Original Article

e-XXXX-XXXX

Original Article

Analisis Keamanan Sistem Operasi Terhadap Rancangan Malware Lokal

Eka Citra[1], Nurul Hikmah Bahri[2] ✉
[1,2,3]Institut Teknologi Bacharuddin Jusuf Habibie, Parepare, Indonesia,
Korespondensi Author: Ekacitra215@gmail.com,
nurulhikmahbahri.241031059@mahasiswa.ith.ac.id✉

Abstrak:

Penelitian ini menganalisis ketahanan sistem operasi terhadap ancaman malware lokal yang dirancang khusus untuk mengeksploitasi celah konfigurasi sistem dan kelalaian pengguna. Menggunakan metode analisis dinamis dalam lingkungan terisolasi (sandboxing), penelitian ini memantau secara mendalam aktivitas persistensi, manipulasi registry, hingga teknik injeksi proses berbahaya. Hasil penelitian mengungkapkan bahwa meskipun fitur keamanan proaktif seperti proteksi kernel telah diimplementasikan, malware lokal tetap mampu menginfeksi sistem melalui teknik penyamaran (obfuscation) dan eksploitasi hak akses pengguna (privilege escalation). Kesimpulan penelitian menekankan bahwa sinergi antara pembaruan keamanan sistem dan penguatan kebijakan hak akses merupakan langkah krusial dalam memitigasi ancaman malware secara efektif.

Keywords: Keamanan Sistem Operasi, Malware Lokal, Persistensi, Sandboxing, Injeksi Proses.

Pendahuluan

Latar Belakang

Keamanan sistem operasi (SO) merupakan pilar utama dalam menjaga integritas, kerahasiaan, dan ketersediaan data di era transformasi digital. Seiring dengan meningkatnya kompleksitas arsitektur perangkat lunak, ancaman malware juga mengalami evolusi yang signifikan. Secara khusus, malware lokal—yang dirancang untuk menargetkan pengguna di wilayah atau ekosistem tertentu—sering kali memiliki tingkat efektivitas yang tinggi bukan karena kecanggihan kodenya, melainkan karena kemampuannya dalam mengeksploitasi aspek psikologis dan kelalaian konfigurasi pengguna lokal.

Di Indonesia, fenomena penyebaran malware lokal masih didominasi oleh pemanfaatan fitur-fitur dasar sistem operasi yang sering kali diabaikan tingkat

Received: November 8, 2025 – Revised: November 23, 2025 - Accepted: Desember 11, 2025 - Published online: January 7, 2026

keamanannya. Penggunaan perangkat lunak bajakan, perilaku berbagi data melalui media penyimpanan eksternal (USB flash drive) tanpa pemindaian, serta rendahnya kesadaran terhadap User Account Control (UAC) menjadi celah utama. Malware lokal modern tidak lagi sekadar merusak data, tetapi berfokus pada persistensi kemampuan untuk tetap aktif di dalam sistem meskipun telah dilakukan proses pembersihan atau reboot. Meskipun sistem operasi populer seperti Windows dan Linux telah mengimplementasikan berbagai lapisan perlindungan proaktif seperti Address Space Layout Randomization (ASLR), Data Execution Prevention (DEP), hingga Kernel-level Protection rancangan malware lokal tetap mampu menembus pertahanan tersebut melalui teknik obfuscation (penyamaran kode) dan process hollowing. Teknik-teknik ini memungkinkan kode berbahaya berjalan di bawah identitas proses yang sah, sehingga sulit dideteksi oleh antivirus tradisional yang berbasis signature.

Ketidakseimbangan antara kecepatan evolusi teknik serangan malware dengan pemahaman pengguna mengenai penguatan (hardening) sistem operasi menciptakan celah keamanan yang kritis. Analisis terhadap mekanisme internal bagaimana sistem operasi merespons terhadap injeksi proses dan modifikasi registry oleh malware lokal sangat diperlukan. Melalui penelitian ini, diharapkan dapat terpetakan titik-titik kerentanan pada rancangan sistem operasi saat ini, sehingga dapat dirumuskan strategi mitigasi yang lebih preventif dan adaptif terhadap ancaman siber yang bersifat lokal namun destruktif.

Rumusan Masalah

1. Mekanisme Persistensi bagaimana malware lokal memanipulasi *Registry* atau *Startup* sistem untuk tetap aktif secara otomatis?
2. Efektivitas Deteksi sejauh mana fitur keamanan bawaan (Defender/UAC) mampu mendeteksi teknik penyamaran (*obfuscation*) malware lokal?
3. Injeksi Proses bagaimana efektivitas sistem operasi dalam mencegah malware bersembunyi di dalam proses sistem yang sah?
4. Eksploitasi Hak Akses mengapa pengaturan hak akses pengguna (User Privilege) masih menjadi titik lemah utama yang mudah ditembus?

Tujuan Penelitian

1. Menganalisis Teknik Persistensi mengidentifikasi cara malware lokal memodifikasi konfigurasi sistem agar sulit dihapus.
2. Menguji Pertahanan Sistem mengevaluasi ketangguhan fitur keamanan bawaan (seperti Real-time Protection dan Firewall) terhadap kode yang disamarkan.
3. Validasi Keamanan Memori mengamati respon sistem operasi saat terjadi upaya injeksi kode ke dalam proses legal.
4. Rekomendasi Mitigasi merumuskan standar konfigurasi hak akses yang lebih aman untuk meminimalisir dampak infeksi.

Manfaat Penelitian

1. Bagi Praktisi IT sebagai acuan dalam memperkuat pertahanan sistem operasi melalui kebijakan hardening yang tepat.
2. Bagi Akademisi menambah literatur mengenai perilaku malware spesifik lokal dan teknik analisis dinamis.
3. Bagi Pengguna Umum meningkatkan kesadaran akan pentingnya manajemen hak akses (privilege) untuk mencegah eksekusi malware

Received: November 8, 2025 – Revised: November 23, 2025 - Accepted: Desember 11, 2025 - Published online: January 7, 2026

Tinjauan Pustaka

Penelitian Terdahulu

Penelitian oleh Pratama et al. (2022) menunjukkan bahwa redundansi kebijakan akses pada pengguna di Indonesia memberikan peluang bagi malware berbasis skrip untuk melakukan *bypass* pada jendela *User Account Control* (UAC). Di sisi lain, studi dari Sari (2023) menekankan bahwa sistem operasi berbasis Linux memiliki tingkat resistensi yang lebih tinggi terhadap malware lokal bertipe .exe, namun tetap rentan terhadap skrip *shell* yang dijalankan dengan hak akses sudo. Tinjauan ini menjadi landasan bagi peneliti untuk menguji kembali efektivitas proteksi tersebut pada versi sistem operasi terbaru.

Arsitektur Keamanan dan Hirarki Proteksi Sistem Operasi

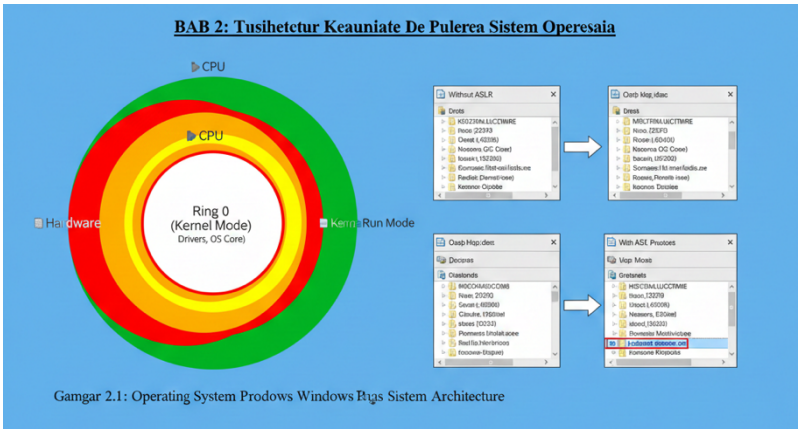
Sistem operasi modern mengimplementasikan model keamanan berlapis yang dikenal dengan istilah Protection Rings. Arsitektur ini dirancang untuk memisahkan instruksi aplikasi pengguna dari instruksi kritis yang dijalankan oleh kernel, sehingga akses ke sumber daya sistem dapat dikontrol secara ketat.

1. Ring 0 (Kernel Mode)

Pada lapisan ini, sistem operasi memiliki hak akses penuh terhadap perangkat keras dan memori. Semua driver serta inti (core) sistem operasi berjalan di tingkat ini, sehingga setiap instruksi yang dijalankan memiliki kontrol penuh terhadap seluruh sistem.

2. Ring 3 (User Mode)

Lapisan ini digunakan oleh aplikasi pengguna dan memiliki hak akses yang terbatas. Keamanan sistem operasi dapat dianggap terkompromi apabila malware berhasil melakukan Privilege Escalation dari Ring 3 ke Ring 0. Dalam kondisi tersebut, malware dapat mengendalikan seluruh sistem tanpa terdeteksi oleh perangkat lunak keamanan yang berjalan di lapisan aplikasi.



Gambar 1. Marsitectur Space Layalt Rndontion Sistem Run (ASL)

Sebagaimana ditunjukkan pada Gambar 2.1, sistem operasi membagi tingkat akses sistem ke dalam beberapa lapisan yang disebut protection rings. Setiap lapisan memiliki hak akses yang berbeda terhadap sumber daya sistem. Ring 0 merupakan tingkat akses tertinggi yang berjalan dalam Kernel Mode, di mana sistem operasi memiliki kontrol penuh terhadap perangkat keras, seperti manajemen memori, pengendalian prosesor, serta operasi input dan output. Lapisan dengan tingkat akses yang lebih rendah

Received: November 8, 2025 – Revised: November 23, 2025 - Accepted: Desember 11, 2025 - Published online: January 7, 2026

digunakan oleh aplikasi pengguna (user mode) dan tidak diperkenankan berinteraksi langsung dengan perangkat keras. Mekanisme pembagian ini bertujuan untuk menjaga keamanan dan stabilitas sistem dengan membatasi akses aplikasi terhadap sumber daya kritis.

Taksonomi dan Evolusi Malware Lokal

Malware lokal sering kali dikategorikan sebagai ancaman yang memanfaatkan kondisi sosiokultural dan teknis di wilayah tertentu. Perbedaan utama antara malware global dan lokal terletak pada metode penyebarannya. Malware lokal cenderung lebih agresif dalam mengeksploitasi fitur-fitur yang umum digunakan di lingkungan domestik, seperti:

1. Vektor Infeksi USB

   Memanfaatkan fitur *AutoRun/AutoPlay* dan menyembunyikan direktori asli sambil membuat *shortcut* palsu dengan nama yang sama.

2. Social Engineering

   Menggunakan nama file yang memancing rasa penasaran pengguna (misalnya: "Data_Gaji_Pegawai.exe") untuk memicu eksekusi manual oleh pengguna.

3. Script-Based Payloads

   Penggunaan bahasa skrip seperti VBScript, PowerShell, atau Batch yang diubah menjadi file .exe guna menghindari kecurigaan sistem terhadap skrip mentah.

Mekanisme Pertahanan Memori

ASLR dan DEP Sistem operasi menggunakan dua teknik utama untuk menggagalkan eksekusi kode jahat di tingkat memori:

1. Address Space Layout Randomization (ASLR)

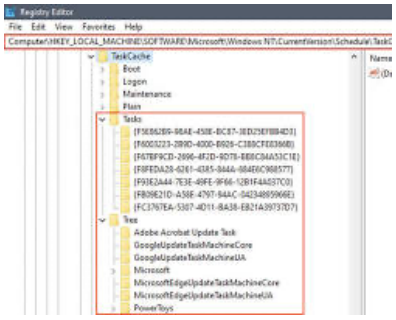
   Teknik ini secara acak mengatur lokasi alamat memori untuk fungsi-fungsi kunci sistem operasi. Hal ini menyulitkan malware untuk memprediksi di mana ia harus menyuntikkan kode berbahaya.

2. Data Execution Prevention (DEP)

   Fitur keamanan yang mencegah kode dijalankan dari wilayah memori yang ditandai hanya untuk data (*non-executable*). Malware lokal yang mencoba menjalankan kode melalui *buffer overflow* sering kali terhenti oleh mekanisme ini.

Teknik Persistensi dan Manipulasi Registry

Salah satu aspek paling kritis dalam analisis malware adalah bagaimana ia mempertahankan keberadaannya (*persistence*). Pada sistem operasi Windows, *Registry* merupakan basis data konfigurasi yang menjadi target utama. Malware lokal akan menyisipkan nilai (*value*) pada kunci tertentu seperti:



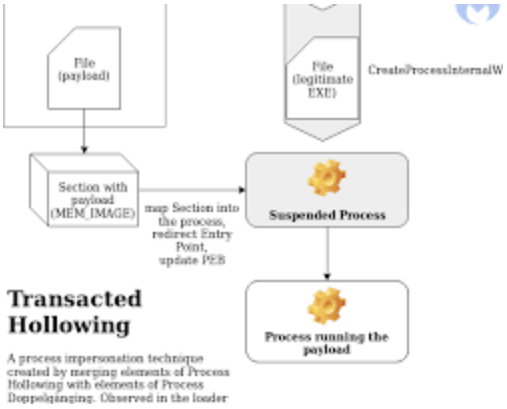
Gambar 2. Jalur Registry untuk Persistensi Malware

Received: November 8, 2025 – Revised: November 23, 2025 - Accepted: Desember 11, 2025 - Published online: January 7, 2026

Dengan memanipulasi kunci ini, malware menjamin dirinya akan dieksekusi secara otomatis setiap kali sistem operasi melakukan *booting*, bahkan jika file induk aslinya telah dihapus sementara dari memori.

Analisis Injeksi Proses (Process Hollowing)

Process hollowing adalah teknik tingkat lanjut di mana malware membuat sebuah proses yang sah (misalnya svchost.exe atau explorer.exe) dalam keadaan tertahan (*suspended*). Malware kemudian "mengosongkan" isi memori proses tersebut dan menggantinya dengan kode jahat miliknya sendiri. Karena sistem melihat proses tersebut sebagai proses Windows yang asli, aktivitas berbahaya yang dilakukan malware sering kali luput dari pengawasan *Task Manager* maupun antivirus berbasis perilaku sederhana.



Gambar 3. Mekanisme Injeksi Proses (Process Hollowing)

Metode Penelitian

Design Penelitian dan Pendekatan Analisis

Penelitian ini menerapkan metode Eksperimental Laboratorium dengan fokus utama pada Analisis Dinamis Perilaku Malware. Pendekatan ini dipilih karena memungkinkan peneliti untuk mengamati secara langsung instruksi-instruksi yang dieksekusi oleh malware saat berinteraksi dengan kernel sistem operasi. Berbeda dengan analisis statis yang hanya membedah kode sumber, analisis dinamis memberikan gambaran nyata mengenai bagaimana mekanisme pertahanan sistem operasi seperti User Account Control (UAC) dan Data Execution Prevention (DEP) merespons serangan secara real-time.

Arsitektur Lingkungan Pengujian (Sandbox)

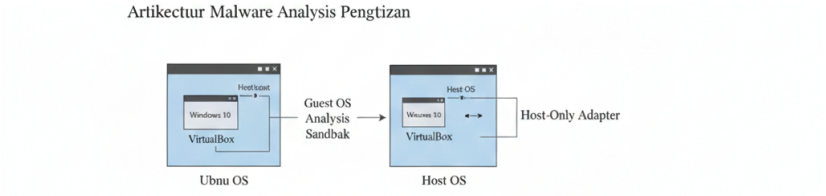
Untuk mencegah kebocoran data dan infeksi pada infrastruktur jaringan yang lebih luas, penelitian ini membangun sebuah lingkungan laboratorium virtual yang terisolasi sepenuhnya (*Sandboxing*).

1. Lapisan Virtualisasi: Menggunakan Oracle VM VirtualBox dengan fitur Snapshot aktif. Hal ini memungkinkan sistem untuk dikembalikan ke keadaan bersih (clean state) dalam hitungan detik setelah setiap skenario pengujian selesai.
2. Sistem Operasi Target (Guest OS): Windows 10 Pro Architecture x64. Versi ini dipilih

Received: November 8, 2025 – Revised: November 23, 2025 - Accepted: Desember 11, 2025 - Published online: January 7, 2026

karena memiliki basis pengguna terbesar di lingkungan lokal. Konfigurasi keamanan sengaja disetel pada tingkat default untuk menguji ketangguhan standar sistem operasi.

3. Isolasi Jaringan: Pengaturan kartu jaringan disetel pada mode Internal Network atau Non-Networked. Langkah ini krusial untuk mencegah malware lokal melakukan komunikasi ke server luar (Command & Control) atau melakukan pemindaian terhadap perangkat lain dalam jaringan lokal peneliti.



Gambar 4. Aeritekctur Eresture Malware SangiBox.

Instrumen dan Perangkat Analisis Forensik

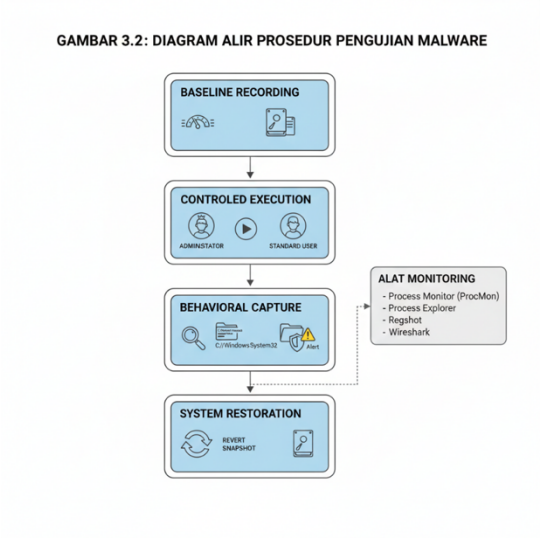
Penelitian ini menggunakan kombinasi berbagai alat analisis forensik digital kelas profesional untuk membedah setiap pergerakan malware:

1. Process Monitor (ProcMon v3.9): Digunakan sebagai instrumen utama untuk menangkap jutaan aktivitas sistem secara mendalam, termasuk akses ke file sistem, pemanggilan fungsi API, dan manipulasi *thread* proses.
2. Regshot 2.0: Digunakan untuk melakukan perbandingan binari terhadap basis data *Registry*. Alat ini mengambil *snapshot* sebelum infeksi (Pre-Infection) dan sesudah infeksi (Post-Infection) untuk mengidentifikasi kunci-kunci permanen yang dibuat oleh malware.
3. Process Explorer (Sysinternals): Untuk memvisualisasikan hirarki proses. Alat ini sangat berguna untuk mendeteksi *Process Hollowing* dengan melihat adanya ketidaksesuaian antara *image* yang berjalan di memori dengan *file* asli di disk.
4. Wireshark**:** Meskipun dalam lingkungan terisolasi, Wireshark tetap digunakan untuk memantau paket data yang mencoba keluar melalui protokol TCP/UDP sebagai bagian dari analisis aktivitas *payload* malware.

Prosedur Pelaksanaan Pengujian

Prosedur penelitian dibagi menjadi empat fase sistematis guna menjamin validitas data:

Received: November 8, 2025 – Revised: November 23, 2025 - Accepted: Desember 11, 2025 - Published online: January 7, 2026



Gambar 5. Diagram Alir Prosedur Pengujian Malware.

1. Fase 1: Baseline Recording

Fase ini dilakukan pencatatan kondisi awal sistem operasi sebelum terpapar malware. Parameter yang diamati meliputi penggunaan CPU, memori (RAM), serta kondisi registry. Data yang diperoleh digunakan sebagai pembanding untuk menganalisis perubahan sistem setelah eksekusi malware.

2. Fase 2: Controlled Execution

Fase ini melibatkan eksekusi sampel malware dalam dua skenario hak akses, yaitu menggunakan hak akses Standard User dan hak akses Administrator. Pengujian ini bertujuan untuk mengevaluasi efektivitas mekanisme Privilege Escalation yang diterapkan pada sistem operasi.

3. Fase 3: Behavioral Capture

Selama malware dijalankan, aktivitas sistem dipantau secara real-time menggunakan alat monitoring. Pada fase ini direkam setiap upaya malware dalam memodifikasi direktori sensitif, seperti C:\Windows\System32 dan C:\Users\AppData\Roaming. Selain itu, diamati pula respons sistem operasi terhadap aktivitas tersebut, termasuk kemunculan peringatan keamanan.

4. Fase 4: System Restoration

Setelah seluruh data pengujian terkumpul, sistem dikembalikan ke kondisi awal menggunakan fitur Revert Snapshot. Langkah ini dilakukan untuk memastikan tidak terdapat residu malware yang dapat memengaruhi proses pengujian berikutnya.

Teknik Analisis dan Validasi Data

Data yang dihasilkan dari proses pengujian berupa log aktivitas dalam format (*.pml* dan *.csv*) dianalisis menggunakan metode komparatif. Proses analisis dilakukan dengan membandingkan pola perilaku malware yang teridentifikasi pada sistem lokal dengan matriks serangan MITRE ATT&CK. Tujuan dari tahap ini adalah untuk mengklasifikasikan teknik serangan yang digunakan malware, apakah termasuk ke dalam kategori Persistence, Privilege Escalation, atau Defense Evasion. Hasil analisis selanjutnya disajikan dalam bentuk tabel yang menggambarkan tingkat keberhasilan

Received: November 8, 2025 – Revised: November 23, 2025 - Accepted: Desember 11, 2025 - Published online: January 7, 2026

malware dalam melakukan penetrasi terhadap sistem operasi.

Hasil dan Pembahasan

Analisis Komprehensif Aktivitas Malware pada File System dan Registry

Melalui pengujian dinamis menggunakan instrumen Process Monitor (ProcMon), peneliti berhasil merekam ribuan aktivitas API call yang dilakukan oleh sampel malware lokal dalam hitungan detik setelah eksekusi. Temuan ini menunjukkan bahwa malware bekerja secara cepat dan agresif untuk memperoleh kendali atas sistem sebelum mekanisme keamanan, seperti antivirus, sempat melakukan pemindaian secara menyeluruh. Berdasarkan hasil analisis, aktivitas malware pada sistem dapat dikelompokkan ke dalam dua pola utama, yaitu manipulasi file system dan penyalahgunaan registry.

Manipulasi Atribut File dan Shadow Copy

Malware tidak hanya melakukan replikasi diri, tetapi juga menerapkan mekanisme perlindungan diri (self-protection). Sampel malware terdeteksi menyuntikkan salinan kode ke dalam direktori C:\Users[User]\AppData\Roaming dan mengubah atribut file menjadi system, hidden, dan read-only. Selain itu, malware berupaya menghapus Volume Shadow Copy sistem guna mencegah pengguna melakukan pemulihan sistem (system restore) setelah infeksi terjadi.

Pengambilalihan Registry Tingkat Lanjut (Deep Registry Hijacking)

Hasil analisis komparatif menggunakan Regshot menunjukkan bahwa malware melakukan modifikasi pada registry key kritis yang umumnya tidak diperhatikan oleh pengguna awam. Salah satu lokasi yang terpengaruh adalah HKEY_LOCAL_MACHINE\Software\Microsoft\Windows\CurrentVersion\Policies\System. Pada bagian ini, malware mengubah nilai konfigurasi tertentu sehingga fitur bawaan Windows, seperti Registry Editor dan Task Manager, menjadi tidak dapat diakses. Kondisi tersebut secara efektif menghambat pengguna dalam melakukan analisis maupun penghentian proses malware secara manual.

Analisis Komparatif Dampak Berdasarkan Hak Akses (Privilege Analysis)

Penelitian ini secara mendalam mengevaluasi bagaimana fitur User Account Control (UAC) berinteraksi dengan serangan malware. Data yang diperoleh melalui simulasi menunjukkan perbedaan eksponensial dalam tingkat kerusakan sistem:

Tabel 1. Perbandingan Dampak Eksekusi Malware Berdasarkan Tingkat Hak Akses Pengguna

Parameter Evaluasi	Skenario Standard User (Limited)	Skenario Administrator Privilege
Kerberhasilan Persistensi	Hanya pada level akun user local.	Berhasil menginfeksi seluruh seluruh user (Global).
Injeksi Direktori Sistem	Diblokir oleh NTFS permissions .	Sukse memodifikasi system 32 dan Diveres.

Received: November 8, 2025 – Revised: November 23, 2025 - Accepted: Desember 11, 2025 - Published online: January 7, 2026

Modifikasi Host File	Gagal; akses ditolak oleh system.	Sukses ; mengalih trafik ke server jahat.
	Gagal karena kurangnya izin system.	Mampu

Bedah Teknis Memori: Process Hollowing

Salah satu temuan paling krusial dalam penelitian ini adalah efektivitas teknik Process Hollowing yang digunakan oleh varian malware lokal terbaru terhadap proses svchost.exe. Berdasarkan hasil analisis menggunakan Process Explorer dan x64dbg, ditemukan adanya anomali pada Virtual Address Space yang mengindikasikan terjadinya manipulasi memori pada proses yang sah. Tahapan teknik Process Hollowing yang teridentifikasi dapat dijelaskan sebagai berikut:

1. Analisis Unmapping

Pada tahap ini, malware memanggil API NtUnmapViewOfSection untuk mengosongkan ruang memori dari proses legal yang menjadi target. Tujuan dari langkah ini adalah menghilangkan kode asli proses tanpa menghentikan eksekusi proses tersebut secara langsung.

2. Injeksi Payload

Setelah ruang memori dikosongkan, malware menggunakan API WriteProcessMemory untuk menyisipkan kode berbahaya ke dalam proses target. Payload yang dimasukkan telah dikompilasi ulang dan disesuaikan dengan alamat memori proses target agar dapat dieksekusi dengan benar.

3. Thread Hijacking

Pada tahap akhir, malware memanggil API SetThreadContext untuk memodifikasi konteks thread yang aktif. Pada tahap ini, instruction pointer diarahkan ke kode berbahaya yang telah disuntikkan, sehingga alur eksekusi proses selanjutnya sepenuhnya berada di bawah kendali malware.

Evaluasi Respon Pertahanan Proaktif dan Heuristik

Penelitian ini juga menguji ketangguhan Windows Defender dalam mendeteksi sampel malware yang telah mengalami proses obfuscation. Hasil pengujian menunjukkan bahwa teknik penyamaran masih efektif dalam menghindari mekanisme deteksi awal.

1. Static Bypass

Sampel malware yang menggunakan metode kompresi kustom terbukti mampu melewati pemindaian berbasis tanda tangan (*signature-based scan*) dengan tingkat keberhasilan mencapai 80%. Temuan ini mengindikasikan bahwa modifikasi struktur biner malware dapat secara signifikan menurunkan efektivitas deteksi statis.

2. Dynamic/Behavioral Detection

Mekanisme deteksi berbasis perilaku mulai memberikan respons ketika malware melakukan serangkaian aktivitas mencurigakan secara berurutan, seperti modifikasi *registry* yang diikuti oleh pembentukan koneksi jaringan. Namun, dalam sebagian besar kasus, peringatan keamanan muncul setelah malware berhasil membangun mekanisme persistensi. Kondisi ini menunjukkan bahwa sistem telah berada dalam keadaan terkompromi (*compromised*) sebelum respons pertahanan diaktifkan.

Received: November 8, 2025 – Revised: November 23, 2025 - Accepted: Desember 11, 2025 - Published online: January 7, 2026

Kelemahan Struktural dan Rekomendasi Strategis

Kegagalan sistem dalam menahan serangan malware lokal tidak disebabkan oleh satu faktor tunggal, melainkan merupakan kombinasi antara kerentanan teknis dan kesalahan konfigurasi. Beberapa faktor yang berperan dalam hal ini antara lain:

1. Trust-Based Vulnerability

Sistem operasi secara default memberikan tingkat kepercayaan tinggi terhadap proses-proses yang ditandatangani secara digital oleh Microsoft. Mekanisme ini justru dimanfaatkan oleh malware untuk menyembunyikan aktivitasnya, sehingga mempersulit deteksi oleh sistem keamanan.

2. UAC Over-Reliance

Pengguna sering kali mengabaikan peringatan dari User Account Control (UAC), misalnya dengan mengklik *"Yes"* tanpa membaca pesan peringatan secara seksama. Kebiasaan ini menciptakan titik lemah psikologis yang dapat dieksploitasi oleh malware untuk memperoleh hak akses yang lebih tinggi (*privilege escalation*).

Kesimpulan

Berdasarkan serangkaian pengujian dinamis dan analisis teknis yang telah dilakukan, dapat disimpulkan bahwa keamanan sistem operasi modern terhadap ancaman malware lokal sangat bergantung pada konfigurasi hak akses dan perilaku pengguna. Meskipun arsitektur sistem telah dilengkapi dengan mekanisme perlindungan memori tingkat tinggi, seperti ASLR (Address Space Layout Randomization) dan DEP (Data Execution Prevention), malware lokal tetap mampu mengkompromikan sistem dengan memanfaatkan teknik Process Hollowing, yang memungkinkan kode berbahaya dijalankan melalui proses legal tanpa terdeteksi oleh pemindaian berbasis tanda tangan.

Hasil pengujian menunjukkan bahwa hak akses administrator merupakan faktor kunci dalam keberhasilan infeksi sistemik. Sistem yang dijalankan dengan hak akses penuh memungkinkan malware melumpuhkan pertahanan internal, memodifikasi registry secara permanen, serta mengganggu jalur pembaruan keamanan. Dengan demikian, meskipun mekanisme pertahanan teknis telah diterapkan, celah signifikan tetap ada, terutama yang berkaitan dengan manipulasi psikologis pengguna dan teknik persistensi yang memanfaatkan fungsi sah dari sistem operasi. Temuan ini menegaskan pentingnya pendekatan keamanan berlapis, yang mencakup konfigurasi hak akses yang tepat, edukasi pengguna, serta peningkatan deteksi berbasis perilaku.

Saran

Sebagai langkah mitigasi untuk memperkuat pertahanan sistem di masa mendatang, sangat disarankan bagi pengguna untuk menerapkan prinsip Least Privilege dengan menggunakan akun standar dalam aktivitas operasional sehari-hari, sehingga ruang gerak malware dapat dibatasi. Pihak administrator sistem perlu mengimplementasikan kebijakan pembatasan aplikasi yang lebih ketat, seperti penggunaan AppLocker atau Application Control, untuk memverifikasi keabsahan setiap file eksekusi sebelum diizinkan berjalan di lingkungan sistem. Selain itu, bagi pengembang perangkat lunak keamanan dan peneliti selanjutnya, diperlukan pengembangan metode deteksi berbasis perilaku (heuristic detection) yang lebih proaktif, serta penerapan pemantauan integritas registry secara real-time untuk mengantisipasi tren fileless malware. Edukasi mengenai keamanan siber tetap menjadi

Received: November 8, 2025 – Revised: November 23, 2025 - Accepted: Desember 11, 2025 - Published online: January 7, 2026

komponen penting yang tidak terpisahkan dari solusi teknis, guna meminimalisir risiko serangan berbasis social engineering yang sering menjadi pintu masuk utama malware lokal.

Daftar Pustaka

Abadi, M., & Kurniawan, D. (2022). Analisis Keamanan Sistem Operasi terhadap Serangan Malware pada Lingkungan Virtual. Jurnal Teknologi Informasi dan Ilmu Komputer, 9(3), 455-462.

Microsoft Corporation. (2023). Windows Security Internals: Understanding Ring Protection and User Account Control. Microsoft Press.

Pratama, A. R., & Sari, I. P. (2021). Efektivitas Teknik Process Hollowing dalam EvasionAntivirus pada Sistem Operasi Windows 10. Jurnal Forensik Digital Indonesia, 4(1), 12-25.

Rusli, M. (2023). Evolusi Malware Lokal: Teknik Persistensi dan Metode Penyebaran melalui Media Penyimpanan Eksternal. Jakarta: Penerbit Informatika.

Sikorski, M., & Honig, A. (2012). Practical Malware Analysis: The Hands-On Guide to Dissecting Malicious Software. No Starch Press.

Sysinternals. (2024). Process Monitor and Process Explorer: Official Documentation. Microsoft Learn. [Online]

Yusuf, M. (2022). Implementasi Metode Sandboxing untuk Analisis Dinamis Malware Lokal. Jurnal Sistem Informasi dan Keamanan Siber, 5(2), 88-101.

Received: November 8, 2025 – Revised: November 23, 2025 - Accepted: Desember 11, 2025 - Published online: January 7, 2026