangan, konfigurasi, hingga hasil analisis data. Kesimpulan akan dibuat berdasarkan perbandingan antara hasil eksperimen dengan tujuan penelitian yang telah ditetapkan sebelumnya.

3.2. Sumber Data

Data yang digunakan dalam penelitian ini terdiri dari:

1. Populasi: Seluruh sampel malware Windows yang tertangkap honeypot TPot/Dionaea selama 2 Bulan.
2. Sampel: 5 file PE Windows unik, dipilih purposive berdasarkan kriteria: format *valid*, *hash unik*, tipe beragam (*ransomware*, *trojan*, *infostealer*).
3. Instrumen Pengumpulan:
   a. *Honeypot* logs (TPot, Dionaea).
   b. File hash (MD5/SHA256) untuk deduplikasi.

3.3. Teknik Pengumpulan Data

Dalam penelitian ini, data dikumpulkan dengan melakukan pengujian langsung pada sistem yang diteliti. Beberapa pengujian yang dilakukan yaitu:

1. Pengumpulan sampel malware menggunakan honeypot seperti Cowrie dan Dionaea yang dikonfigurasi pada server yang terkoneksi ke internet dengan kontrol keamanan yang ketat. *Honeypot* akan mencatat aktivitas serangan dan menyimpan file yang diunduh atau dikirim penyerang ke direktori tertentu.
2. Ekstraksi dan seleksi sampel, di mana file yang terkumpul di honeypot dipindahkan ke lingkungan analisis yang terisolasi. Pada tahap ini dilakukan verifikasi format file (menggunakan *file utility* atau *PE analyzer*) untuk memastikan bahwa file tersebut benar-benar *executable Windows*, serta pengecekan hash untuk menghindari duplikasi. Hanya file yang memenuhi kriteria sampel yang akan masuk ke tahap analisis lebih lanjut.
3. pengumpulan data hasil *static analysis*, dengan menjalankan tools seperti PEStudio, VirusTotal, DIE, dan Floss/strings terhadap setiap sampel. Hasil Akhir berupa informasi *header*, daftar impor, hasil *deteksi packer*, dan strings disimpan dalam bentuk file teks atau laporan terstruktur.
4. Pengumpulan data hasil *dynamic analysis*, yaitu menjalankan setiap sampel menggunakan tools seperti Procmon, Process Explorer, Wireshark,dan juga FakeNet.
5. pengumpulan data hasil *reverse engineering*, di mana sampel dianalisis menggunakan Ghidra untuk mengidentifikasi fungsi penting, teknik *obfuscation*, dan logika serangan. Temuan utama dicatat dalam bentuk catatan analisis (*analysis notes*), *screenshot*, dan *export decompiled code* untuk bagian-bagian relevan. Seluruh data ini kemudian disatukan dalam basis data kecil atau *spreadsheet* untuk memudahkan perbandingan antara metode dan penyusunan IoC.

3.4. Teknik Analisis Data

Teknik analisis data dalam penelitian ini dibagi menjadi dua bagian besar, yaitu analisis kualitatif deskriptif dan analisis kuantitatif sederhana. Analisis kualitatif

deskriptif dilakukan dengan cara membaca, menafsirkan, dan membandingkan laporan hasil *static analysis*, *dynamic analysis*, dan *reverse engineering* untuk setiap sampel. Dari sini, peneliti mengidentifikasi pola serangan, teknik yang digunakan (misalnya *persistence*, *privilege escalation*, *C2 communication*), serta peran masing-masing metode dalam mengungkap detail tersebut.

Analisis kuantitatif dilakukan dengan menyusun matriks IoC yang memuat daftar indicator beserta informasi apakah indikator tersebut ditemukan oleh *static analysis*, *dynamic analysis*, *reverse engineering*, atau kombinasi di antaranya. Dari matriks ini dihitung beberapa metrik, antara lain yang pertama jumlah total IoC per metode, kedua jumlah IoC unik per metode,ketiga overlap IoC antara metode, dan keempat *persentase* cakupan metode terhadap total IoC yang diperoleh dari kombinasi ketiganya.

BAB IV
HASIL DAN PEMBAHASAN

4.1 Analisis Kebutuhan

Tools yang akan digunakan dalam penelitian ini perlu juga untuk diperhatikan, karena dapat sangat membantu dalam proses analisis. Analisis Malware menggunakan *virtual machine* (VM) adalah Teknik yang sangat umum digunakan untuk menghindari dampak kebocoran/ berbahaya dari malware pada *system* utama (Pc host) dan untuk memberikan lingkungan yang terisolasi untuk melakukan analisis. Berikut ini adalah beberapa yang dibutuhkan untuk dapat melakukan penelitian:

1. Kebutuhan Perangkat Lunak (Software)
    a. VPS/Server *Honeypot*: CPU 2 core, RAM 2 GB, Storage 50 GB, Bandwidth unlimited, fungsinya untuk Menjalankan honeypot Dionaea yang terkoneksi internet untuk menangkap malware aktif.
    b. Mesin Analisis (Host): CPU 4 core, RAM 8 GB, Storage 100 GB SSD, Menjalankan VM FLARE-VM (Windows 11) untuk seluruh proses analisis.
    c. Vmware sebagai solasi lingkungan analisis pada sampel malware
    d. Jaringan Lab: Memastikan *traffic* malware tidak keluar ke internet nyata
2. kebutuhan pengumpulan sampel (honeypot)
    a. Sistem honeypot harus dapat beroperasi secara online dan terhubung ke internet untuk menangkap malware dari penyerang nyata secara real-time, bukan dari dataset statis.
    b. *Honeypot* Dionaea harus mampu mengemulasi layanan rentan (*SMB, HTTP, FTP, MSSQL*) yang menjadi target serangan malware Windows, khususnya port 445 untuk menangkap varian worm seperti WannaCry.

c. Sistem harus dapat menyimpan file malware yang dikirim penyerang ke direktori tertentu dalam format asli (binary PE) tanpa modifikasi.
d. Sistem harus memiliki mekanisme deduplikasi hash (MD5/SHA-256) untuk menghindari analisis sampel yang identik, memastikan keberagaman dataset.
e. Sistem harus dapat beroperasi minimal satu bulan untuk mengumpulkan sampel.

3. Kebutuhan *Static analysis*
   a. Sistem harus dapat mengidentifikasi format file dan memverifikasi bahwa sampel adalah file PE (*Portable Executable*) Windows yang valid sebelum analisis dimulai.
   b. Sistem harus dapat menghasilkan nilai hash (MD5, SHA-1, SHA-256) setiap sampel untuk keperluan IoC dan verifikasi integritas.
   c. Sistem harus dapat mendeteksi keberadaan *packer, obfuscator*, atau protector pada setiap sampel menggunakan Detect-It-Easy (DIE) untuk identifikasi *compiler, linker, dan overlay.*
   d. Sistem harus dapat menganalisis struktur PE header *(sections, timestamp, compiler, flags, overlay*) dan *Import Address Table* (IAT) menggunakan PEStudio
   e. Sistem harus dapat mengekstrak *strings* yang tertanam dalam binary *(URL, IP, registry key, nama file, mutex)* menggunakan *tool Strings*, termasuk string yang dikonstruksi secara dinamis di stack menggunakan FLOSS.
   f. Sistem harus dapat memverifikasi deteksi *lintas-engine* menggunakan VirusTotal untuk konfirmasi keluarga malware dan threat label.

4. Kebutuhan *Dynamic analysis*
   a. Sistem harus memiliki lingkungan sandbox terisolasi yang aman untuk mengeksekusi malware tanpa resiko infeksi ke sistem

produksi atau jaringan luar. Lingkungan pengujian dibatasi pada Windows 11 sesuai batasan penelitian.

b. Sistem harus dapat mengeksekusi malware secara manual menggunakan rundll32.exe (untuk sampel DLL) di dalam lingkungan sandbox FLAREVM, dengan pemantauan *real-time* dari tools monitoring yang aktif bersamaan.
c. Sistem harus dapat merekam setiap operasi *file, registry*, proses, dan jaringan secara *real-time* menggunakan Process Monitor (ProcMon) dan Process Explorer.
d. Sistem harus dapat merekam dan menganalisis seluruh traffic jaringan menggunakan Wireshark.
e. Sistem harus dapat menyimulasikan layanan jaringan (DNS, HTTP, HTTPS) menggunakan FakeNet-NG agar malware yakin berhasil terhubung ke server eksternal.

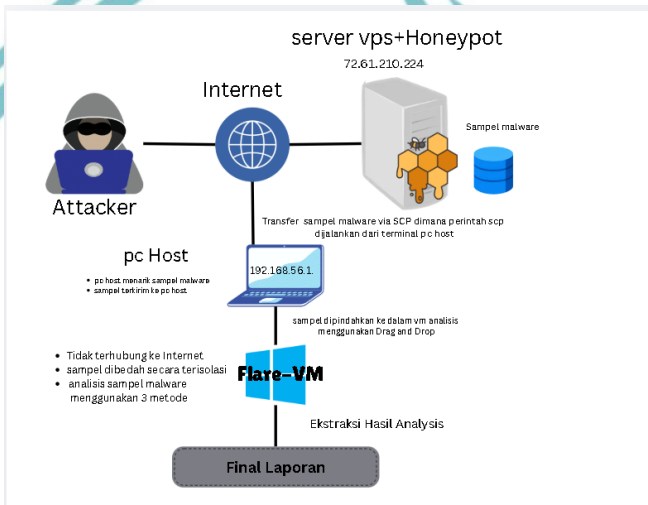
5. Kebutuhan *Reverse Engineering*
   a. Sistem harus dapat melakukan *disassembly* dan *decompilation binary* malware menggunakan Ghidra untuk menghasilkan pseudocode tingkat tinggi yang dapat dibaca analis.
   b. Sistem harus dapat mengidentifikasi fungsi-fungsi utama, *call chain* antar fungsi, dan *cross-reference* (XRef) dalam binary.
   c. Sistem harus dapat mengekstrak IoC lanjutan.

4.2 Perancangan Sistem

Perancangan sistem pada penelitian ini mencakup tiga komponen utama, yaitu (1) arsitektur sistem keseluruhan, menggambarkan topologi honeypot dan lab analisis. (2) alur kerja penelitian, menggambarkan tahapan dari pengambilan sampel hingga ekstraksi IoC, dan (3) perancangan matriks IoC, sebagai instrumen pengukuran komparatif tiga metode. Seluruh perancangan mengacu pada metodologi yang diterapkan di bab III.

4.2.1 Arsitektur Sistem Keseluruhan

Arsitektur sistem penelitian ini terdiri dari dua zona utama yang terpisah secara fisik dan logis. Zona honeypot (online, terhubung internet) dan zona lab analis (offline, terisolasi). Pemisahan ini adalah keharusan keamanan untuk memastikan malware yang sedang dianalisis tidak dapat menyebar ke luar lingkungan penelitian.



Gambar 4. 1 Topologi Sistem

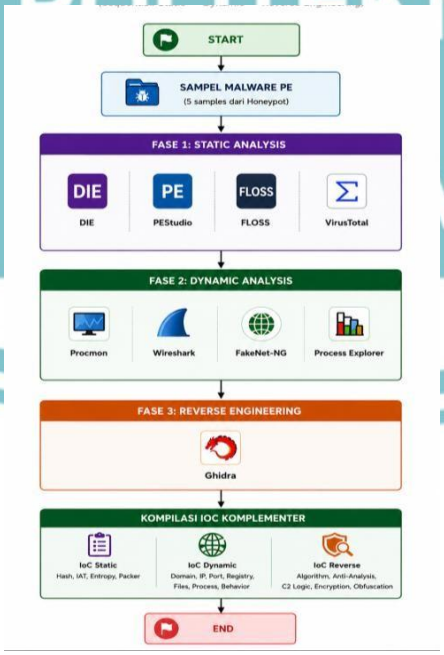
Penjelasan komponen arsitektur pada gambar 4.1:

1. Attacker(Penyerang): Representasi penyerang eksternal yang secara aktif melakukan scanning port 445 (SMB), port 22 (SSH), dan port layanan lain yang ditargetkan malware Windows. Penyerang tidak mengetahui bahwa mereka sedang menyerang honeypot.
2. Internet: Media transmisi publik yang bertindak sebagai jembatan lalu lintas data utama dunia luar. Jalur internet ini menghubungkan *Attacker* ke server, serta menghubungkan Laptop peneliti (*PC Host*) ke server.
3. Server (VPS Server + *Honeypot*): Server yang terhubung ke internet dengan IP publik, menjalankan honeypot Dionaea yang mengemulasi layanan rentan. Setiap file malware yang dikirim penyerang disimpan secara otomatis.

4. Pc Host: Laptop fisik utama milik peneliti yang berada di jaringan lokal (internal). Laptop ini berfungsi sebagai stasiun kendali pusat (*management station*) untuk melalukan *remote* ke alamat IP server vps dengan tujuan menarik sampel berkas berbahaya dari VPS jarak jauh menggunakan hak akses administrator.
5. Zona Lab Analisis (FLARE-VM): Mesin analisis dengan VM Windows 11 yang sepenuhnya terisolasi. Berisi semua tools analisis yang dibutuhkan (DIE, PEStudio, Strings/FLOSS, ProcMon, Process Explorer, Wireshark, FakeNet-NG, Ghidra). *Network* adapter VM dikonfigurasi sebagai *host-only.*
6. Final Laporan: Berisi hasil dari penelitian berupa rangkuman data dari kelima sampel malware menggunakan tiga metode *analysis.*

4.2.2 Perancangan Alur Kerja Penelitian

Alur kerja penelitian dirancang sebagai pipeline berurutan dengan enam fase mulai dari pengumpulan sampel di honeypot hingga penyusunan matriks IoC. Setiap *fase* menghasilkan output yang menjadi *input fase* berikutnya memastikan *traceable chain of evidence* yang diperlukan dalam penelitian forensic malware.



Gambar 4. 2 Flowchart Alur Kerja Penelitian Tri-Metode Analisis Malware

Metodologi penelitian pada Gambar 4.2 dimulai dengan pengumpulan sampel malware menggunakan honeypot Dionaea yang menangkap file PE secara *realtime*, dilanjutkan dengan seleksi dan verifikasi 5 sampel berdasarkan keunikan hash MD5/SHA-256. Kelima sampel tersebut lalu dianalisis secara statis menggunakan DIE, PEStudio, dan VirusTotal untuk menghasilkan IOC file, kemudian dianalisis secara dinamis di lingkungan FLARE-VM menggunakan Procmon dan Wireshark untuk mengamati perilakunya. Analisis dilanjutkan dengan *reverse engineering* menggunakan Ghidra untuk membongkar algoritma dan *call chain*. Tahap akhir adalah korelasi semua laporan menjadi matriks IOC yang dipetakan ke *framework MITRE ATT&CK* lengkap dengan rekomendasi mitigasi.

4.2.3 Perancangan Detail Alur *Static Analysis*

Alur *static analysis* dirancang mengikuti urutan dari pemeriksaan yang paling cepat dan tidak merusak (*hash verification*) hingga yang paling mendalam (*string extraction dan packer analysis*). Urutan ini memastikan analis memiliki gambaran dasar terlebih dahulu sebelum mendalami detail:

1. Verifikasi Hash: Menjalankan Get-FileHash pada PowerShell untuk menghasilkan MD5 dan SHA-256. Catat sebagai file-based IoC pertama.
2. Pemeriksaan VirusTotal: Upload atau cari hash di VirusTotal. Catat *persentase* deteksi, *threat label*, *family label*, tag analisis, dan *Contacted URLs*.
3. Identifikasi Packer/Compiler (DIE): Menjalankan Detect-It-Easy terhadap sampel. Catat compiler, linker, OS target, dan keberadaan overlay. DIE memberikan identifikasi yang lebih akurat dibandingkan tool sejenis.
4. Analisis PE Header (PEStudio): Memeriksa tipe file, arsitektur, *compiler/linker*, *timestamp*, *sections* (nama, ukuran, *entropy*, *permissions*), *export table*, dan *overlay*. *Flag section* dengan *entropy* > 7 sebagai tersangka *packing/enkripsi*.
5. Analisis Import (PEStudio): Menggunakan PEStudio untuk *overview* IAT dan indikator mencurigakan (*flags* merah).

6. Ekstraksi Strings: Menjalankan Strings (*Sysinternals*) dan FLOSS terhadap sampel. Grep hasil untuk: URL (*http/https/ftp*), IP address, *registry path* (*HKCU/HKLM*), nama file *(.exe/.dll*), dan path Windows *(%AppData%, %Temp%, %System%).*
7. Dokumentasi IoC Static: Mencatat semua temuan dalam template IoC. Tandai tipe (*File Hash, Network, Registry, File Path, String*).

4.2.4 Perancangan Detail Alur *Dynamic Analysis*

Alur dynamic analysis dirancang dengan urutan yang memastikan semua monitoring tools aktif sebelum malware dieksekusi. Urutan ini sangat kritis karena data yang hilang tidak dapat dipulihkan:

1. Konfigurasi Network: Memastikan network adapter VM dalam mode host-only. Verifikasi tidak ada koneksi internet nyata.
2. Aktifkan Fake-Net: Menjalankan Fake-Net sebelum eksekusi malware. Pastikan *DNS, HTTP, HTTPS* simulator aktif.
3. Aktifkan ProcMon: Menjalankan Process Monitor dengan filter PID malware, memulai capture.
4. Mulai Wireshark Capture: Memastikan Wireshark merekam pada interface yang tepat (*host-only adapter*).
5. Eksekusi Manual: Menjalankan sampel secara langsung. Untuk sampel berbentuk DLL, menggunakan perintah: *rundll32.exe sample.dll,Play Game.*
6. Observasi Real-time (3-5 menit): Memantau Process Explorer untuk proses baru, koneksi jaringan, dan penggunaan CPU/memory. Perhatikan proses yang berjalan sebagai SISTEM.
7. Stop *Capture* dan Simpan: Hentikan ProcMon (simpan .PML), hentikan Wireshark (simpan .PCAP), mencatat temuan FakeNet..
8. Analisis Hasil Monitoring: Membaca hasil dari ProcMon (*file/registry/network events*), Process Explorer (*process tree, TCP connections, env vars*), Wireshark *(traffic PCAP)*, dan FakeNet-NG (*DNS/HTTP requests* dari malware).

4.2.5 Perancangan Detail Alur *Reverse Engineering*

Alur *reverse engineering* dirancang untuk memanfaatkan hasil static analysis (fungsi API yang diimpor, *sections* yang mencurigakan) sebagai titik investigasi di ghidra, sehingga waktu analisis lebih efisien. Berikut penjelasan alur nya:

1. Import Binary ke Ghidra: Membuat project baru di Ghidra, import sampel sebagai PE 32-bit atau 64-bit. Verifikasi *base address* dan format sudah benar.
2. Auto-Analysis: Menjalankan *auto-analysis* Ghidra dengan semua *analyzer* yang diaktifkan. Lalu menunggu hingga selesai (beberapa menit untuk file besar).
3. Identifikasi *Entry Points* dan Export: melakukan periksaan Symbol Tree > Exports untuk melihat fungsi yang di-export. Untuk WannaCry, mencari Play Game. Identifikasi DllMain/entry point utama.
4. Navigasi Fungsi Kritis (dari hasil IAT): Berdasarkan temuan PEStudio Explorer, menggunakan *Symbol Tree* > *Functions* untuk menemukan dan *men-decompile* fungsi yang memanggil API tersebut.
5. Identifikasi: string yang dikonstruksi dinamis, kondisi *anti-debugging*, *call* ke Windows API berbahaya.
6. *Cross-Reference Analysis* (XRef): Menggunakan *XRef* (X) untuk melihat siapa yang memanggil fungsi kritis dan apa yang dipanggil fungsi tersebut. *Rekonstruksi call chain* lengkap.
7. Deteksi Anti-Analysis: Cari pola *IsDebuggerPresent*, *NtQueryInformationProcess*, *RDTSC (timing check)*, pemeriksaan *registry VM*.
8. Ekstraksi IoC Advanced: Mencatat *mutex name*, *path hardcoded*, *C2 domain/IP* (jika tidak terenkripsi), algoritma enkripsi (dari pola instruksi *XOR loop*, *S-box lookup = RC4*, *dll*.).

4.3 Implementasi Sistem

Implementasi sistem dilakukan di dalam virtual mesin menggunakan os windows 11. Dalam melengkapi implementasi pada analisis yang akan dilakukan, maka di dalam vm windows akan diinstall tools yang berhubungan dengan tiga metode analisis yang akan dilakukan.

4.3.1 Honeypot

Dalam konteks penelitian ini, honeypot berfungsi sebagai infrastruktur koleksi sampel malware dari *environment* produksi yang sesungguhnya, bukan hanya dari repositori basis data publik yang mungkin sudah usang atau telah dianalisis sebelumnya.

Honeypot diimplementasikan dengan mensimulasikan layanan-layanan yang rentan terhadap eksploitasi, sehingga terlihat sebagai target yang menarik bagi penyerang. Instalasi T-Pot dilakukan pada sistem operasi Ubuntu Server LTS dengan spesifikasi minimal *processor multi-core*, memori RAM 16 GB, dan kapasitas penyimpanan minimal 50 GB untuk mengakomodasi akumulasi sampel malware. Proses instalasi diawali dengan pembaruan repositori sistem menggunakan perintah apt-update dan apt-upgrade untuk memastikan seluruh paket berada pada versi terbaru.

Instalasi T-Pot dilakukan melalui skrip instalasi otomatis yang dikloning dari repositori resmi, kemudian dieksekusi untuk mengunduh dan mengonfigurasi seluruh layanan honeypot yang diperlukan. Fokus konfigurasi khusus diarahkan pada Dionaea. Dionaea adalah jenis *low-interaction honeypot* yang memiliki kemampuan untuk menjebak peretas dan menyimpan berkas berbahaya (*Binary Malware*) yang dikirimkan melalui berbagai jalur protocol jaringan.

Gambar 4. 3 Struktur direktori dan berkas konfigurasi Dionaea

Konfigurasi diatur secara terstruktur kedalam tiga bagian utama yang saling terintegrasi berdasarkan struktur folder pada gambar 4.3, yaitu:

1. Konfigurasi pusat sistem (Dionaea.cfg)

   Folder Dionaea.cfg merupakan otak utama yang mengatur dasar operasional Dionaea. Folder ini dikonfigurasi agar *honeypot* mengikat seluruh alamat IP yang tersedia agar bisa menerima serangan dari arah manapun serta memerintahkan sistem untuk otomatis membaca semua umpan dan perekam yang dimasukkan kedalam folder aktif (*enabled*). Isi dari konfigurasi utama tersebut adalah sebagai berikut:

```
# Lokasi Berkas: /usr/local/etc/dionaea/dionaea.cfg
logging:
  default:
    loggers:
      - type: file
        file: "/usr/local/var/log/dionaea/dionaea.log"
        level: "info"
      - type: file
        file: "/usr/local/var/log/dionaea/dionaea-errors.log"
        level: "warning"

network:
  listen:
    addresses:
      - "0.0.0.0"
    interfaces:
      - "eth0"

processors:
  workers: 2

modules:
  includes:
    - "/usr/local/etc/dionaea/ihandlers-enabled/*.yaml"
    - "/usr/local/etc/dionaea/services-enabled/*.yaml"
```

2. Konfigurasi layanan tiruan sebagai umpan (*services-enabled*)

Folder ini diisi dengan berkas-berkas konfigurassi layanan palsu agar menarik peretas untuk masuk. Sesuai dengan data pada folder penyimpanan sistem *(/usr/local/var/lib/Dionaea/)*, layanan yang dibuka sebagai pintu jebakan, yaitu:

a. HTTP (Port 80) Untuk menyamar sebagai server situs web berbasis apache ubuntu agar memancing attacker yang sedang mencari celah keamanan web.
b. FTP (Port 21) Untuk menyamar sebagai server pengiriman berkas untuk menjebak serangan pencurian kata sandi (*brute-force*)
c. TFTP (Port 69) Untuk meniru *protocol transfer file* cepat yang sering dimanfaatkan malware untuk mengirim *payload* tambahan.

Berikut ini adalah isi berkas konfigurasi untuk umpan layanan HTTP dan FTP yang diterapkan:

```
# Lokasi Berkas: /usr/local/etc/dionaea/services-available/http.yaml
define:
  - name: "http"
    config:
      port: 80
      root: "/usr/local/var/lib/dionaea/http"
      banner: "Apache/2.4.41 (Ubuntu)"

# Lokasi Berkas: /usr/local/etc/dionaea/services-available/ftp.yaml
define:
  - name: "ftp"
    config:
      port: 21
      root: "/usr/local/var/lib/dionaea/ftp"
      banner: "220 FTP version 1.0 ready"
```

3. Konfigurasi perekaman dan karantina serangan (*inhandlers-enabled*)

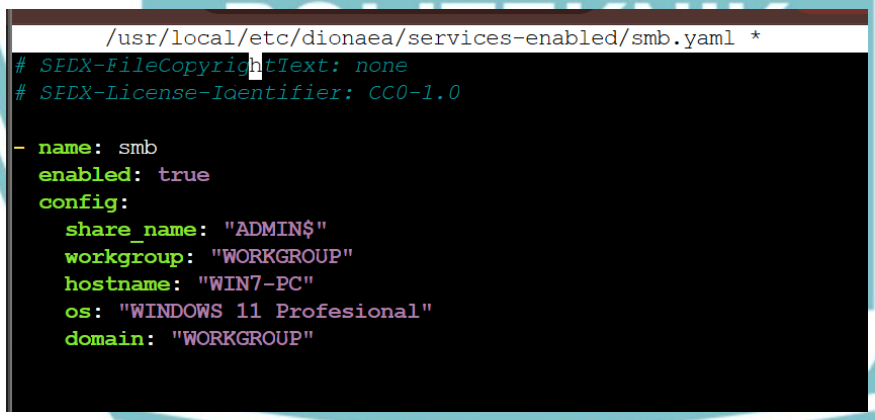
Seluruh aktivitas penyerang akan direkam secara senyap Ketika penyerang/peretas berintekrasi dengan honeypot. Modul penangan insiden (*incident handlers*) dikonfigurasi untuk mencatat data serangan ke dua tempat, yaitu database local SQLite (Dionaea.sqlite) untuk mencatat identitas IP penyerang dan waktu serangan, serta berkas log berformat

JSON (Dionaea.json) untuk kebutuhan analisis lanjutan. Berikut adalah isi konfigurasi perekaman database lokal yang dipasang:

```
# Lokasi Berkas: /usr/local/etc/dionaea/ihandlers-available/log_sqlite.yaml
define:
 - name: "log_sqlite"
   config:
    filename: "/usr/local/var/lib/dionaea/dionaea.sqlite".
```

Jika peretas mencoba mengirimkan atau mengunduh virus kedalam honeypot, Dionaea secara otomatis memotong perintah eksekusinya. File tersebut akan langsung disalin dan disimpan denga naman didalam folder */usr/local/var/lib/Dionaea/binaries/*, sedangkan rekaman aliran datanya disimpan pada folder *bistreams/*. Berkas-berkas di dalam folder inilah yang nantinya akan dijadikan sebagai sampel utama dalam proses analisis *forensic* keamanan siber.

Selain layanan berbasis web dan transfer dokumen, *protocol* lain yang menjadi target utama *eksploitasi* oleh malware adalah SMB (*Server Message Block*). Pada penelitian ini, berkas konfigurasi untuk umpan SMB yang aktif dapat dilihat pada gambar 4.4 berikut ini:



Gambar 4. 4 Berkas konfigurasi Layanan SMB

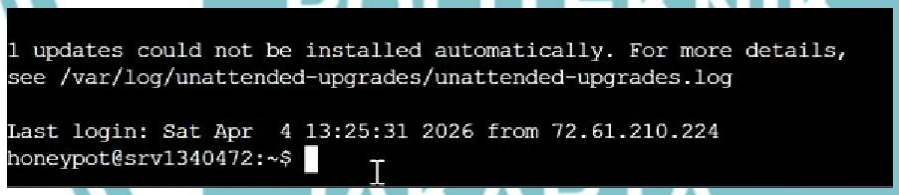
Melalui konfigurasi diatas, beberapa parameter penting sengaja dipasang sebagai umpan tiruan, yaitu:

a. share_name: "ADMIN$": Membuat folder bagikan (network share) palsu dengan nama *folder administrator* tersembunyi (ADMIN$). Folder jenis ini

biasanya menjadi incaran utama malware untuk menaruh berkas virus secara ilegal.

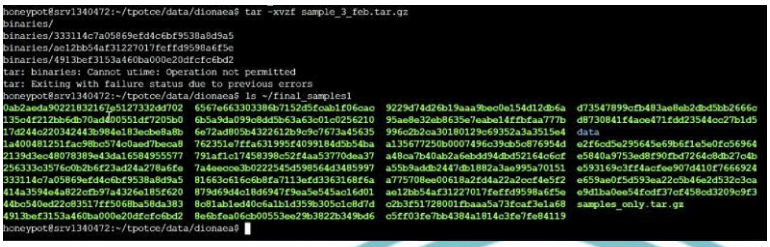
b. hostname: "WIN7-PC" dan os: "WINDOWS 11 Profesional", Dionaea memberikan informasi identitas palsu kepada pemindai luar bahwa mesin yang sedang berjalan ini adalah komputer Windows biasa. Ketidaksesuaian sengaja dibuat agar terlihat seperti komputer pengguna (user) yang tidak dikelola dengan baik oleh tim IT, sehingga peretas merasa yakin untuk menyerangnya.
c. workgroup dan domain: "WORKGROUP": Mengatur agar komputer tiruan ini tergabung dalam kelompok jaringan standar (Workgroup), yang lazim ditemukan pada jaringan komputer perumahan atau kantor skala kecil yang rentan.

Dengan aktifnya protokol SMB tiruan ini bersama layanan-layanan lainnya, Dionaea memiliki jangkauan jebakan yang luas. Hal tersebut memungkinkan untuk mengelabui/menjebak penyerang untuk mengirimkan paket *eksploitasi*, yang kemudia ditangkap dan diproses oleh sistem perekaman serta folder penyimpanan sampel malware.



Gambar 4. 5 Tampilan Antarmuka User Session setelah login ke *Honeypot* (Tpot)

Gambar 4.5 menunjukkan kondisi terminal pada sistem operasi ubuntu server setelah *user*/pengguna berhasil melewati proses *autentikasi* via ssh ke akun honeypot.



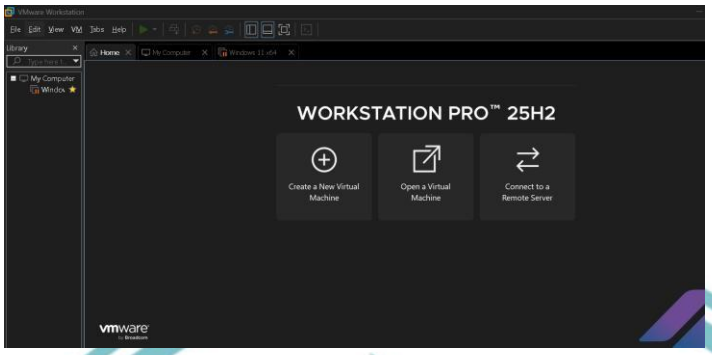
Gambar 4. 6 Arsip Sampel Malware yang berhasil dikumpulkan

Gambar 4.6 menampilkan beberapa file biner bawaan malware dan representasi nilai Hash dari masing masing malware yang berhasil ditangkap. *Honeypot* Dionaea sengaja mengubah nama asli berkas yang diunggah oleh penyerang/*attacker* menjadi nilai *cryptographic hashnya* dengan tujuan untuk memastikan bahwa setiap sampel *biner* yang disimpan adalah unik dan siap digunakan sebagai parameter pencarian (*query*) pada *database intelijen* ancaman global seperti VirusTotal untuk proses identifikasi jenis malware pada tahapan analisis selanjutnya.

4.3.2 Vmware

Platform ini dipilih sebagai fondasi infrastruktur laboratorium untuk menyediakan *isolated computing environments* yang dapat dikontrol secara *deterministic* sambil mempertahankan kapabilitas *advanced virtualization features.*

Instalasi VMware Workstation Pro versi 25H2 dilakukan pada host system dengan spesifikasi Windows 11 (build terbaru), processor dengan virtualization extensions enabled (Intel VT-x atau AMD-V), RAM 4 GB, dan *available storage* space 105 GB pada *primary disk drive*. Proses instalasi difasilitasi oleh *executable installer* yang diperoleh dari official VMware *distribution channels.*



Gambar 4. 7 Tampilan Awal Vmware

Pada gambar 4.7 ini adalah tampilan utama pada virtual mesin yang akan digunakan yaitu Vmware Workstation Pro versi 25H2. Vmware adalah sebuah perangkat lunak mesin virtual yang digunakan untuk menjalankan beberapa sistem operasi secara simultan dalam satu perangkat keras seperti laptop atau komputer.

4.3.3 Windows 11

Instalasi Windows 11 dilakukan pada virtual machine dengan *resource allocation:* 8 virtual CPU cores, 16 GB RAM, dan 50 GB virtual disk storage dengan dynamic expansion capability. VM boot dari Windows 11 x64 ISO image yang diperoleh dari official Microsoft distribution channels. Installation wizard mengarahkan user melalui disk partitioning, filesystem initialization (NTFS), dan user account provisioning.

Power management settings dikonfigurasi untuk prevent automatic sleep state dan display lock, ensuring continuous availability untuk monitoring dan analysis operations. Windows Defender diatur dalam *limited capacity mode* mengingat Flare-VM akan menginstal *security tools* yang mungkin *conflict* dengan default Windows *protection mechanisms.*

4.3.4 Flare-VM

Flare-VM merupakan specialized Windows distribution yang telah dikonfigurasi *comprehensively* untuk digital *forensic analysis*, malware *reverse engineering*, dan binary program *analysis*. Instalasi Flare-VM dieksekusi pada Windows 11

guest VM yang telah mencapai *operational readiness*. *Procedure* dimulai dengan *cloning* official Flare-VM *repository* dari Mandiant GitHub via *git version control system (git clone https://github.com/mandiant/flare-vm.git). Repository* berisi *comprehensive installation scripts, configuration manifests,* dan *pre-validated dependencies lists.*

Setelah itu, skrip instalasi dijalankan melalui PowerShell dengan hak akses *administrator*. Skrip ini secara otomatis menginstal berbagai perangkat lunak forensik yang dibutuhkan, mulai dari mengunduh aplikasi dari sumber resmi, memeriksa kompatibilitas antarperangkat lunak, mengatur konfigurasi sistem, membuat direktori kerja, hingga melakukan penyesuaian pada sistem operasi. Proses instalasi berlangsung cukup lama, yaitu sekitar 5–10 jam, tergantung pada kecepatan dan kestabilan koneksi internet yang digunakan. Selama proses berlangsung, sistem dapat melakukan restart secara otomatis beberapa kali untuk menerapkan perubahan pada sistem operasi serta mengaktifkan driver dan komponen tertentu yang memerlukan proses reboot sebelum dapat digunakan dengan baik.

Gambar 4. 8 Halaman Utama Pada OS Windows 11

Pada Gambar 4.8 adalah tampilan halaman utama pada Windows 11x64 yang telah dikonfigurasikan dengan arsitektur Flare-vm. Tampilan desktop didominasi oleh latar belakang (wallpaper) resmi Flare-Vm (platform distribusi alat forensic digital dari Mandiant/fireeye) yang menampilkan robot komputer sedang menganalisis artefak siber berbahaya.

4.4 Pengujian

Pengujian analisis sampel malware dilakukan menggunakan tiga metode yaitu *static*, *dynamic*, dan *reverse engineering* yang akan saling melengkapi untuk menghasilkan gambaran lengkap mengenai karakteristik dan perilaku dari sampel malware yang dianalisis.

4.4.1 *Honeypot*

Setelah proses pengumpulan selesai, selanjutnya melakukan proses seleksi untuk memilih sampel malware yang akan dianalisis lebih lanjut. Kriteria seleksi yang diterapkan meliputi validasi format yaitu memastikan file merupakan bentuk *executable* yang valid. Kedua, keunikan hash yaitu memiliki sampel dengan hash yang berbeda-beda untuk menghindari duplikasi analisis terhadap sampel. Ketiga, keberagaman tipe, yaitu memastikan sampel mewakili berbagai jenis malware agar hasil penelitian dapat memberikan gambaran yang komprehensif. Berdasarkan kriteria seleksi tersebut peneliti memilih 5 sampel malware untuk dianalisis menggunakan tiga metode analisis (*static*, *dynamic*, dan *reverse engineering*). Kelima sampel tersebut beserta dengan informasi hash-nya adalah sebagai berikut ini:

Tabel 4. 1 Data Sampel Malware

Sample Malware	MD5 Hash	SHA-256 Hash	Tipe Malware
Sampel 1	7a4eecce3b0222545 d598564d3485997	C4ec46a0ce486cc3b61c6dc1 623d88c4907e0974178904d 89cc5046c647faab0	Launcher
Sampel 2	44bc540ed22c8351	529dad9429b7d3a12dec140f	Launcher

	7ff5068ba58da383	c790d7fe842c6b8cf4cdc3330deec6b94e050609	
Sampel 3	95ae8e32eb8635e7eabe14ffbfaa777b	fb648bfb485f910e065cc18778364a56be32044d1ac4729449f3cc28221b12e8	WannaCry
Sampel 4	791af1c17458398c52f4aa53770dea37	60c9675c271480cb1bf369d56a3bd1f2d4fe1453d50717fe194a9f569b99be71	WannaCry
Sampel 5	996c2b2ca3018012 9c693515e4	df6d5b29a97647bca44e2306069f7675ef992f591c8c761af99bdc17cfa7692	WannaCry

Berdasarkan data yang ada pada tabel 4.1 adalah 5 (lima) sampel malware yang telah diseleksi sebagai objek penelitian. Kelima sampel malware tersebut yang akan digunakan untuk penelitian menggunakan tiga metode yaitu *static analysis*, *dynamic analysis*, dan *reverse engineering*.

4.4.2 Static Analysis

Analisis static dilakukan untuk mengidentifikasi hash kriptografis, pemeriksaan *PE header*, *Import Address Table (IAT)*, deteksi *packer*, *ekstraksi strings*, dan analisis kapabilitas berbasis MITRE ATT&CK tanpa mengeksekusi file malware

tersebut. Dalam penelitian ini analisis *static* dilakukan menggunakan tiga tools yang saling melengkapi:

1. PEStudio

PEStudio digunakan untuk memeriksa struktur PE header, *string* yang tertanam *(hardcoded strings*), *import table*, *entropy*, dan *section table* dari kelima sampel. Ringkasan temuan disajikan pada tabel 4.2

Tabel 4. 2 Ringkasan Hasil Analisis PEStudio

Aspek	Sampel 1	Sampel 2	Sampel 3	Sampel 4	Sampel 5
Format dan arsitektur	PE32 DLL, x86, GUI subsystem	PE32 DLL, x86, GUI subsystem	PE32 DLL, x86, GUI subsystem	PE32 DLL, x86, GUI subsystem	PE32 DLL, x86, GUI subsystem
Ukuran file		5.267.459 bytes	~5 MB	5.267.459 bytes	~5 MB
Entropy		4.097 (rendah–moderat)	4.326 (rendah–sedang)	6.418 (tinggi)	6.413 (tinggi)
Nama asli (export table)	launcher.dll	launcher.dll	launcher.dll	launcher.dll	
C2 Domain Hardcoded		iuqerfsodp9ifjaposdfjhgosurijfaewrwergwoff.com	Sama dengan Sampel 2		
Compiler /IDE		MS Linker 6.0, VC++ 6.0–8.0, VS 6.0	Identik dengan Sampel 2		
Import API		CreateProcessA, WriteFile			
overlay		Offset 0x00506000, 3 bytes, entropy 0.000	Identik Sampel 2		
Entry point				0x000111E9 (.text)	0x000011E9 (.text)
Indikasi utama	DLL "launcher" untuk menjalankan payload lain	Kemiripan kuat dengan WannaCry (C2 kill-switch), belum di-packing berat	di-packing beratCode reuse dari Sampel 2, satu kampanye/bui	Entropy naik tajam → mulai ada packing/obfuscation	Entropy tinggi konsisten Sampel 4, payload tersembunyi

			lder yang sama		di `.rsrc` (99,61%)

Point analisis dari tabel 4.2 diatas adalah:

a. Kelima sampel merupakan DLL 32-bit dengan nama asli "launcher.dll", mengindikasikan kampanye yang terkoordinasi atau dibuat dari *builder/framework* yang sama.
b. Sampel 2 dan 3 memiliki *C2 domain hardcoded* yang identik dan menyerupai kill-switch domain WannaCry, serta overlay identik pada offset dan ukuran yang sama yang mengindikasikan proses build otomatis dan standar.
c. Terjadi evolusi tingkat *obfuscation* yaitu pada sampel 1–3 berentropy rendah (mudah dianalisis langsung), sedangkan Sampel 4–5 berentropy tinggi (>6.4), menandakan *packing/encryption* yang lebih agresif. Pada Sampel 5, hampir seluruh *payload* (99,61%) tersimpan terenkripsi di section .rsrc, sehingga static analysis tidak dapat langsung mengakses logika malware sebenarnya.

2. DIE

Detect It Easy (DIE) merupakan tool analysis yang dirancang untuk *automated signature-based detection* terhadap *compiler artifacts, packing mechanisms,* dan *cryptographic signatures* dalam *binary executables.* Ringkasan hasil ditampilkan pada tabel 4.3 berikut ini:

Tabel 4. 3 Ringkasan Hasil Analisis DIE

Aspek	Sampel 1	Sampel 2	Sampel 3	Sampel 4	Sampel 5
Format	PE32, Little Endian	PE32, Little Endian	PE32, Little Endian	PE32, Little Endian	PE32, Little

					Endian
Ukuran	5,02 MiB	5,02 MiB	5,02 MiB	5,02 MiB	5,02 MiB
Arsitektur	i386, DLL	i386, DLL	i386, DLL	i386, DLL	i386, DLL
Os baseline		Windows (95)			
Linker	MS Linker (legacy)	MS Linker 6.00.8168	MS Linker 6.00.8168	MS Linker 6.00.8168	
Compiler	MSVC (legacy)	MSVC 12.00.8168 [C++]	MSVC 12.00.8168 [C++]	MSVC 12.00.8168 [C++]	
IDE	Visual Studio 6.0	Visual Studio 6.0	Visual Studio 6.0	Visual Studio 6.0	
Overlay	Offset 0x00506000, 3 bytes, Unknown		Identik Sampel 1		

Poin analisis pada tabel 4.3 diatas adalah:

a. Seluruh sampel konsisten sebagai PE32, *Little Endian*, arsitektur i386, dan berjenis DLL berukuran ±5,02 MiB.

b. *Compiler artifact* identik (Linker 6.00.8168, MSVC 12.00.8168, Visual Studio 6.0) pada seluruh sampel menjadi bukti kuat bahwa kelima sampel berasal dari satu *development environment* atau malware builder yang sama, yang dapat mendukung upaya *threat actor attribution.*

c. Penandaan "Windows (95)" pada Sampel 2 menunjukkan malware dirancang agar kompatibel dari Windows lama hingga versi terbaru, memperluas potensi target infeksi.

d. Overlay identik pada offset dan ukuran yang sama di beberapa sampel mengindikasikan *mekanisme embedding* yang sistematis (bukan kebetulan), kemungkinan sebagai *build marker* atau *checksum.*

3. FLOSS

FLOSS digunakan untuk mengekstrak *strings* yang tersembunyi melalui Teknik *obfuscation* seperti *stackstrings* dan *strings decoding*. Hasil ektraksi pada sampel 1 dirangkum pada tabel 4.4.

Tabel 4. 4 Ringkasan Hasil Ektraksi Strings FLOSS

Tahap	Temuan
Static strings	Ekstraksi string ASCII/UTF-16 yang tersimpan tanpa enkripsi dalam PE file
Stackstrings	Terdeteksi pada 4 fungsi — indikasi teknik obfuscation string secara dinamis di stack memory
Strings Decoded	Emulasi fungsi decoding pada alamat 0x100010ab
Waktu ekstraksi	111,95 detik

Poin analisis pada tabel 4.4 adalah:

a. Ditemukannya stackstring pada 4 fungsi menunjukkan malware secara sengaja menyembunyikan string sensitif (misalnya URL, path, atau perintah) agar tidak langsung terlihat pada *static string extraction* biasa.

b. Waktu eksekusi ±112 detik per sampel menunjukkan FLOSS tidak cocok untuk deteksi *real-time,* melainkan lebih sesuai digunakan untuk kebutuhan analisis forensik secara offline.

4.4.3 Dynamic Analysis

Analisis dinamis dilakukan dengan mengeksekusi sampel malware di dalam lingkungan sandbox FLARE-VM (Windows 11) yang terisolasi. Pemantauan dilakukan menggunakan Process Monitor (ProcMon), Process Explorer, Wireshark, dan FakeNet untuk mengamati perilaku nyata malware secara real-time.

1. Process Monitor (Procmon)

   Process Monitor merupakan tools pemantauan tingkat rendah yang mampu merekam seluruh operasi file sistem, *registry*, dan proses dalam waktu nyata. Berikut adalah hasil dari ringkasan temuan untuk kelima sampel yang disajikan pada tabel 4.5

Tabel 4. 5 Ringkasan Hasil Pemantauan Procmon

Sampel	Proses utama	Aktivitas kunci	Target file/registry	indikasi
1	`internet detect` (PID 1572)	Massive Write File operations	AppData\Local	Process masqueradi ng & persistence mechanism
2	ctfmon.exe	Registry enumeration ekstensif (RegOpenKey /RegQueryKe y sukses; RegQueryVal ue *NAME NOT FOUND*)	`HKLM\SOFTWARE\Microsoft\Input\Locales`, `HKCU\Software\Microsoft\Input\Settings`; `CreateFile` ke `System32\en-`	Exploratory registry query & upaya manipulasi system library yang tidak ditemukan pada sistem target

			US\mssp7e... (gagal)	
3	mssecsvr.exe (PID 7152)	Aggressive TCP connection attempts (TCP Reconnect/Disconnect berurutan, 06:38:05); Create/WriteFile	Multiple public IP; C:\Windows\tasksche.exe	Masquerading sebagai *Microsoft Security Service*; homographic filename mirip Task Scheduler; korelasi kuat dengan WannaCry
4	mssecsvr.exe, Wireshark.exe, Explorer.EXE (PID 5584)	ReadFile pada DLL (Qt5Core.dll, vfapi.dll); RegQueryKey CLSID enumeration (status SUCCESS/NAME NOT FOUND berulang)	HKCU\Software\Classes, HKCR\CLSID	Reconnaissance/environment discovery enumerasi komponen COM & shell extension, tanpa modifikasi langsung
5	rundll32.exe (PID 8804)	Loading DLL sistem (Kernel32.dll, KernelBase.dll, combase.dll, SHCore.dll); Reg operations	HKLM\System\CurrentControlSet\Control	Eksekusi malware via exported function DLL; pengumpulan konfigurasi sistem (driver, service, HAL)

		(RegOpenKey, RegQueryValue, RegCloseKey)		

Poin analisis dari tabel 4.5 diatas adalah:

a. Sampel 1, 3, dan 4/5 secara konsisten menampilkan pola masquerading (nama proses menyerupai komponen sistem legitimate) sebagai teknik utama untuk menghindari deteksi.
b. Sampel 3, 4, dan 5 menunjukkan keterkaitan langsung dengan mssecsvr.exe, komponen yang secara historis dikenal sebagai bagian dari ransomware WannaCry.
c. Pola reconnaissance (enumerasi registry/CLSID) tanpa perubahan sistem langsung terlihat pada Sampel 2, 4, dan 5, mengindikasikan tahap pengumpulan informasi sebelum eksekusi *payload* utama.

2. Process Explorer

Process Explorer digunakan untuk melihat informasi detail tentang proses yang sedang berjalan, termasuk struktur proses (*tree hierarchy*), modul yang dimuat, dan penggunaan memori. Berikut adalah ringkasan temuan dari kelima sampel disajikan pada tabel 4.6.

Tabel 4. 6 Ringkasan Hasil Pemantauan Process Explorer

Sampel	Temuan utama	Detail	indikasi
1	Thread anomali pada explorer.exe	Start Address 0x0000000000000000 (null pointer)	Signature process injection (kemungkinan via APC) evasion technique
2	powershell.exe dijalankan oleh explorer.exe (PID 5140)	Stack protection (stack cookie, DEP, ASLR) tidak aktif	Proses berjalan user-level namun tanpa mitigasi keamanan memori standar

3	mssecsvr.exe (PID 7152)	Koneksi ke banyak IP publik via dynamic source port (54820, 54823, 54824); status SYN_SENT	SMB scanning (port 445) & indikasi EternalBlue exploit ciri khas WannaCry
4	Process tree normal (csrss.exe, wininit.exe services.exe, svchost.exe)	SystemSettings.exe (PID 924) berstatus Suspended	Anomali minor; monitoring tercatat via explorer.exe - procexp.exe (64-bit, 26.696 KB)
5	mssecsvr.exe	Registry read pada HKLM\...\LanmanServer\Parameters (status SUCCESS)	Persiapan eksploitasi berbasis SMB konsisten dengan pola WannaCry

Poin analisis pada tabel 4.6 diatas adalah:

a. Process *injection* pada explorer.exe (Sampel 1) dan koneksi SMB agresif oleh mssecsvr.exe (Sampel 3 & 5) merupakan indikator teknis paling kuat bahwa sampel-sampel ini berperilaku sebagai worm/ransomware bergaya WannaCry/*EternalBlue.*
b. Tidak aktifnya *stack protection* pada Sampel 2 memperbesar permukaan serang (*attack surface*) jika terjadi eksploitasi memori lanjutan.

3. Wireshark

Wireshark digunakan untuk memantau dan merekam seluruh aktivitas jaringan yang dilakukan oleh malware, termasuk koneksi ke server *command-and-control.* Berikut adalah ringkasan temuan disajikan pada tabel 4.7 untuk kelima sampel.

Tabel 4. 7 Ringkasan Hasil Pemantauan Wireshark

Sampel	Protocol dominan	Aktivitas kunci	Indikasi
1	DNS (UDP/53)	Standard Query berulang ke wpad.localdomain, officeclient.microsoft.com, g.live.com, ecs.office.com; ditemukan retransmission	Resolusi domain umum Microsoft; retransmission menandakan respon DNS yang tidak stabil
2	ICMPv6, SMB/BROWSER	Multicast Listener Report (MLDv2) ke ff02::16; Host Announcement broadcast ke 192.168.81.255	Environment dual-stack IPv4/IPv6; broadcast identitas host di jaringan lokal

3	IGMPv3, ICMPv6, LLMNR, DHCP	Join/Leave multicast group; pertukaran DHCP Request–ACK (Transaction ID 0x6a21cb2d)	Aktivitas jaringan lokal standar (bukan indikasi C2 langsung)
4	ICMP, NetBIOS/SMB	Echo Request beruntun ke DNS publik (8.8.8.8, 8.8.4.4, 1.1.1.1, 1.0.0.1); Negotiate Protocol Request SMB oleh mssecsvr.exe	Uji konektivitas + upaya eksploitasi SMBv1 (srv.sys) untuk Remote Code Execution (EternalBlue)
5	ICMP	Echo Request masif ke 1.0.0.1 tanpa balasan (no response found!)	Kemungkinan kill-switch/C2 diblokir, packet loss, atau server tujuan melakukan silent drop

Poin analisis pada tabel 4.7 diatas adalah:

a. Sampel 4 memberikan bukti jaringan paling eksplisit terkait eksploitasi SMBv1 (*EternalBlue*), selaras dengan temuan Procmon dan Process Explorer pada mssecsvr.exe.
b. Sampel 5 menunjukkan pola percobaan komunikasi keluar yang gagal total, konsisten dengan mekanisme kill-switch WannaCry yang bergantung pada domain/IP tertentu untuk menghentikan eksekusi.

4. Fake Net

FakeNet merupakan tool jaringan simulasi yang dirancang khusus untuk analisis malware dinamis. Fakenet digunakan untuk membuat environment jaringan tiruan agar malware terhubung ke server palsu. Tools ini berperan sebagai server palsu yang merespons permintaan dari malware seolah-olah ia adalah server *commandand-control (C2)* yang asli.

4.4.4 Reverse Engineering

Reverse Engineering merupakan analisis tingkat lanjut yang dilakukan untuk memahami logika internal, algoritma, dan mekanisme kerja malware secara mendalam. Pada penelitian ini, *Reverse Engineering* dilakukan menggunakan Ghidra sebagai tools utama untuk proses *disassembly* dan *decompilation*. Ghidra merupakan tools *Reverse Engineering* gratis dari NSA yang mampu melakukan *disassembly* dan *decompilation* secara otomatis untuk berbagai

arsitektur processor. Berikut adalah ringkasan untuk kelima sampel disajikan pada tabel 4.8.

Tabel 4. 8 Ringkasan Hasil Analisis *Reverse Engineering*

Sampel	Fungsi/Simbol Kunci	Temuan Teknis	Indikasi
1	Import FindResource, LoadResource, LockResource, SizeofResource; CreateFile, CreateProcessA, CloseHandle	XRefs menunjukkan CALL langsung ke CreateFileA (0x1000107c) dan CreateProcessA (0x100010f3); string "mssecsvc..." di-push sebagai parameter pada 3 lokasi berbeda (0x10001077, 0x100010e2, 0x10001123)	Mekanisme Dropper/Packer — payload terenkripsi disimpan di .rsrc dan diekstrak ke memori saat eksekusi; string terkait manajemen service
2	FUN_100010ab	Pseudo-code mendeklarasikan STARTUPINFO dan PROCESS_INFORMATION; dipanggil secara UNCONDITIONAL CALL dari alamat 0x10001136	Fungsi ini secara pasti mengonfigurasi environment dan membuat proses baru setiap kali dieksekusi
3	Play Game (exported function)	XRefs menunjukkan 2 referensi bertipe Data (bukan Call); tercatat sebagai Entry (Export Address Table) dengan pointer ibo32 ke 0x10001114	Fungsi didaftarkan sebagai entry point ekspor DLL yang dapat diakses dari luar
4	Play Game (exported), FUN_10001016, FUN_100010ab	Pseudo-code PlayGame(void) memanggil sprintf untuk membangun path C:\%s\%s\ + WINDOWS + mssecsvr.exe, lalu memanggil FUN_10001016() dan FUN_100010ab() secara berurutan	File path construction dinamis menuju mssecsvr.exe di direktori Windows, diikuti eksekusi sub-rutin serangan lanjutan
5	entry (DllMain-like), 7 fungsi total	entry(param1, param2, param3) sesuai signature DllMain; fungsi terbesar FUN_1000113e (171 bytes) dipanggil langsung oleh entry (157 bytes); FUN_10001016 (149 bytes)	entry menjalankan logika kondisional utama melalui pemanggilan ke FUN_1000113e saat DLL dimuat

Poin analisis pada tabel 4.8 diatas adalah:

a. Sampel 1 menunjukkan mekanisme *dropper/packer klasik*: *payload* disembunyikan di *section .rsrc* dan diekstrak ke memori via

FindResource/LoadResource/LockResource, kemudian dieksekusi melalui *CreateProcessA. String "mssecsvc..."* yang di-push pada 3 titik kode berbeda konsisten dengan penamaan komponen WannaCry yang ditemukan pada tahap *dynamic analysis*.

b. Sampel 2 dan 4 memperlihatkan pola pemanggilan fungsi yang identik (FUN_10001016, FUN_100010ab) — mengindikasikan struktur kode yang direplikasi dari *template/builder* yang sama, konsisten dengan temuan *compiler artifact identik* pada tahap *static analysis.*

c. Sampel 4 secara *eksplisit* membuktikan secara logika kode bahwa malware membangun *path* menuju mssecsvr.exe di direktori Windows sebelum menjalankan *sub-rutin* serangan — mengonfirmasi temuan *dynamic analysis* mengenai proses mssecsvr.exe sebagai komponen inti WannaCry.

d. Sampel 3 dan 4 berbagi fungsi *ekspor Play Game* yang sama, menguatkan hipotesis bahwa kelima sampel berasal dari satu basis kode atau builder yang sama.

e. Sampel 5 menunjukkan struktur *entry* selayaknya DllMain standar Windows, dengan logika kondisional utama didelegasikan ke sub-rutin FUN_1000113e — pola umum pada malware bertipe DLL yang baru menjalankan logika inti setelah proses loading selesai.

4.5 Evaluasi Pengujian

Bagian ini menyajikan evaluasi dari *hasil static analysis*, *dynamic analysis*, dan *reverse engineering* pada sub-bab 4.4, dalam bentuk Indikator Kompromi / *Indicators of Compromise* (IoC) yang merupakan jejak digital yang mengindikasikan bahwa sistem telah terinfeksi atau dikompromikan oleh aktivitas malware tertentu. IoC ini disusun sebagai acuan deteksi dan *respons insiden siber*.

4.5.1 Indikator File (Hash, Nama, dan Struktur Biner)

Indicator berbasis file merupakan bentuk IoC paling dasar dan banyak digunakan dalam *threat intelligence* karena bersifat unik dan dapat langsung dicocokkan dengan database ancaman global. Nilai hash dari kelima sampel malware yang telah diverifikasi pada tahap honeypot dan *static analysis* dikompilasi Kembali agar dapat digunakan untuk pencarian maupun pencocokkan sampel serupa dilingkungan lain. Tabel 4.9 menyajikan nilai hash kriptografis lengkap dari kelima sampel malware yang dapat digunakan untuk pencarian dan identifikasi sampel yang serupa:

Tabel 4. 9 Tabel Indikator file

Sample Malware	MD5 Hash	SHA-256 Hash	SHA-1
Sampel 1	7a4eecce3b0222545d598564d3485997	C4ec46a0ce486cc3b61c6dc1623d88c4907e0974178904d89cc5046c647faab0	89fd53dc1e95ed18065de1ffb54f45e79b83c1f0
Sampel 2	44bc540ed22c83517ff5068ba58da383	529dad9429b7d3a12dec140fc790d7fe842c6b8cf4cdc3330deec6b94e050609	f8c0d90a190912a004d540dc069fa39903b28c46
Sampel 3	95ae8e32eb8635e7eabe14ffbfaa777b	fb648bfb485f910e065cc18778364a56be3204 4d1ac4729449f3cc28221b12e8	d5872c3f694a9e23c0583c4ae3e5c59eab26c021
Sampel 4	791af1c17458398c52f4aa53770dea37	60c9675c271480cb1bf369d56a3bd1f2d4fe1453d50717fe194a9f569b99be71	2b4d9b73092c311fc7dc5f94d6e23ee947d3399c
Sampel 5	996c2b2ca30180129c693515e4	df6d5b29a97647bca44e2306069f7675ef992f591c8c761af99bdc17cfa7692	6d788a5a77719ef3157c409108909da2456bf996

Karakteristik umum kelima sampel sebagai berikut:

1. Nama file asli (*export table*): launcher.dll identik pada seluruh sampel, mengindikasikan signature pembuat/builder yang konsisten.
2. Format: PE32, arsitektur i386 (32-bit), tipe DLL dengan subsistem GUI; dikompilasi menggunakan MS Visual C++ 6.0 dan MS Linker 6.0.
3. Ukuran fisik: konsisten ±5.267.459 bytes (~5 MB) pada mayoritas sampel mengindikasikan berasal dari template/builder yang sama.
4. Magic byte: 4D 5A (MZ) signature standar PE Windows yang valid.

4.5.2 Indikator Proses (Process Indicators)

Indicator proses menggambarkan entitas eksekusi yang secara langsung bertanggung jawab atas aktivitas malicious pada sistem korban. Identifikasi proses ini penting karena dapat langsung diimplementasikan sebagai aturan deteksi pada *Endpoint Detection and Response* (EDR) atau antivirus berbasis behavior tanpa memerlukan analisis mendalam lebih lanjut. Berdasarkan hasil *dynamic analysis* pada sub-bab 4.4.3 ditemukan enam proses yang menunjukkan perilaku mencurigakan mulai dari proses yang menyamar sebagai layanan sistem (*masquerading*) hingga proses dengan indikasi manipulasi timestamp dan injeksi memori. Ringkasan proses-proses tersebut beserta perilaku dan indikasinya disajikan pada tabel 4.10.

Tabel 4. 10 Proses Mencurigakan dan Perilakunya

Proses	Perilaku Kunci	Indikasi
mssecsvr.exe	Massive write ke AppData\Local; membuat file tiruan tasksche.exe; koneksi TCP agresif (Reconnect/Disconnect) ke puluhan IP publik	Persistence, masquerading, network propagation (worm)
internet_detect (PID 1572)	Nama menyerupai tool Windows legitimate; write ke AppData\Local	Evasion + persistence
ctfmon.exe	Registry scanning intensif pada HKLM/HKCU; akses file System32\en-US\mssp7e... (NAME NOT FOUND)	Reconnaissance/environment discovery

powershell.exe	Timestamp pembuatan 1972 (sebelum PowerShell dirilis/2006); stack protection nonaktif	Timestomping (antiforensik) + process masquerading
rundll32.exe	Digunakan untuk memicu eksekusi payload berbasis DLL	Mekanisme eksekusi standar untuk malware berbasis DLL
explorer.exe	Thread dengan Start Address 0x0000000000000000	Signature process/thread injection

4.5.3 Indikator Registry (*Registry Indicators*)

Windows *Registry* sering dimanfaatkan malware sebagai media penyimpanan konfigurasi maupun sebagai sumber informasi lingkungan sistem sebelum melancarkan aktivitas serangan lanjutan. oleh karena itu jejak akses *registry* yang tidak wajar (baik berupa pencarian konfigurasi yang tidak wajar maupun akses ke parameter layanan jaringan sensitive) dapat dijadikan *indicator* kompromi yang efektif untuk mendeteksi aktivitas *reconnaissance* maupun persiapan eksploitasi.

Hasil pemantauan registry pada sub-bab 4.4.3 mengidentidfikasi tiga resgistry key yang diakses secara mencurigakan oleh proses ctfmon.exe dan mssecsvr.exe dengan salah satu diantaranya menunjukkan keterkaitan langsung dengan konfigurasi SMB yang menjadi *vector* utama eksploitasi Wannacry. Rincian *registry key* tersebut disajikan pada tabel 4.11.

Tabel 4. 11 Registry Key yang diakses secara mencurigakan

Registry Key	Diakses oleh	Status	Indikasi
HKLM\SOFTWARE\Microsoft\Input\Locales	ctfmon.exe	SUCCESS	Pencarian konfigurasi lokalisasi sistem
HKCU\Software\Microsoft\Input\Settings	ctfmon.exe	NAME NOT FOUND (16/144 bytes)	Pencarian setting yang tidak eksis (reconnaissance)
HKLM\System\CurrentControlSet\Services\LanmanServer\Parameters (Null Session Pipes)	mssecsvr.exe	SUCCESS	Persiapan lateral movement/eksploitasi SMB (EternalBlue) khas WannaCry

4.5.4 Indikasi File Sistem (File System Indicators)

Selain *registry*, perubahan pada sistem berkas juga menjadi jejak penting yang dapat digunakan untuk mendeteksi keberadaan malware, terutama file yang dibuat di lokasi-lokasi strategis seperti direktori sistem *(C:\Windows)* atau direktori pengguna *(AppData\Local)* yang jarang dipantau oleh antivirus.

Berdasarkan temuan Procmon pada Sub-bab 4.4.3, teridentifikasi tiga lokasi berkas yang secara konsisten menjadi target operasi tulis oleh proses-proses *malicious*, termasuk lokasi *dropper* utama WannaCry. Ringkasan lokasi tersebut disajikan pada Tabel 4.12 berikut.

Tabel 4. 12 Lokasi File Mencurigakan

Lokasi	Proses terkait	Indikasi
C:\Windows\tasksche.exe	mssecsvr.exe	Dropper WannaCry; homograph masquerading; ditulis dengan elevated privileges
AppData\Local\	internet_detect, mssecsvr.exe	Payload staging multi-proses; lokasi jarang dimonitor antivirus
C:\Windows\System32\en-US\mssp7e...	ctfmon.exe (upaya akses, gagal)	Component/library probing untuk environment discovery

4.5.5 Indikator *Network* (*Network Indicators*)

Indikator jaringan memiliki nilai deteksi yang tinggi karena dapat diimplementasikan langsung pada perangkat keamanan seperti firewall, IDS/IPS, maupun DNS *filtering*, sehingga mampu mencegah penyebaran malware sebelum mencapai *endpoint.* Indikator ini mencakup pola koneksi mencurigakan, *domain command-and-control* yang *di-hardcode*, hingga protokol yang *dieksploitasi.*

Hasil pemantauan Wireshark dan Process Explorer pada Sub-bab 4.4.3 mengungkap beberapa pola jaringan yang signifikan, mulai dari aktivitas *scanning* masif menuju port SMB hingga *domain kill-switch* yang menjadi ciri khas ransomware WannaCry. Ringkasan indikator jaringan tersebut disajikan pada Tabel 4.13 berikut.

Tabel 4. 13 Aktivitas Jaringan Mencurigakan

Kategori	Detail	Indikasi
Network scanning	mssecsvr.exe, puluhan IP publik (mis. 122.164.21.32, 122.243.149.33), status dominan SYN_SENT, port acak lokal	SYN scanning / port scanning masif menuju port 445 (SMB)
TCP reconnect/disconnect	Berulang dalam hitungan detik	Karakteristik worm propagation
Domain hardcoded	iuqerfsodp9ifjaposdfjhgosurijfaewrwergwoff.com (Sampel 2 & 3)	Kill-switch C2 WannaCry, pola menyerupai DGA
DNS Query	Host 192.168.81.128: domain umum Microsoft (officeclient.microsoft.com, g.live.com, dll.), disertai retransmission	Kemungkinan callback ke C2
NetBIOS/SMB	Negotiate Protocol Request pada port 139/445	Persiapan eksploitasi SMBv1 (srv.sys) untuk RCE

4.5.6 Indikator API dan fungsi *Malicious*

Indikator berbasis API dan fungsi diperoleh dari hasil *reverse engineering* pada Sub-bab 4.4.4, dan merepresentasikan bukti paling kuat mengenai kapabilitas teknis malware karena diambil langsung dari logika kode program, bukan sekadar hasil observasi perilaku permukaan. Pola pemanggilan fungsi tertentu secara berurutan dapat menjadi *signature* yang efektif untuk deteksi berbasis perilaku (*behavior-based detection*).

Analisis terhadap fungsi-fungsi yang diimpor maupun diekspor oleh kelima sampel mengonfirmasi keberadaan pola *dropper/packer* klasik serta fungsi kustom yang bertanggung jawab atas eksekusi payload utama. Ringkasan fungsi-fungsi tersebut disajikan pada Tabel 4.14 berikut.

Tabel 4. 14 Fungsi API Mencurigakan

Library	Fungsi	Indikasi
KERNEL32.DLL	FindResourceA, LoadResource, LockResource, SizeofResource	Pola dropper/packer, ekstraksi payload dari .rsrc
MSVCRT.DLL	sprintf	Konstruksi path dinamis untuk payload (C:\%s\%s, mssecsvr.exe)

Export function	PlayGame (Sampel 3, 4, 5	Entry point payload; memanggil sub-rutin eksekusi lanjutan
DLL Entry Point	FUN_1000113e (171 bytes, terbesar)	Logika utama malware tersembunyi di entry point DLL

4.5.7 Indikator *Behavioral*

Berbeda dengan indikator teknis sebelumnya yang bersifat statis (hash, nama file, alamat IP), indikator perilaku menggambarkan pola aktivitas menyeluruh yang dibentuk dari kombinasi beberapa indikator sekaligus. Indikator jenis ini lebih tahan terhadap perubahan *superficial* pada malware (seperti pergantian hash atau domain), sehingga lebih sesuai digunakan untuk deteksi jangka panjang maupun varian baru dari keluarga malware yang sama.

Berdasarkan dari *static*, *dynamic*, dan *reverse engineering*, teridentifikasi tiga pola perilaku utama yang konsisten pada kelima sampel:

1. Dropper/Packer: entropi tinggi pada .rsrc (6,31–6,43) dengan rasio ukuran 99,5%+; ekstraksi via *FindResourceA-LoadResource-LockResource*, *dynamic library loading (LoadLibrary, GetProcAddress)* untuk menghindari jejak IAT statis.
2. Worm/Propagation: TCP scanning agresif menuju port 445, *persistence via tasksche.exe*, *lateral movement* melalui *query LanmanServer\NullSessionPipes*.
3. Evasion:
    a. *Timestomping — build time* dipalsukan menjadi 1 Jul 1972.
    b. *Process masquerading* — empat proses (*internet_detect, ctfmon.exe, mssecsvr.exe, powershell.exe*) menyamar sebagai proses sistem sah.
    c. *Header integrity* — struktur PE header tetap utuh tanpa tampering pasca-kompilasi, mengindikasikan proses build yang "normal" walau kontennya malicious.

4.5.8 *Table* Ringkasan IoC

Untuk memudahkan pemanfaatan langsung oleh praktisi kemanan, seluruh IoC yang telah dibahas pada sub-bab 4.5.1 hingga 4.5.7 dikompilasi ulang kedalam satu tabel ringkas yang dilengkapi dengan tingkat keparahan (*severity*) dan area aplikabilitasnya masing-masing. Tabel ini memudahkan proses *import IoC* ke berbagai *platform* keamanan seperti SIEM, IDS/IPS, maupun *threat intelligence platform*. Tabel 4.15 berikut menyajikan ringkasan IoC yang paling kritis beserta rekomendasi area penerapan nya.

Tabel 4. 15 Ringkasan IoC

Kategori	IoC Deskripsi	Severity	Aplikabilitas
File Hash	SHA-256: C4ec46a0..., 529dad94..., fb648bfb..., 60c9675c..., df6d5b29...	CRITICAL	Threat Intelligence Database
File Name	launcher.dll (nama ekspor konsisten)	HIGH	File System Search
File Path	C:\Windows\tasksche.exe	CRITICAL	File System Monitoring
Process Name	mssecsvr.exe, internet_detect	CRITICAL	Process Monitoring
Registry Key	HKLM\System\CurrentControlSet\Services\ LanmanServer\Parameters	HIGH	Registry Monitoring
Network Port	Port 445 (SMB), Port 139 (NetBIOS)	CRITICAL	Network Monitoring, Firewall Rules
Domain	iuqerfsodp9ifjaposdfjhgosurijfaewrwergwoff .com	HIGH	DNS Filtering, Threat Intelligence
IP Pattern	SYN_SENT status to multiple external IPs	CRITICAL	IDS/IPS Rules
API Pattern	CreateFileA + WriteFile + CreateProcessA + LoadResource	HIGH	Behaviorbased Detection
Memory IoC	Thread Start Address 0x00000000	HIGH	Memory Forensics

4.5.9 Rekomendasi Pemanfaatan IoC

IoC yang telah dikompilasi dapat digunakan untuk:

1) *Threat Intelligence Platform* (TIP): *Import hash* dan *domain indicator* ke *platform* seperti VirusTotal, AlienVault OTX, atau MISP
2) *Endpoint Detection and Response* (EDR): Konfigurasi *alert* pada *process creation (mssecsvr.exe)*, *file writes ke tasksche.exe*, dan *registry access patterns*
3) *Network Intrusion Detection System* (IDS/IPS): *Rule-set* untuk mendeteksi *SYN scanning patterns* dan SMB *exploitation attempts* pada port 445
4) *Security Information and Event Management* (SIEM): *Correlation rules* untuk *multi-stage detection*, mencakup *file creation*, *registry access*, dan *network connection events*.
5) *Forensic Investigation*: Hash *indicators* untuk *quick detection* pada *incident response* dan *post-compromise analysis*.
6) *Hunting Operation*: *Behavioral IoCs* untuk *proactive threat hunting* dalam jaringan organisasi.

BAB V
PENUTUP

5.1 Kesimpulan

Berdasarkan penelitian yang telah dilakukan terhadap lima sampel malware Windows yang diperoleh dari honeypot selama 2 bulan menggunakan tiga metode analisis secara berurutan (*static analysis*, *dynamic analysis*, dan *reverse engineering*), menghasilkan kesimpulan sebagai berikut:

1. Proses pengumpulan sampel dan temuan teknis ketiga metode analisis. *Honeypot* Dionaea yang dijalankan dalam platform T-Pot dan dikonfigurasi pada VPS berhasil menangkap sampel malware secara *real-time* dari penyerang nyata yang menargetkan port SMB (445), HTTP (80), dan FTP (21). Sampel yang tertangkap diverifikasi menggunakan VirusTotal, kemudian diseleksi secara *purposive* (keunikan hash, validitas format PE, keberagaman tipe) sehingga diperoleh 5 sampel *representative* yaitu 2 sampel bertipe Launcher (Sampel 1–2) dan 3 varian WannaCry (Sampel 3–5). Ketiga metode analisis yang diterapkan secara berurutan berhasil mengungkap karakteristik malware secara bertahap dan saling melengkapi. *Static analysis* (PEStudio, DIE, FLOSS) mengidentifikasi bahwa kelima sampel dikompilasi dengan *toolchain* identik dan bernama asli "launcher.dll" mengindikasikan satu *builder framework* yang sama dengan entropi rendah pada Sampel 1–3 (4,0–4,3) dan tinggi pada Sampel 4–5 (6,4+, *payload* terenkripsi pada .rsrc), serta *domain C2 hardcoded* yang menyerupai *kill-switch* WannaCry pada Sampel 2 dan 3. *Dynamic analysis* (Procmon, Process Explorer, Wireshark, FakeNet) mengungkap perilaku *runtime* yang tidak terlihat pada static analysis, di antaranya *masquerading* dan *persistence* pada Sampel 1, *registry enumeration* pada Sampel 2, perilaku *worm* paling agresif dengan upaya eksploitasi SMBv1/*EternalBlue* oleh mssecsvr.exe pada Sampel 3 dan 4, serta persiapan eksploitasi SMB melalui *LanmanServer\Parameters* pada Sampel 5. *Reverse engineering* (Ghidra) mengonfirmasi logika internal di baliknya yaitu fungsi ekspor Play Game pada Sampel 3–5 sebagai *entry point payload*, *mekanisme two-stage*

execution pada Sampel 4 (konstruksi path mssecsvr.exe via sprintf diikuti eksekusi), pola *dropper/packer FindResource-LoadResource-LockResource-SizeofResource* pada Sampel 1, serta *conditional dispatcher logic* pada fungsi entry Sampel 5 yang memanggil sub-rutin FUN_1000113e sebagai logika inti malware.

2. Komplementaritas ketiga metode analisis. *Static*, *dynamic*, dan *reverse engineering* terbukti bersifat saling melengkapi dan tidak dapat saling menggantikan dalam menghasilkan *Indicator of Compromise* (IoC) yang komprehensif. *Static analysis* unggul dalam kecepatan dan keamanan eksekusi untuk *profiling* awal (IoC berbasis *file/hash*), *dynamic analysis* mengungkap *tradecraft* nyata (IoC jaringan, *registry*, dan proses), sedangkan *reverse engineering* menghasilkan IoC tingkat lanjut (mekanisme enkripsi, anti-debugging, dan logika C2) yang tidak dapat diperoleh dari kedua metode lainnya. Kombinasi ketiganya menghasilkan cakupan IoC yang jauh lebih lengkap dibandingkan penggunaan satu atau dua metode saja, mengonfirmasi bahwa pendekatan *mono-* atau *dual-*metode tidak memadai untuk menganalisis malware canggih seperti WannaCry yang mengimplementasikan *packing*, *anti-debugging*, C2 *terenkripsi*, dan *process injection* secara bersamaan.

5.2 Saran

Berdasarkan hasil penelitian dan kesimpulan yang telah dipaparkan, berikut adalah saran yang dapat dijadikan bahan pertimbangan untuk penelitian dan pengembangan selanjutnya:

1. Perluasan sampel dan periode pengumpulan, memperpanjang operasi honeypot menjadi minimal 3 bulan dan menambah sampel menjadi 20–50 untuk hasil *statistik* yang lebih representatif.
2. Integrasi *threat intelligence*, mengintegrasikan IoC (*hash*, *domain C2*, *registry key*, *mutex*) ke platform seperti MISP atau OpenCTI untuk mempercepat respons terhadap varian malware serupa.
3. Pengembangan YARA rules, menyusun dan mengevaluasi YARA rules berbasis IoC hasil penelitian terhadap sampel malware baru dari honeypot.

4. Otomatisasi *pipeline* analisis, mengembangkan scripting (Python/PowerShell) untuk menjalankan tools secara berurutan dan menyusun matriks IoC otomatis, guna meningkatkan skalabilitas dan replikabilitas.
5. Perluasan ke *platform* non-Windows, mengadaptasi metodologi tiga metode ini untuk malware Linux (server/IoT) dan Android (APK), mengingat meningkatnya ancaman pada *platform* tersebut.

DAFTAR PUSTAKA

AV-TEST Institute. (2025). *Malware statistics*. https://www.av-test.org/en/statistics/malware/

Bombardieri, C., & Capretti, S. (2015). *Honeypot-powered malware reverse engineering (HERESy)* [Preprint]. arXiv. https://arxiv.org/abs/1510.03892

Damaševičius, R., Maskeliūnas, R., Blažauskas, T., & Smuikys, A. (2025). FETA: A systematic and efficient approach for feature extraction in malware analysis. *Journal of Information Security and Applications, 90*, 103965. https://doi.org/10.1016/j.jisa.2025.103965

Eliando, E., Lumenta, A. S. M., & Najoan, X. B. N. (2022). LockBit 2.0 ransomware: Analysis of infection, persistence, and impact. *Cogito Smart Journal, 8*(2), 356–370. https://cogito.unklab.ac.id/index.php/cogito/article/view/356

Guo, M. I., Anwer, I., Riasat, A., & Kim, S. (2023). Windows malware detection based on static analysis with multiple features. *PeerJ Computer Science, 9*, e1319. https://doi.org/10.7717/peerj-cs.1319

Holbel, T., Combs, J., Graves, J., & Haynes, T. (2024). Utilizing virtualized honeypots for threat hunting, malware analysis and reporting. *Issues in Information Systems, 25*(1), 265–278. https://iacis.org/iis/2024/1_iis_2024_265-278.pdf

Infosec Institute. (2024). *Static malware analysis tutorial*. https://resources.infosecinstitute.com/topic/static-malware-analysis/

International Journal for Multidisciplinary Research. (2023). Malware analysis and reverse engineering. *IJFMR, 5*(6). https://www.ijfmr.com/papers/2023/6/10296.pdf

Kaspersky. (2024). *Ransomware: The biggest cyber threat of 2024*. Kaspersky Security Blog. https://www.kaspersky.com/resourcecenter/threats/ransomware

Muhammad, R. M., Stiawan, D., & Idris, M. Y. (2021). Integrated security system implementation for network environment using honeypot. *Journal of Navy and University Studies (JONUNS), 1*(1). https://jonuns.com/index.php/journal/article/view/619

StatCounter GlobalStats. (2024). *Desktop operating system market share worldwide*. https://gs.statcounter.com/os-market-share/desktop/worldwide/

Surfshark. (2025). *Malware cases by operating system*. https://surfshark.com/research/chart/malware-cases-windows-macOS/

YARA Project. (2024). *YARA official documentation*. ReadTheDocs. https://yara.readthedocs.io/

Yusirwan, S., Wahyudi, A., & Sanjaya, R. (2020). Implementation of malware analysis using static and dynamic analysis method. *Proceedings of the 2020 International Conference on Electrical Engineering and Informatics (ICEEI)*. IEEE. https://doi.org/10.1109/ICEEI51358.2020.9254935

National Institute of Standards and Technology. (2021). *Malware incident prevention and handling for desktops and laptops* (NIST Special Publication 800-83 Rev. 1). U.S. Department of Commerce. https://doi.org/10.6028/NIST.SP.800-83r1

Afianian, A., Niksefat, S., Sadeghiyan, B., & Baptiste, D. (2022). Malware dynamic analysis evasion techniques: A survey. *ACM Computing Surveys, 54*(6), 1–35. https://doi.org/10.1145/3365001

Grigorios, L., Apostolidis, K. D., & Kanas, V. G. (2021). Comparative review of malware analysis methodologies. *International Journal of Network Security & Its Applications, 13*(6), 91–104. https://doi.org/10.5121/ijnsa.2021.13608

Fortinet. (2024). *What are indicators of compromise (IoC)?* FortiGuard Glossary. https://www.fortinet.com/resources/cyberglossary/indicators-ofcompromise

Microsoft Security. (2024). *Indicators of compromise (IoCs) in Microsoft Defender*. Microsoft Learn. https://learn.microsoft.com/en-us/defenderendpoint/manage-indicators

Mandiant/Google. (2023). *FLARE FLOSS: FireEye Labs Obfuscated String Solver* (Version 3.0) [Software]. GitHub. https://github.com/mandiant/flare-floss

Microsoft Sysinternals. (2023). *Process Monitor (Procmon) for Windows* [Software]. Microsoft. https://learn.microsoft.com/en-us/sysinternals/downloads/procmon

VirusTotal. (2024). *VirusTotal: Analyze suspicious files, domains, IPs and URLs* [Web service]. Google. https://www.virustotal.com/

Wireshark Foundation. (2024). *Wireshark: Network protocol analyzer* (Version 4.2) [Software]. https://www.wireshark.org/

Avanzato, J. (2025). From CPU spikes to defense: How Varonis prevented a ransomware disaster. Varonis Blog. Retrieved from https://www.varonis.com/blog/varonis-prevents-ransomware-disaster

varo, S., Vila, G., & Pegueroles, J. (2026). Automating the detection of evasive Windows malware: An evaluated YARA rule library for anti-VM and antisandbox techniques. Journal of Cybersecurity and Privacy, 6(2), 69. https://doi.org/10.3390/jcp6020069

Kanj, S., Vila, G., & Pegueroles, J. (2026). Automating the detection of evasive Windows malware: An evaluated YARA rule library for anti-VM and antisandbox techniques. Journal of Cybersecurity and Privacy, 6(2), 69. https://doi.org/10.3390/jcp6020069

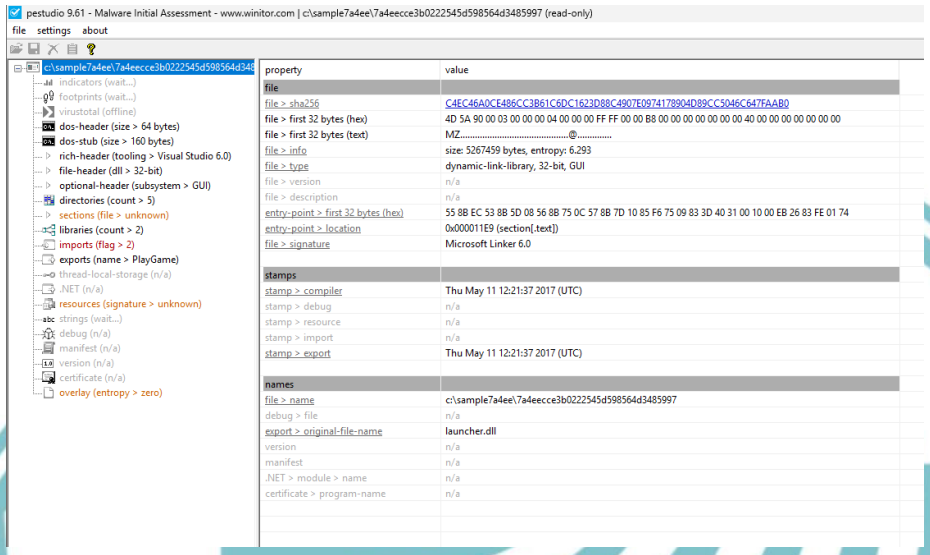
Varonis Systems. (2025). From CPU spikes to defense: How Varonis prevented a ransomware disaster. Retrieved from https://www.varonis.com/blog/varonisprevents-ransomware-disaster

Cyber Security News. (2025, February 6). *Ghidra 11.3 released*. Diakses dari https://cybersecuritynews.com/ghidra-11-3-released/

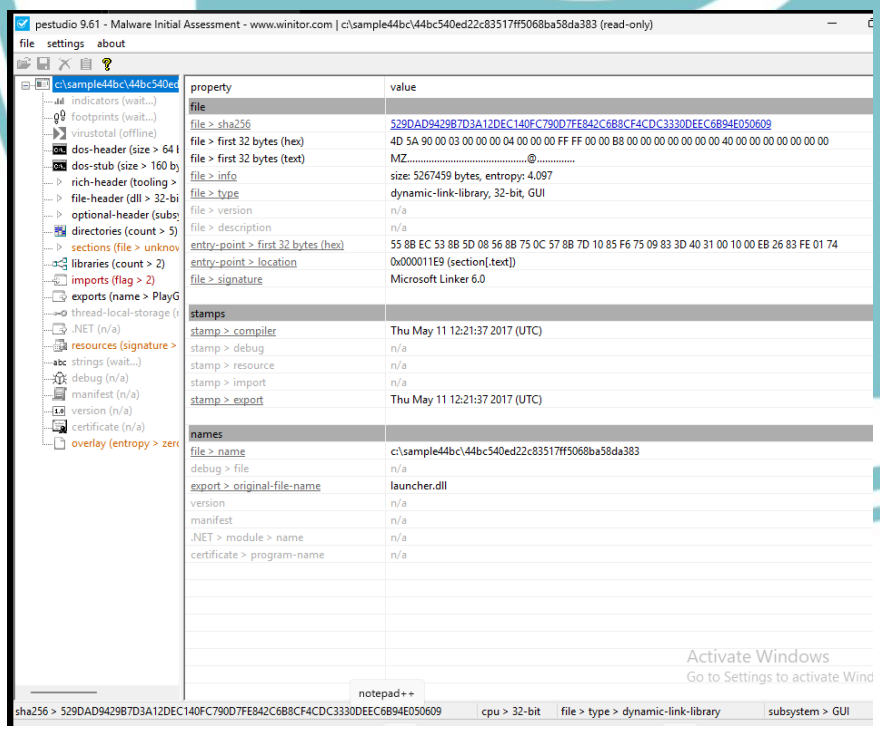
DAFTAR RIWAYAT HIDUP

Dina Yuliasti, akrab disapa Dina. Lahir di Bekasi, pada tanggal 23 Juli 2003. Penulis memulai Pendidikan formal di SD Negeri 062 Mompang Jae pada tahun 2009, setelah itu melanjutkan Pendidikan di SMP Negeri 1 Panyabungan Utara pada tahun 2015, Kemudian meneruskan Pendidikan di SMK Negeri 3 Panyabungan pada tahun 2019. Pada tahun 2022 penulis menempuh Pendidikan Diploma IV di Politeknik Negeri Jakarta dengan jurusan Teknik Informatika dan Komputer pada Program Studi Teknik Multimedia dan Jaringan.

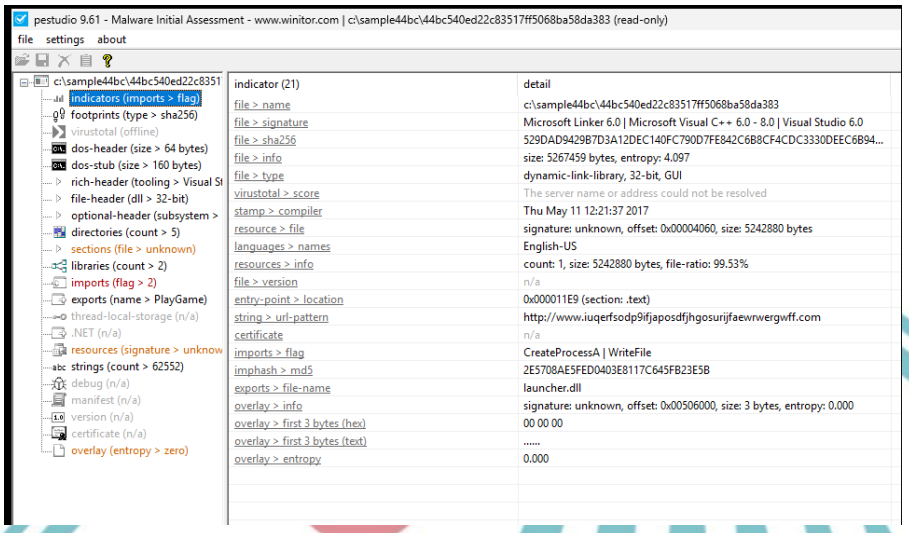
LAMPIRAN



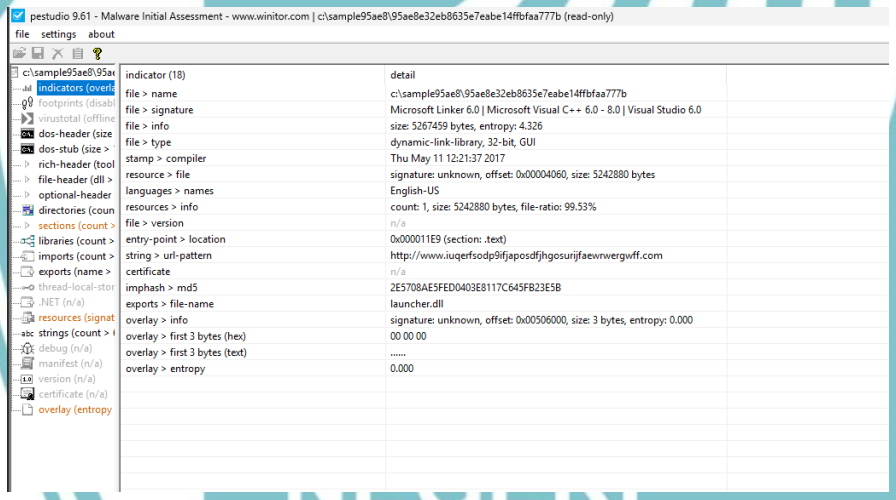
Lampiran 1 Tampilan PEStudio pada tahap analisis sampel 1



Lampiran 2 Tampilan PEStudio pada tahap analisis sampel 2

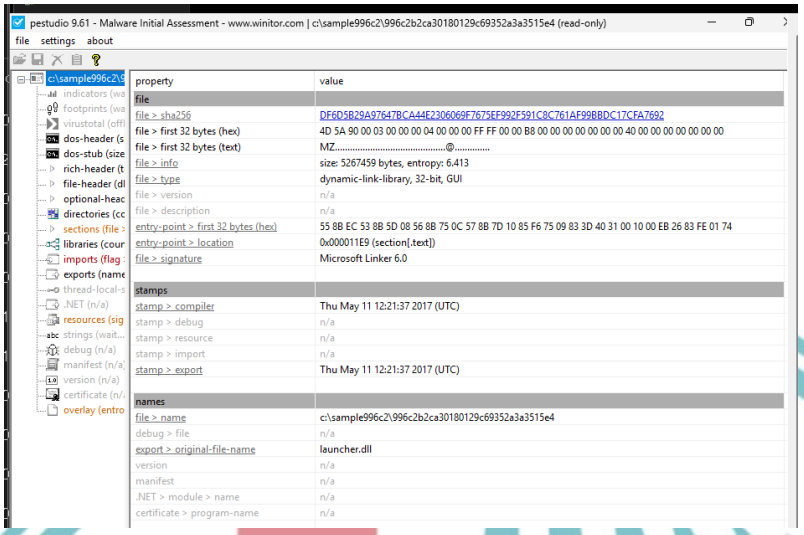


Lampiran 3 . Tampilan jendela indicators pada sampel 2

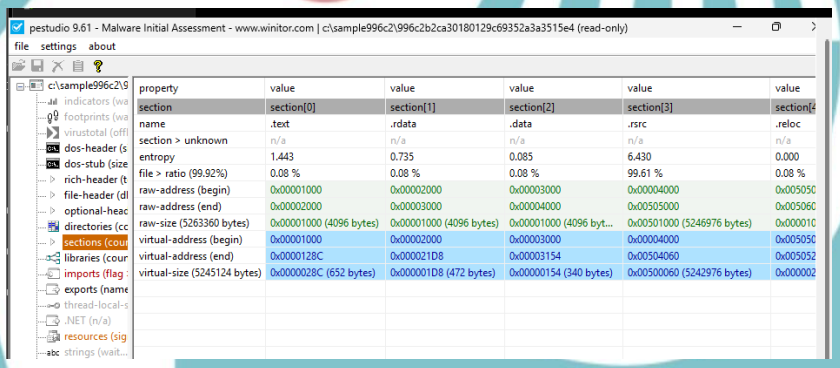


Lampiran 4 Tampilan menu indicators pada sampel 3

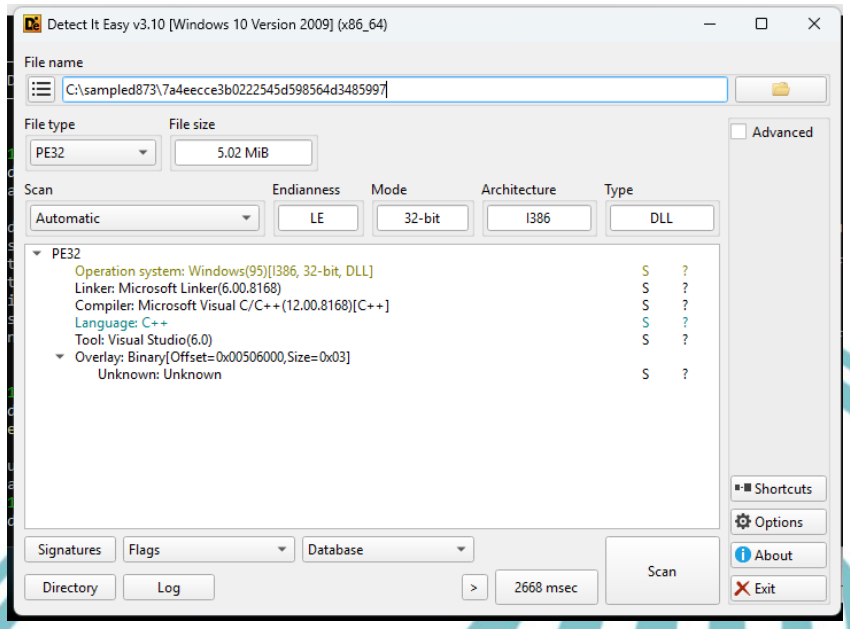
Lampiran 5 Tampilan jendela PEStudio pada sampel 4



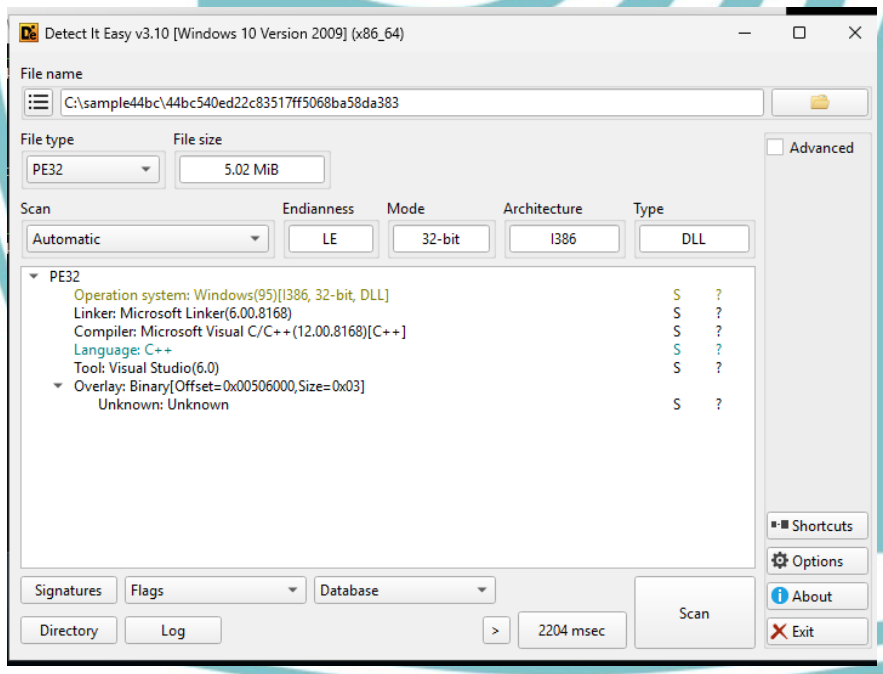
Lampiran 6 Tampilan jendela PEStudio pada sampel 5



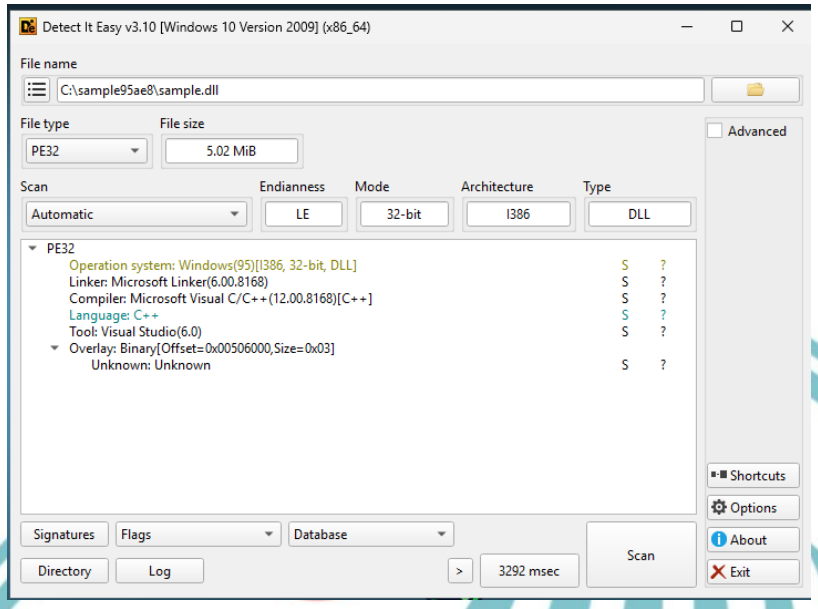
Lampiran 7 Tampilan menu section pada sampel 5



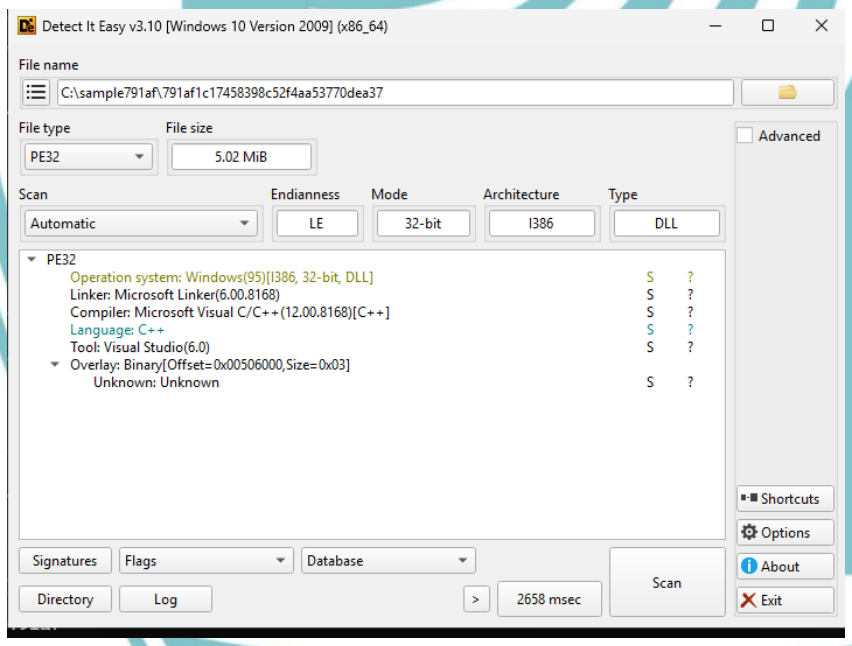
Lampiran 8 Tampilan DIE pada analisis sampel 1



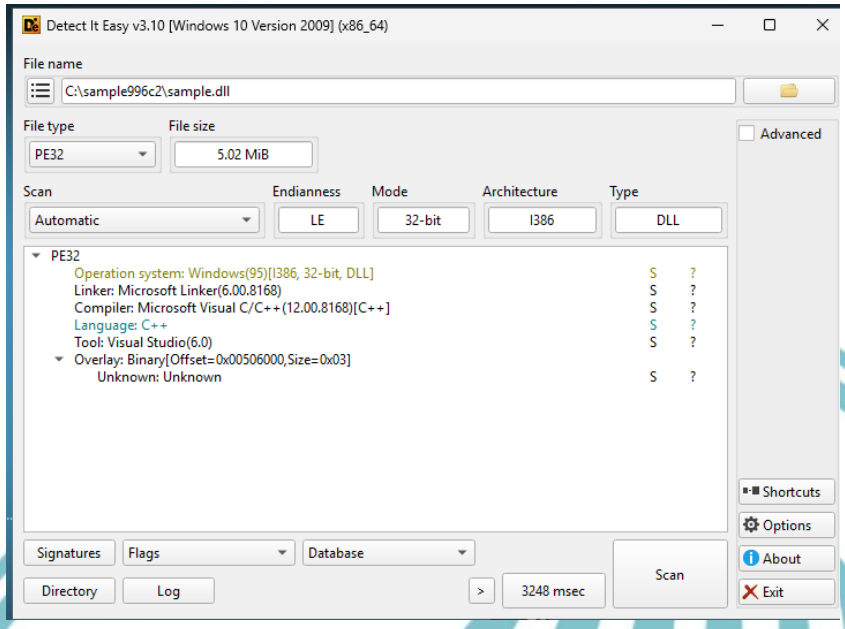
Lampiran 9 Tampilan DIE pada analisis sampel 2



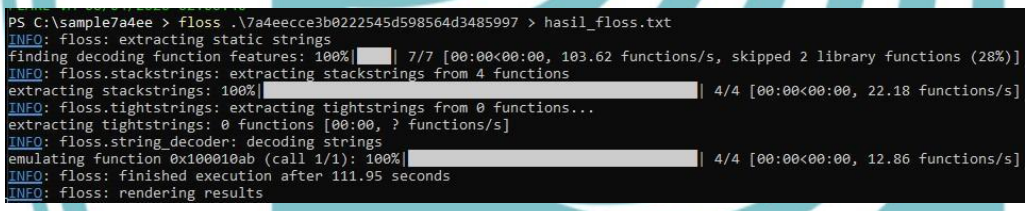
Lampiran 10 Tampilan DIE pada analisis sampel 3



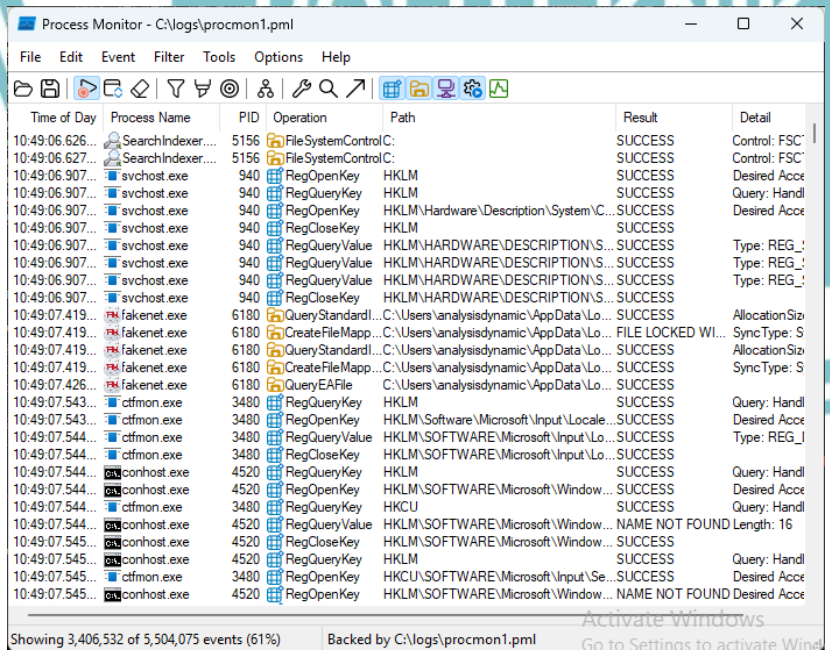
Lampiran 11 Tampilan DIE pada analisis sampel 4



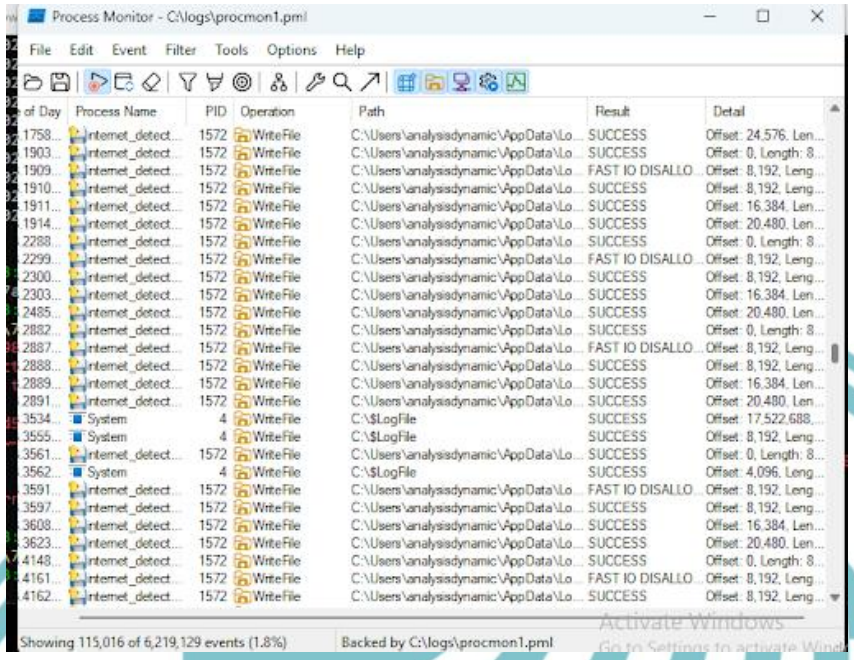
Lampiran 12 Tampilan DIE pada analisis sampel 4



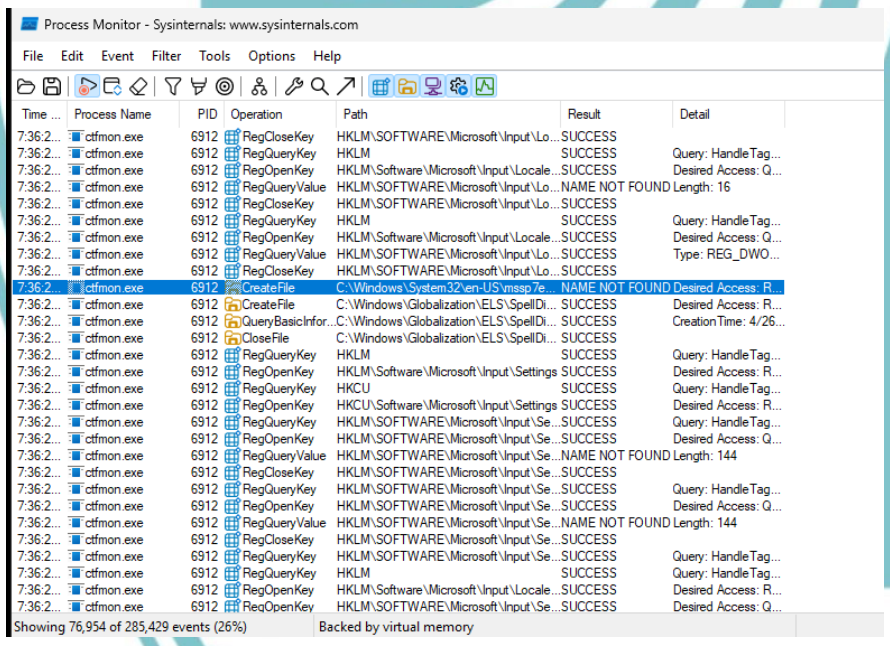
Lampiran 13 Proses ekstraksi dan deskripsi string pada sampel 1



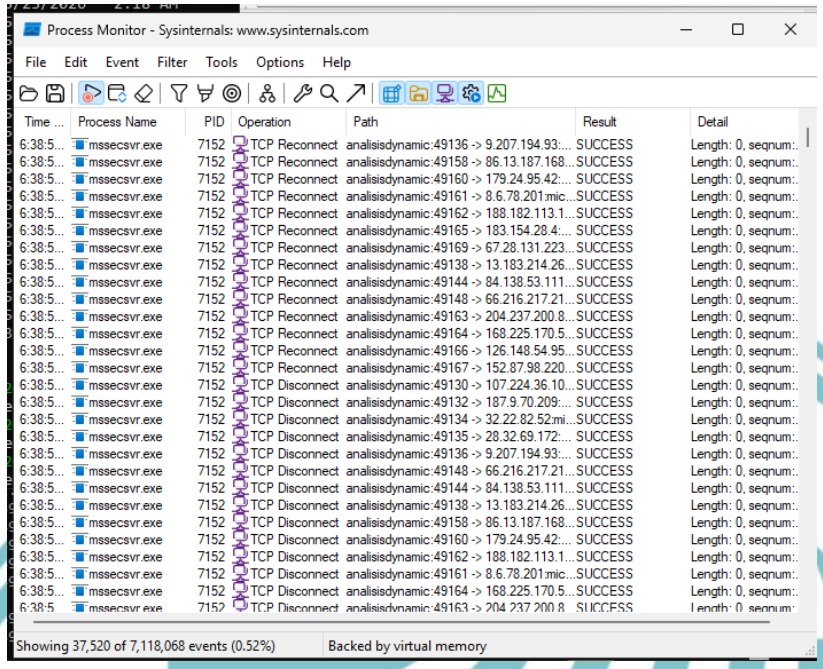
Lampiran 14 Pemantauan aktivitas terhadap sampel 1



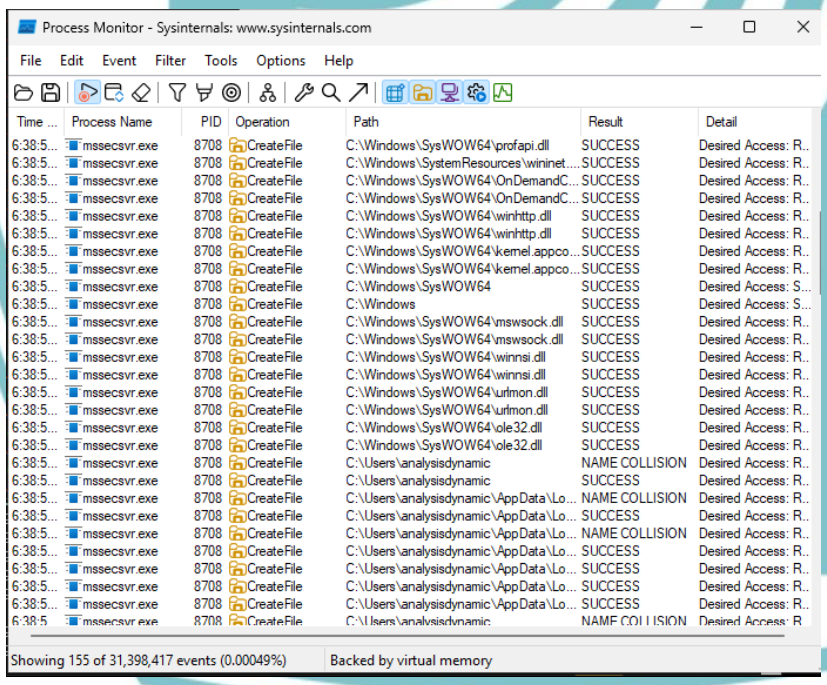
Lampiran 15 Perilaku anomali proses pada sampel 1



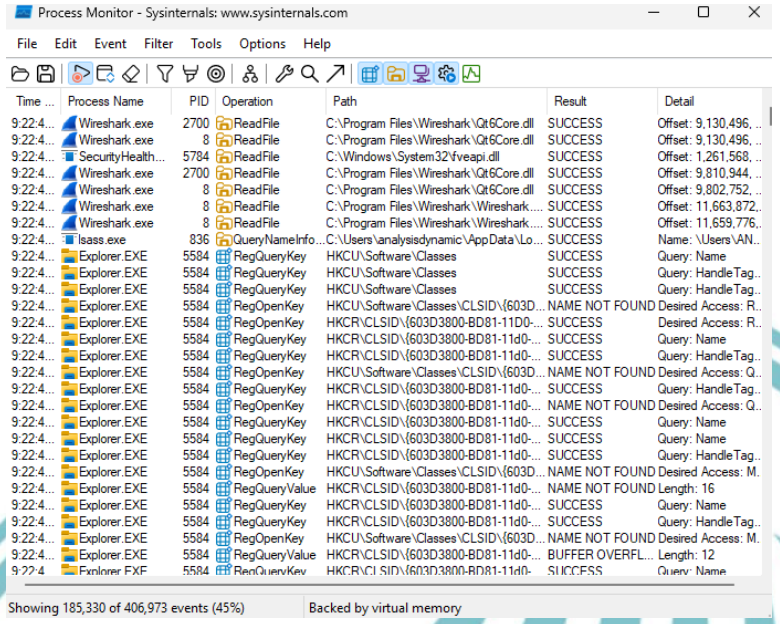
Lampiran 16 Tampilan log aktivitas Procmon pada sampel 2



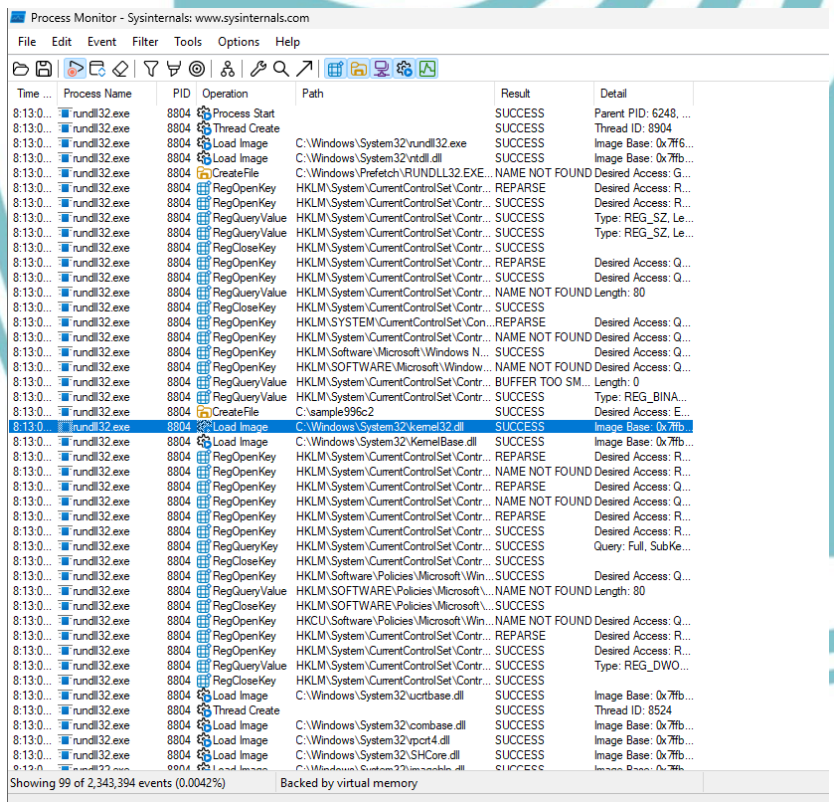
Lampiran 17 Anomali aktivitas jaringan pada sampel 3



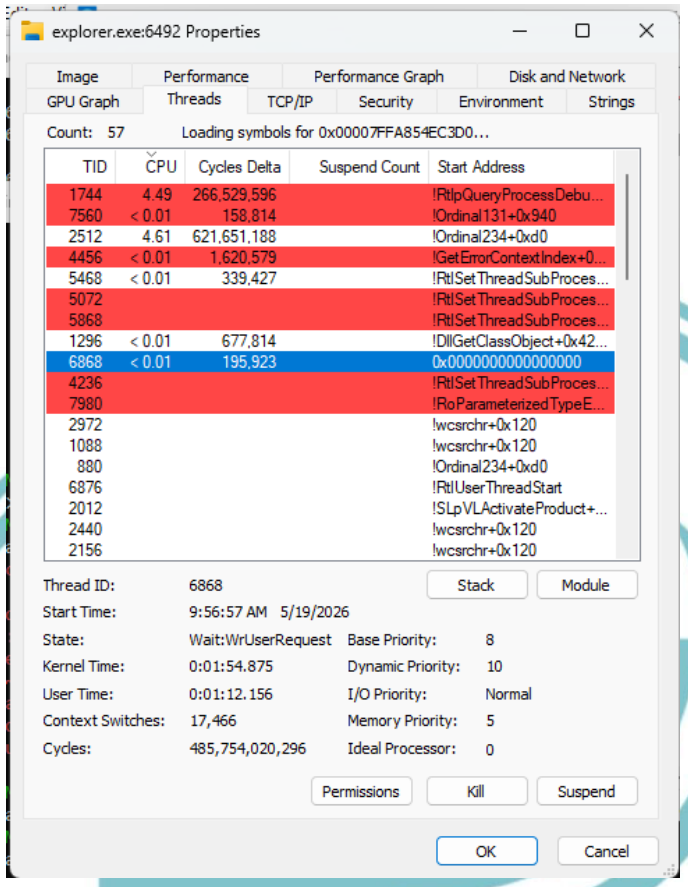
Lampiran 18 Aktivitas file system pada sampel 3



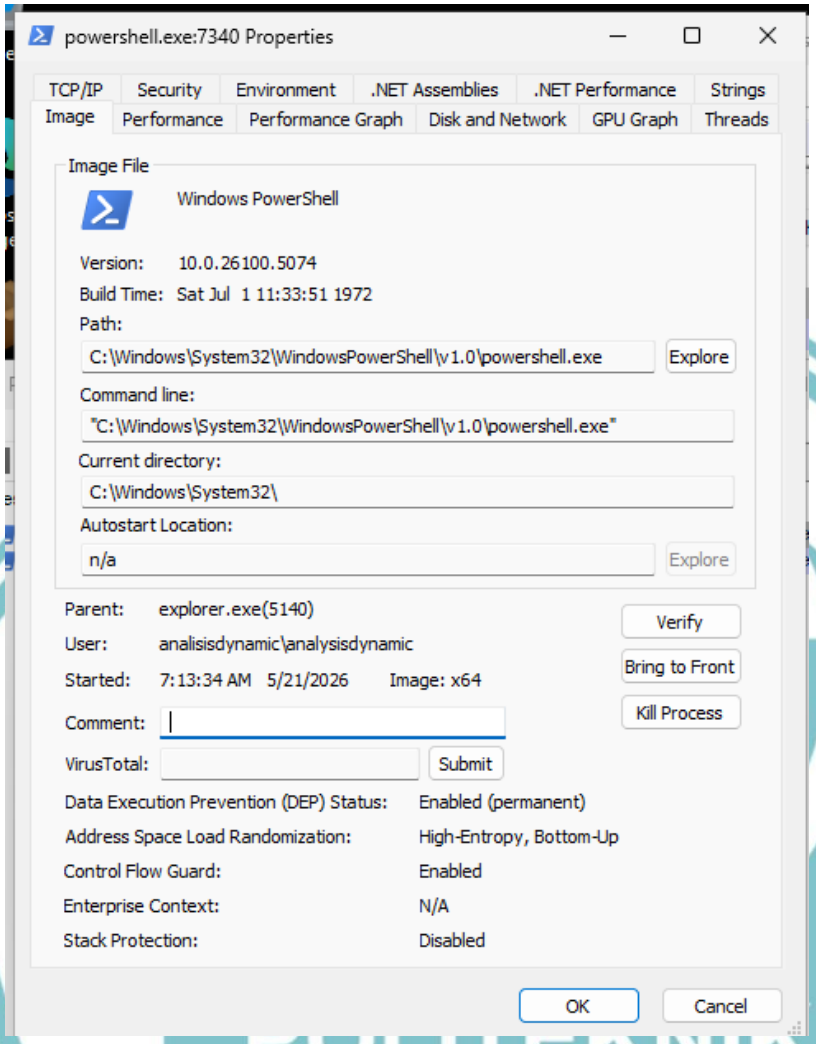
Lampiran 19 Proses monitor pada sampel 4



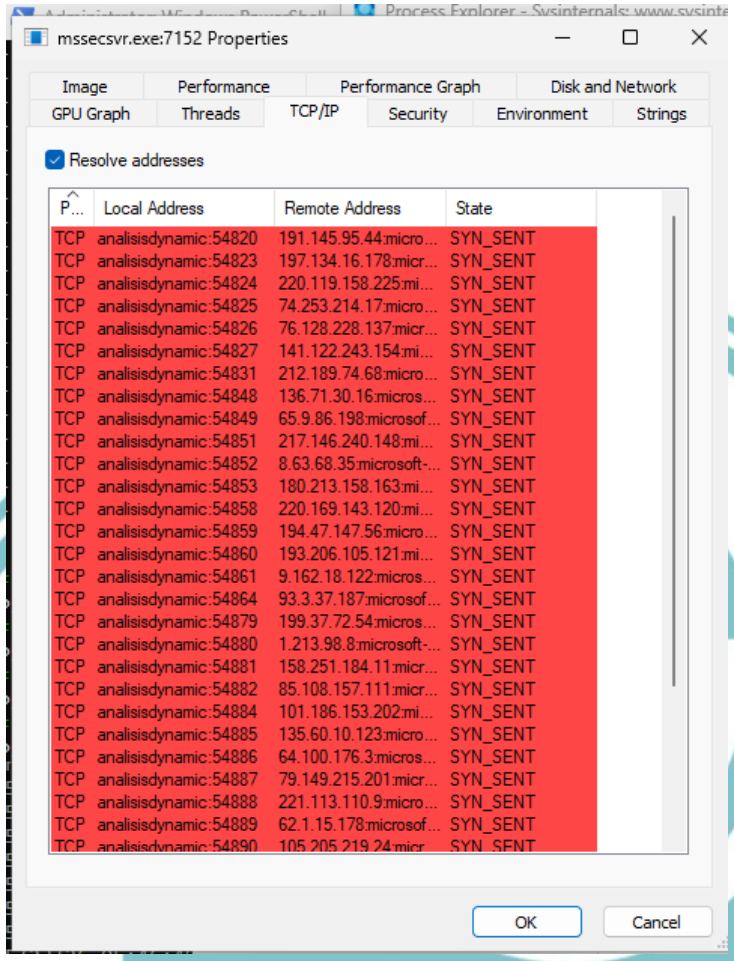
Lampiran 20 Tampilan log Procmon pada sampel 5



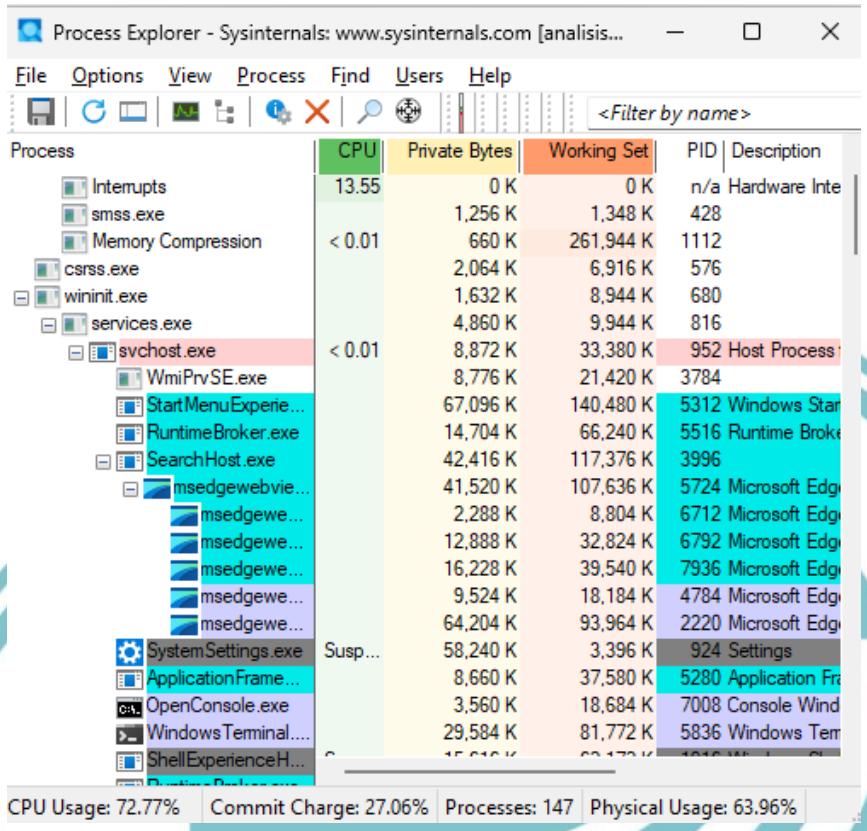
Lampiran 21 Jendela properti Process Explorer pada sampel 1



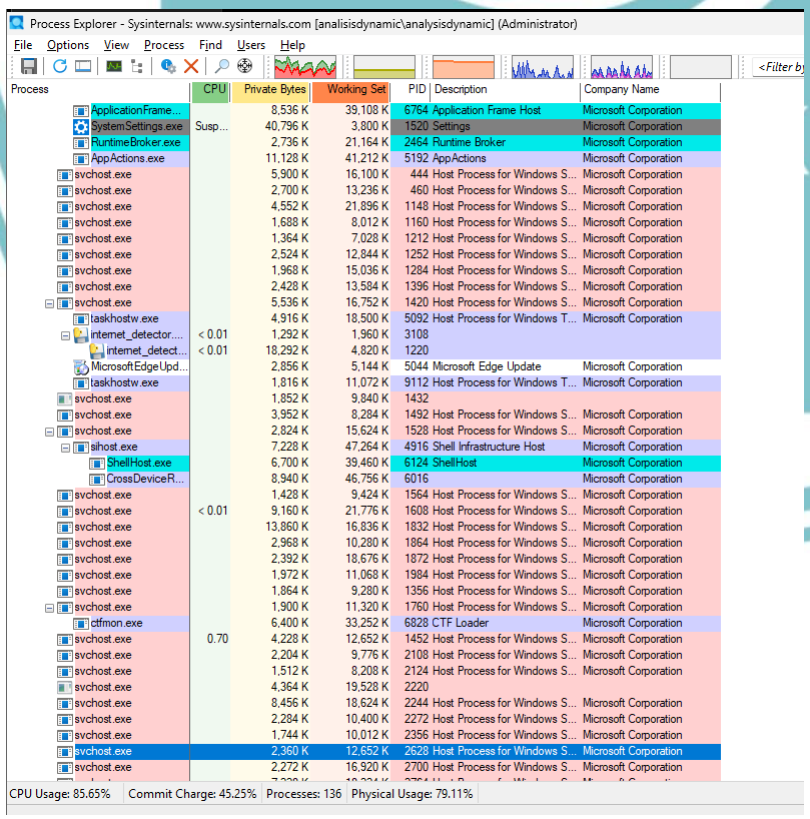
Lampiran 22 Jendela properti Process Explorer pada sampel 2



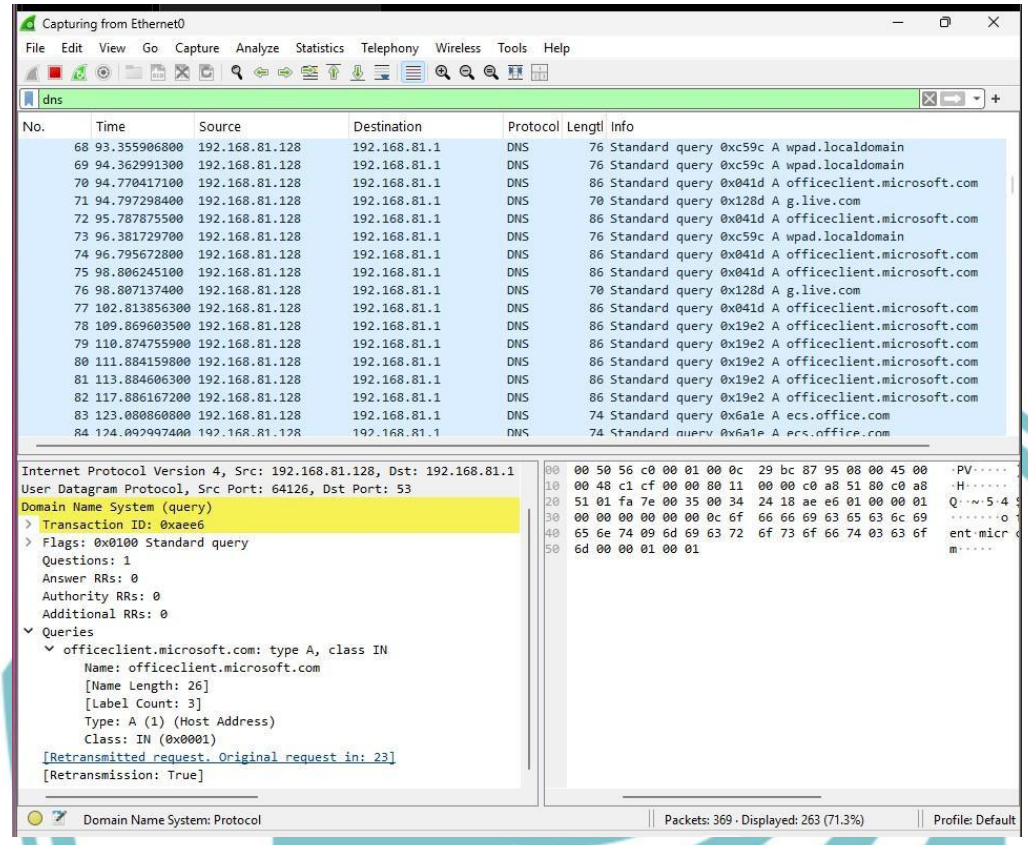
Lampiran 23 Perilaku jaringan pada sampel 3



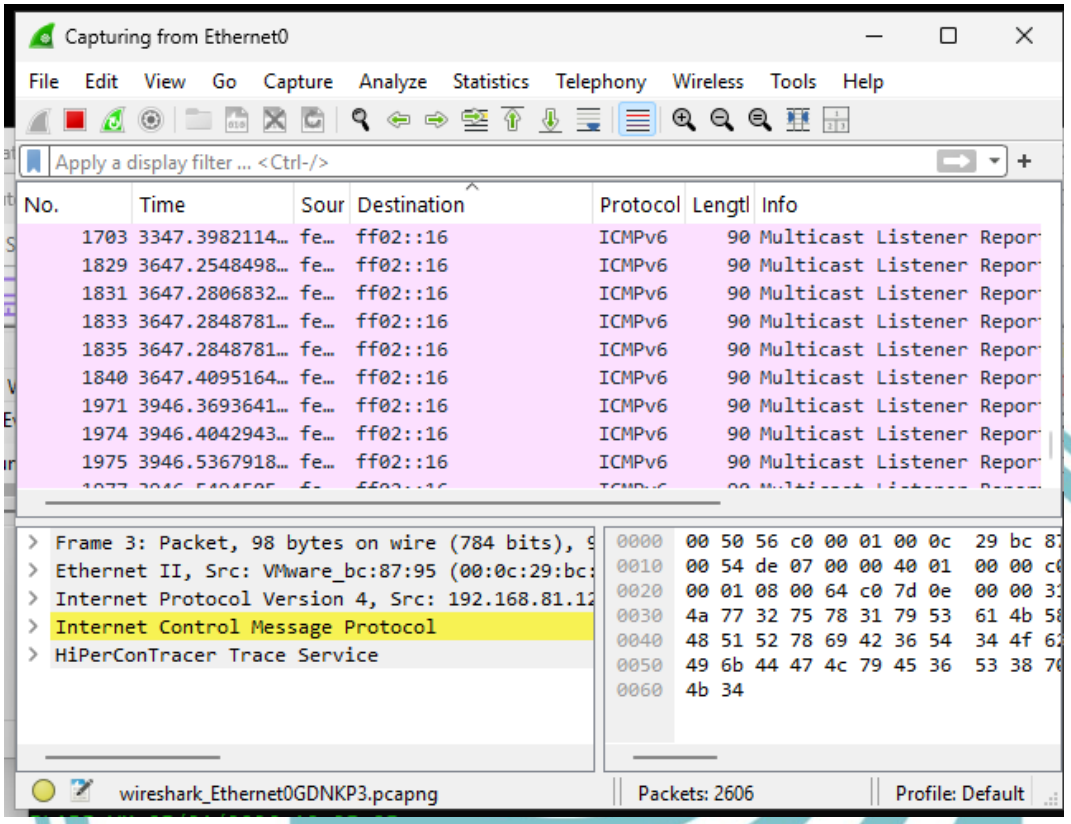
Lampiran 24 Jendela process tree pada sampel 4



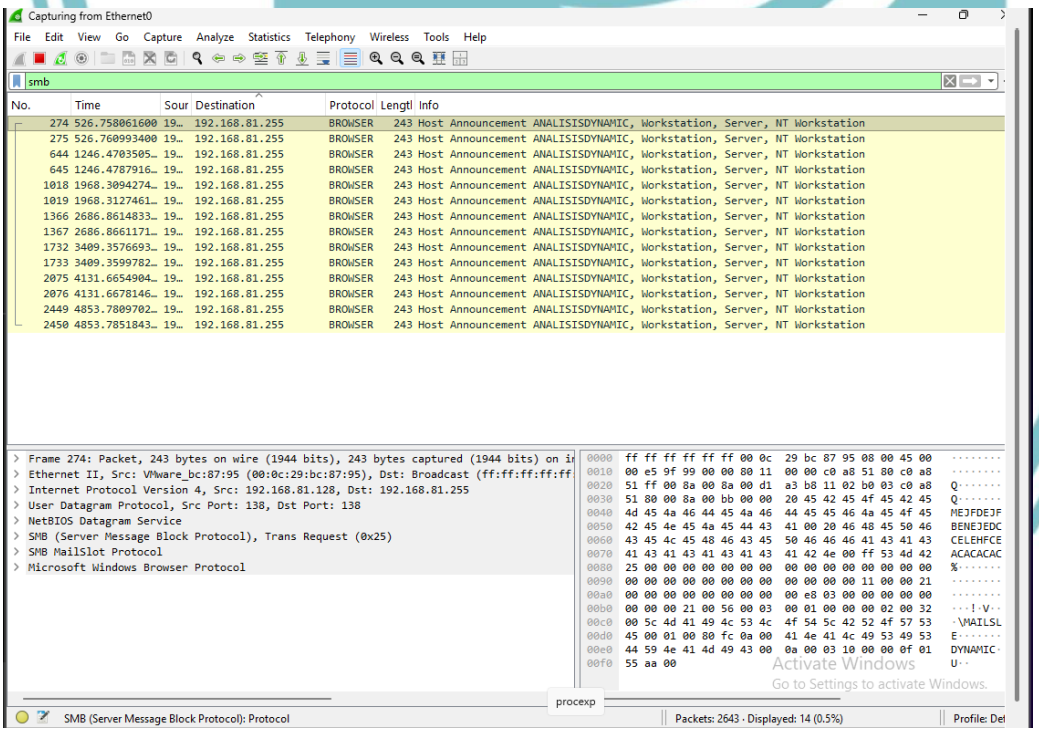
Lampiran 25 Tampilan operasi registri pada sampel 5



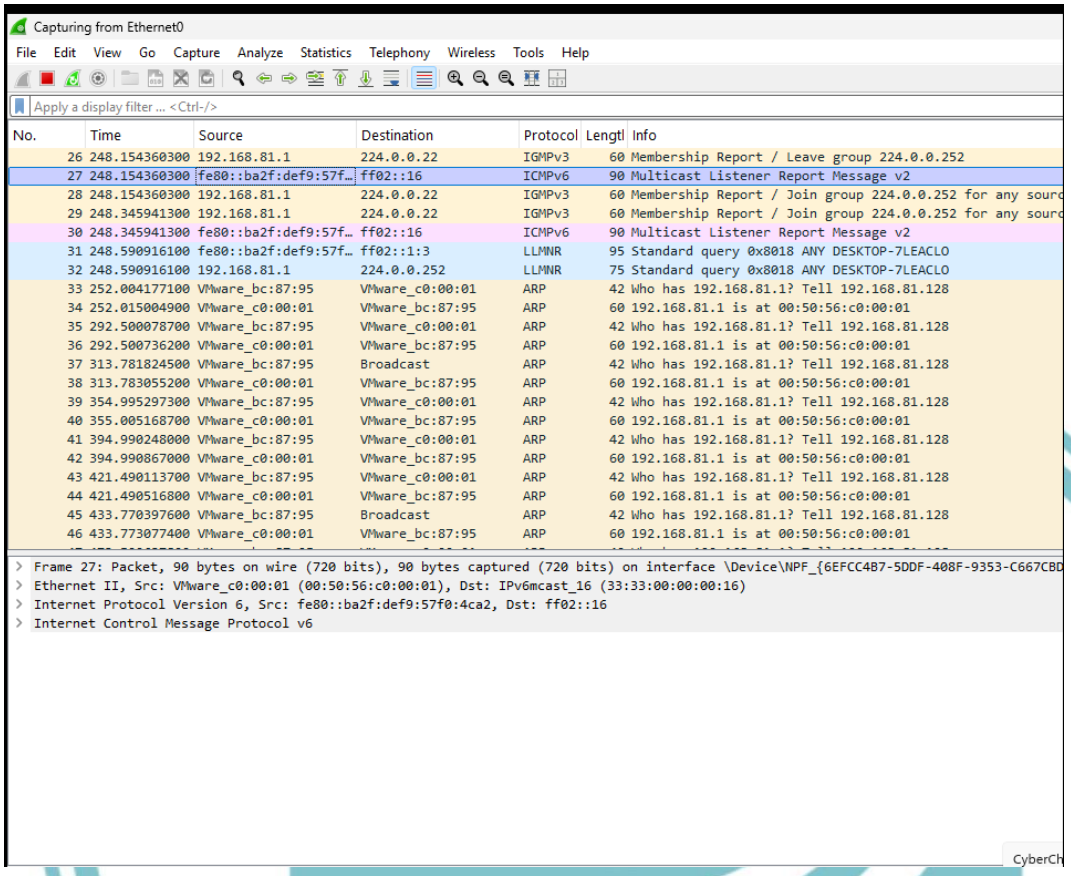
Lampiran 26 Tampilan analisa paket DNS pada sampel 1



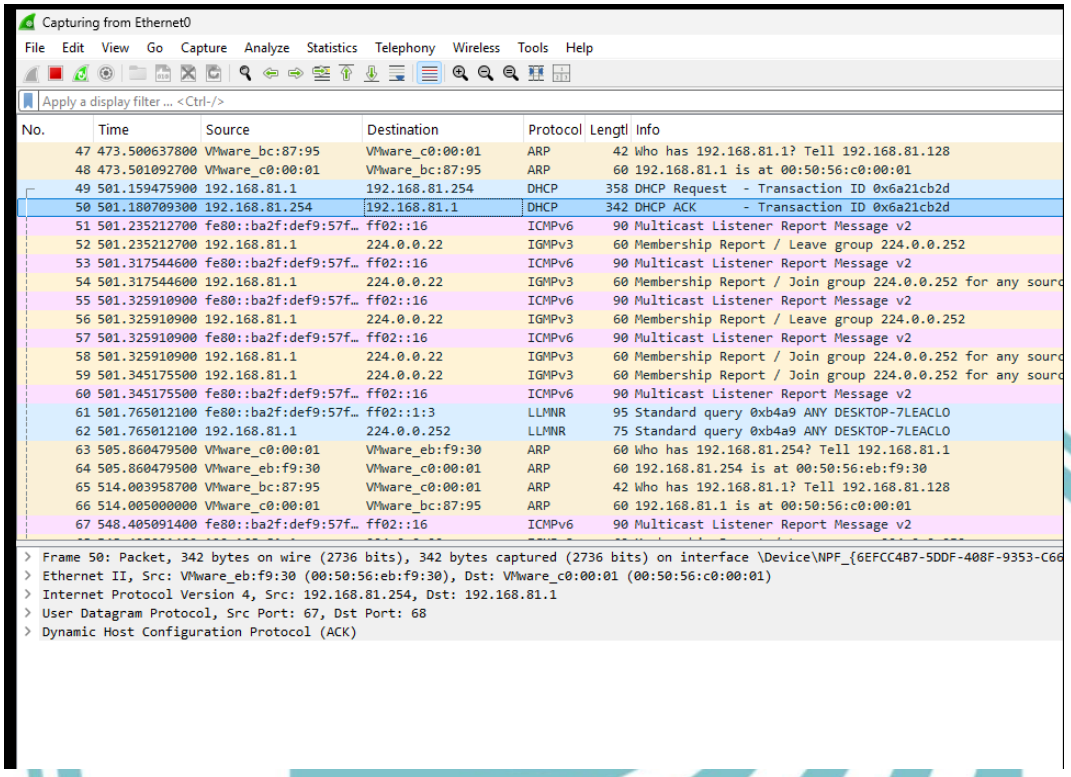
Lampiran 27 Tampilan analisis Wireshark pada sampel 2 (bagian 1)



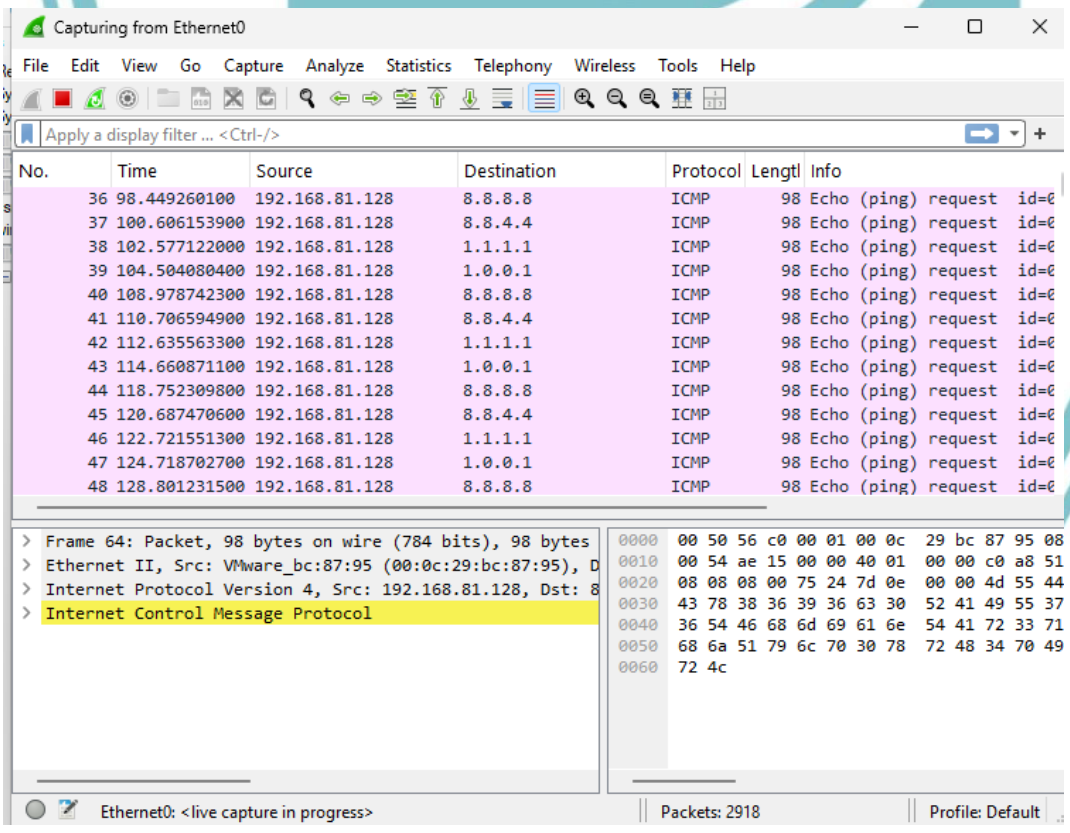
Lampiran 28 Tampilan analisis Wireshark pada sampel 2 (bagian 2)



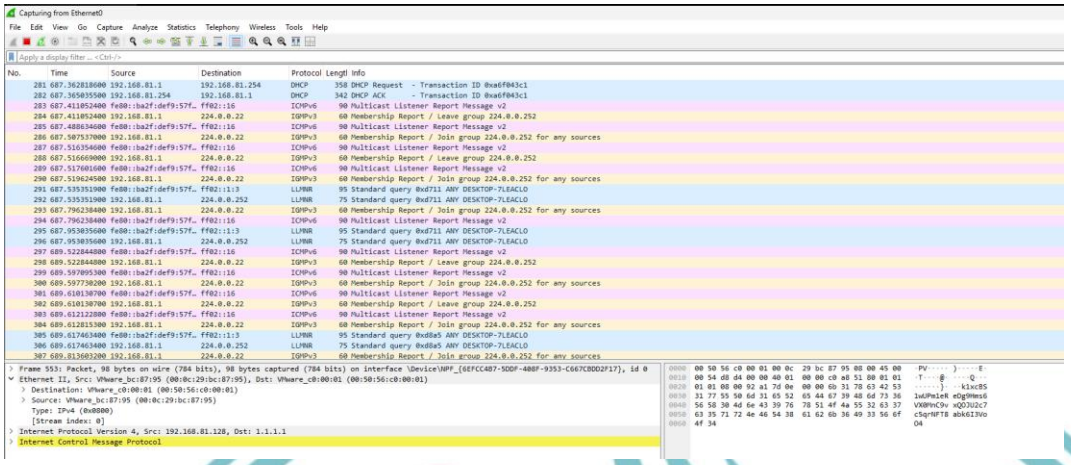
Lampiran 29 Tampilan analisis Wireshark pada sampel 3 (bagian 1)



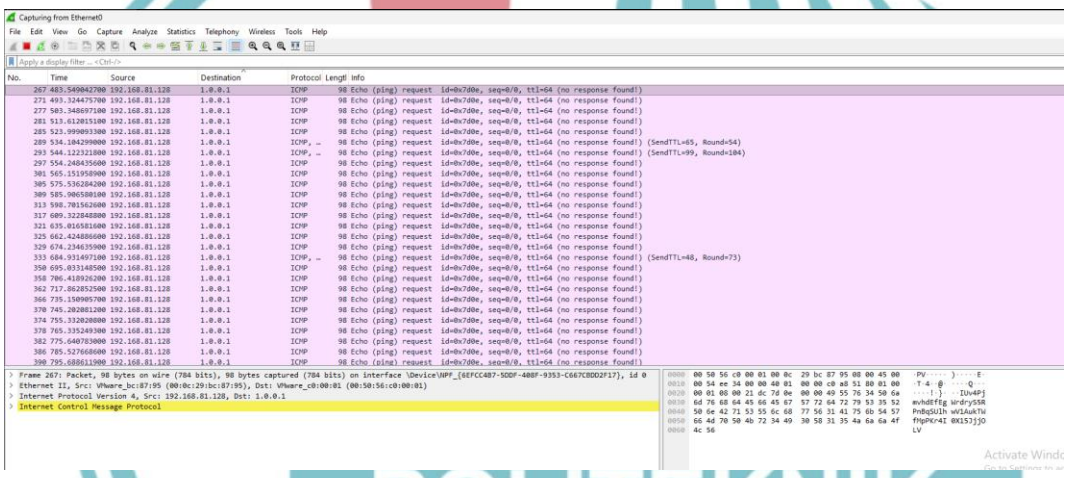
Lampiran 30 Tampilan analisis Wireshark pada sampel 3 (bagian 2)



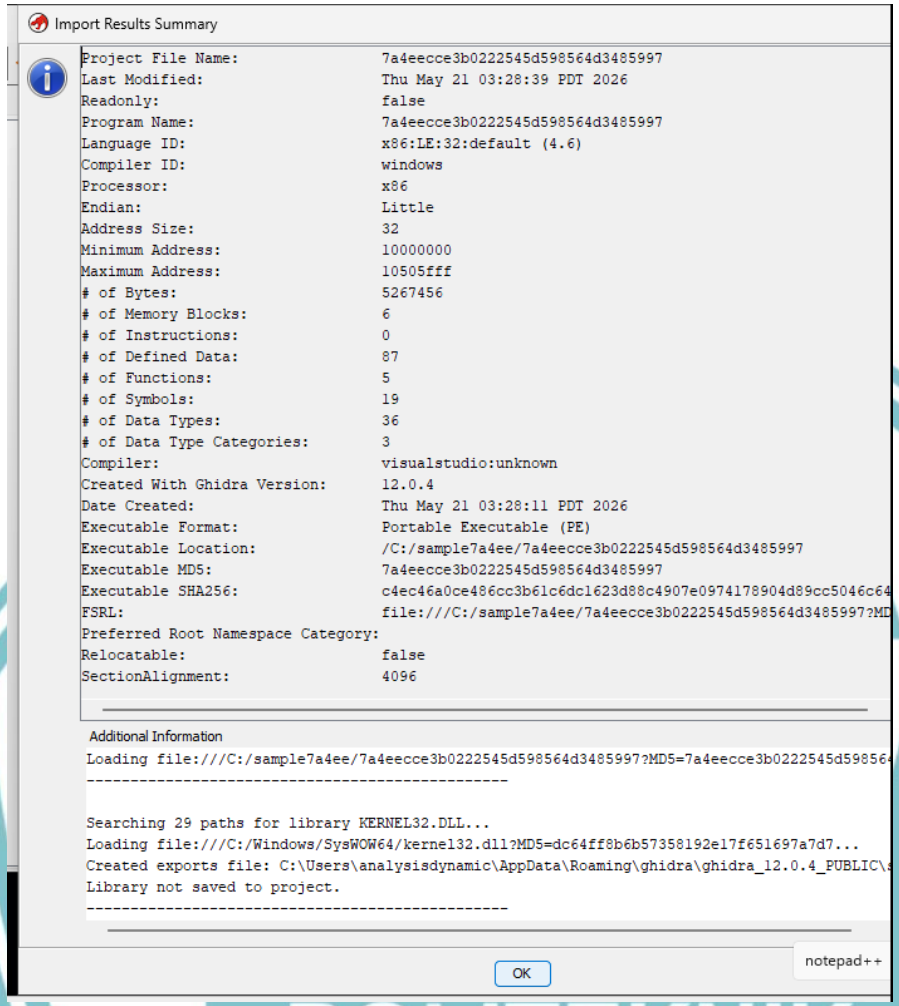
Lampiran 31 Tampilan analisis Wireshark pada sampel 4 (bagian 1)



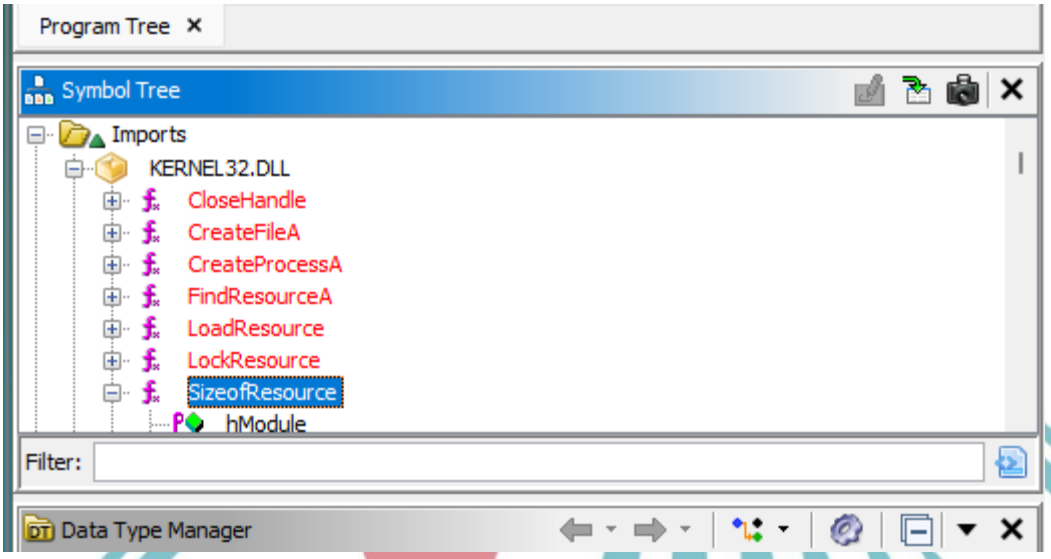
Lampiran 32 Tampilan analisis Wireshark pada sampel 4 (bagian 2)



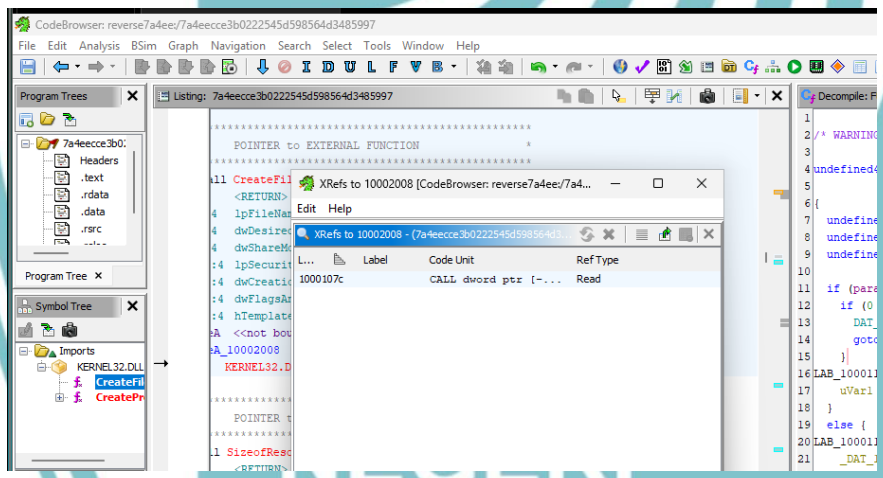
Lampiran 33 Tampilan analisis Wireshark pada sampel 5



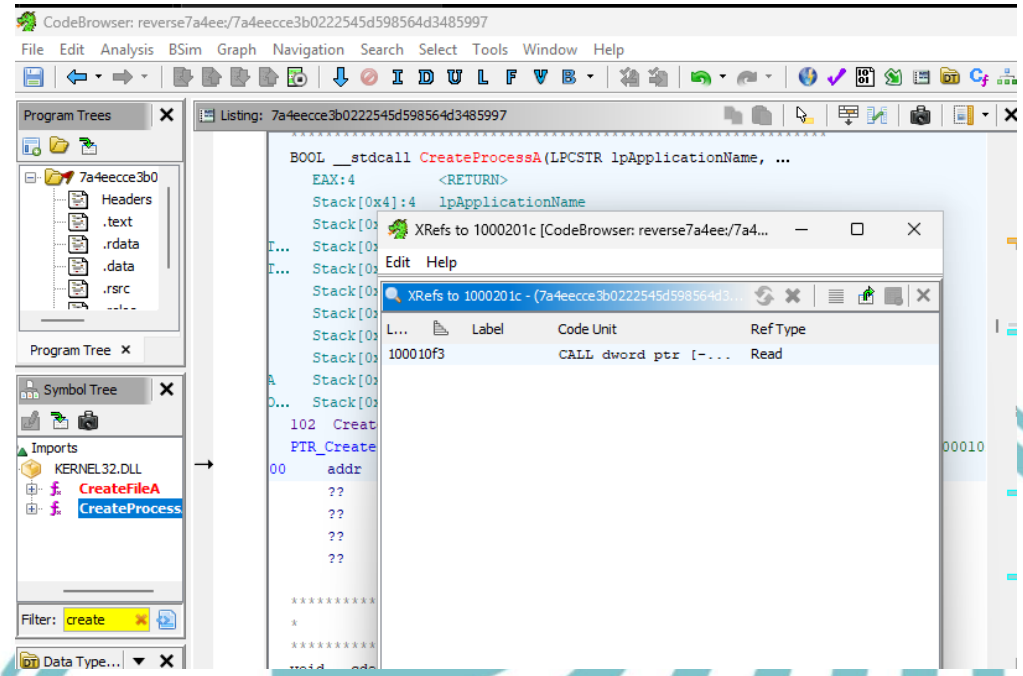
Lampiran 34 Tampilan daftar fungsi pada sampel 1



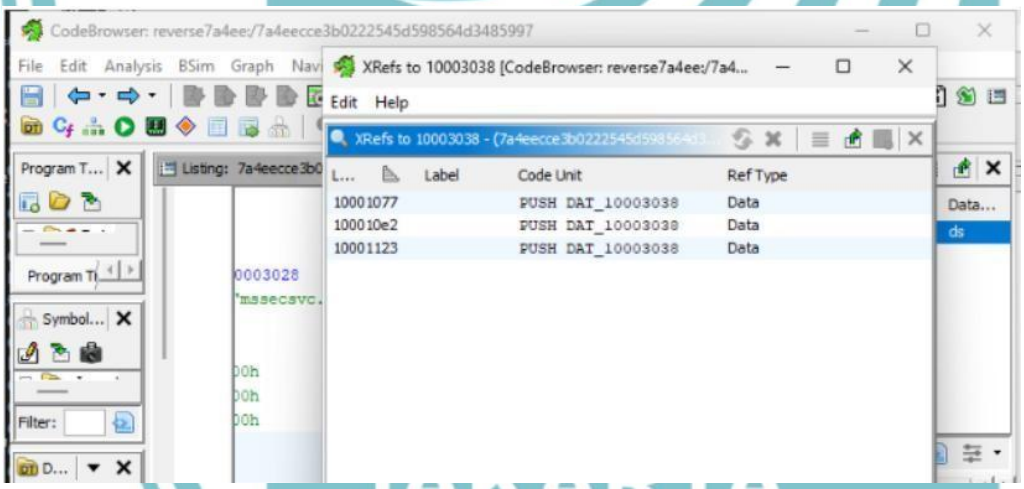
Lampiran 35 Tampilan Symbol Tree pada sampel 1



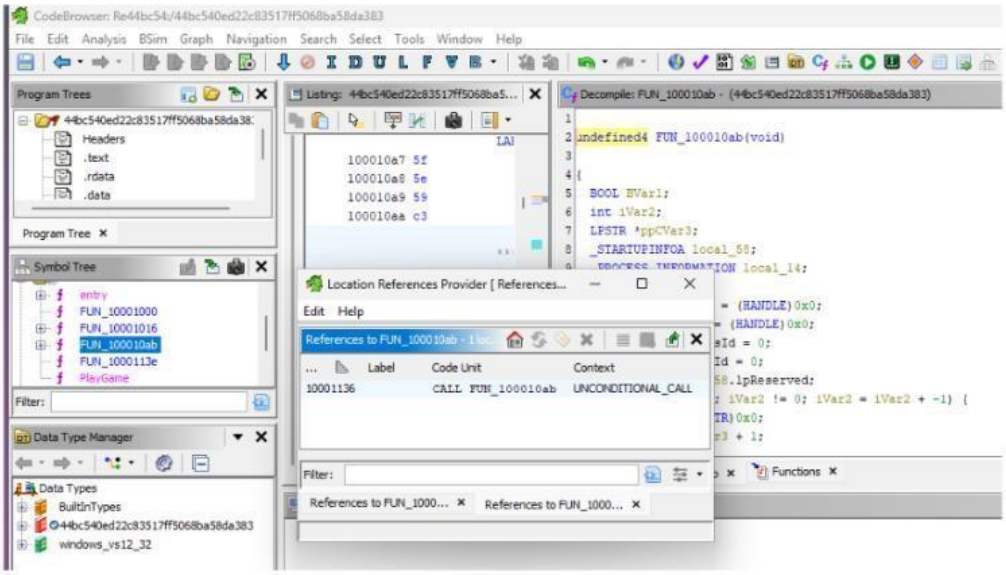
Lampiran 36 Tampilan Cross-Reference (XRefs) pada sampel 1



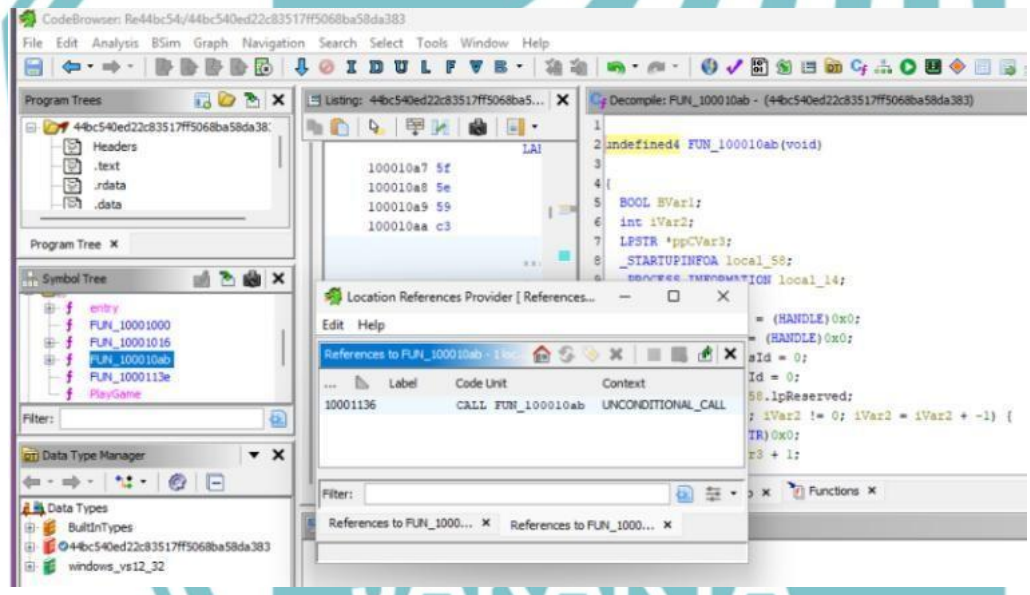
Lampiran 37 Tampilan fungsi CreateProcessA pada sampel 1



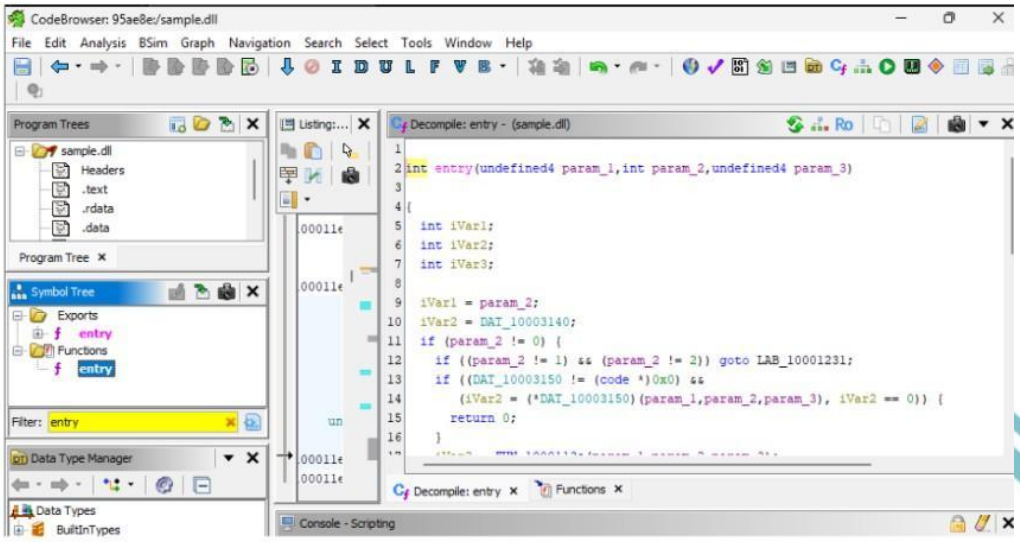
Lampiran 38 Tampilan DAT_10003038 pada sampel 1



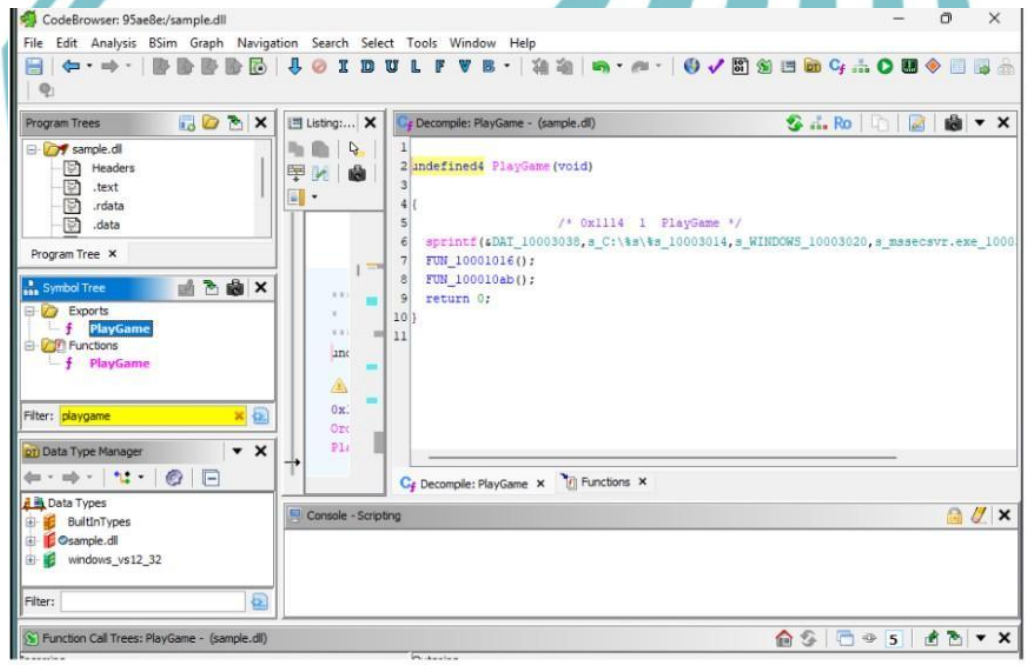
Lampiran 39 Tampilan perintah eksekusi cmd.exe pada sampel 2



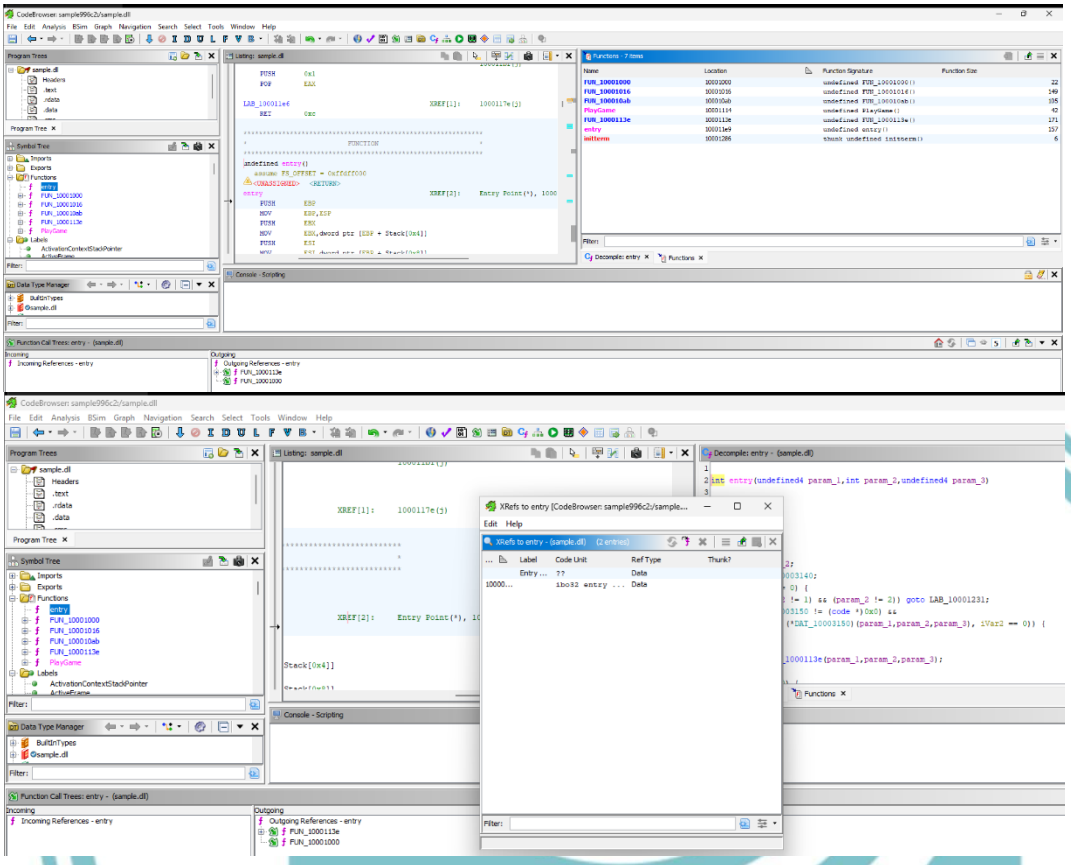
Lampiran 40 mpilan strings parameter layanan pada sampel 2



Lampiran 41 Tampilan decompile pada sampel 3



Lampiran 42 Tampilan decompile pada sampel 4



Lampiran 43 Tampilan Ghidra pada sampel 5