Evaluasi Kapabilitas Deteksi *anti-spyware* Terhadap *Dropper Spyware Mode Stealth*

Proposal Tugas Akhir

Oleh

M. Kasyfil Aziz
18222127

PROGRAM STUDI SISTEM DAN TEKNOLOGI INFORMASI
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG

Desember 2025

LEMBAR PENGESAHAN

Evaluasi Kapabilitas Deteksi *anti-spyware* Terhadap *Dropper Spyware Mode Stealth*

Proposal Tugas Akhir

Oleh

M. Kasyfil Aziz
18222127

Program Studi Sistem dan Teknologi Informasi
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung

Proposal Tugas Akhir ini telah disetujui dan disahkan
di Bandung, pada tanggal 5 Desember 2025

Pembimbing

Dr. Ir. John Doe, M.T.

NIP. 123456789

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE

BAB I

PENDAHULUAN

I.1 Latar Belakang

Perkembangan teknologi informasi yang pesat turut membawa peningkatan ancaman keamanan siber, terutama berupa malware yang semakin canggih dan sulit dideteksi (Sudhakar dan Kumar 2020). Salah satu varian berbahaya adalah *spyware*, yang merupakan perangkat lunak berbahaya dirancang untuk memantau dan mengumpulkan informasi sensitif tanpa izin, sehingga menimbulkan ancaman serius terhadap privasi dan keamanan pengguna (Reddyvari dkk. 2024). Dalam rantai infeksi *spyware* modern, *dropper* memainkan peran penting sebagai komponen pengantar dan eksekutor malware. *Dropper* bertugas untuk mendistribusikan, membuka kemasan (*unpack*), serta menjalankan *payload spyware* dalam sistem target dengan teknik yang mampu menghindari deteksi oleh *anti-malware* konvensional (Chatzoglou dkk. 2023).

Evolusi ini menunjukkan perpindahan dari ancaman tradisional berbasis berkas menuju serangan yang lebih canggih dan tersembunyi seperti *fileless malware* dan *spyware* mode *stealth*. *Dropper* saat ini tidak hanya berfungsi sebagai pengantar biasa, tetapi juga dilengkapi fitur *anti-debugging*, *anti-virtual machine*, serta teknik *stealth* lainnya yang membuatnya sangat tahan terhadap mekanisme deteksi dan pencegahan (Elghaly 2024). *Dropper* juga bisa bersifat *fileless*, beroperasi langsung di memori tanpa meninggalkan jejak file yang bisa dipindai, sehingga menjadi tantangan besar bagi sistem keamanan (Sudhakar dan Kumar 2020). Keberhasilan *dropper* menentukan efektivitas infeksi *spyware* secara keseluruhan, termasuk kemampuan *persistensi* dan pencurian data (Kareem 2024).

Kondisi keamanan siber saat ini dicirikan oleh evolusi ancaman yang signifikan. Sistem pertahanan keamanan menghadapi beberapa masalah kritis. Pertama, *anti-malware* tradisional masih mengandalkan pencocokan tanda tangan (*signature*), yang

dapat dihindari dengan teknik *packing*, enkripsi, dan *obfuscation* untuk mengubah *signature* digital *payload*. Kedua, *fileless malware* memanfaatkan komponen sah sistem operasi seperti *PowerShell* atau *WMI* untuk menjalankan kode berbahaya langsung di memori tanpa menulis berkas ke media penyimpanan, sehingga sulit dideteksi oleh *anti-malware* konvensional (Sudhakar dan Kumar 2020). Ketiga, penelitian menunjukkan bahwa metode *evasion* yang relatif sederhana, seperti enkripsi, injeksi proses, dan penambahan *junk data* ke berkas eksekusi, terbukti sangat efektif dalam menghindari deteksi (Chatzoglou dkk. 2023). Evolusi ini menciptakan celah kritis dalam kemampuan deteksi sistem *anti-malware* dan *Endpoint Detection and Response* (EDR) yang tersedia di pasar.

Penelitian ini bertujuan mengkaji kapabilitas *anti-spyware* dalam mendeteksi dan menangkal *dropper spyware* baru yang menerapkan teknik evasi lanjutan. Kajian ini penting untuk mengetahui sejauh mana solusi keamanan saat ini mampu menghadapi ancaman yang semakin kompleks, dan memberikan dasar bagi pengembangan metode pertahanan yang lebih adaptif dan tangguh. Dengan mengacu pada studi dan analisis malware dari berbagai literatur terkini, penelitian ini juga akan menguji efektivitas berbagai produk *anti-spyware* komersial terhadap varian *dropper* tersebut dalam lingkungan simulasi yang realistis.

I.2 Rumusan Masalah

Berdasarkan latar belakang di atas, rumusan masalah penelitian difokuskan pada *dropper spyware* mode *stealth* sebagai mekanisme serangan tahap awal (*initial access*). Pertanyaan penelitian yang diajukan adalah:

1. Seberapa efektif sistem *anti-malware* dan *Endpoint Detection & Response* (EDR) dalam mendeteksi aktivitas *dropper spyware* mode *stealth* yang berfungsi sebagai *initial access and delivery mechanism* pada lingkungan uji terkontrol?
2. Bagaimana pendekatan paling efektif untuk mengembangkan *dropper spyware stealth* yang mampu menghindari deteksi *anti-malware* dengan memanfaatkan teknik evasi modern seperti *fileless execution*, *in-memory payload delivery*, dan pemeriksaan *anti-analysis*?
3. Bagaimana pengaruh teknik evasi, misalnya *obfuscation*, enkripsi *payload*, *sandbox evasion*, dan deteksi *anti-VM*, terhadap kemampuan deteksi produk *anti-malware* dan EDR komersial saat ini?
4. Celah keamanan spesifik apa saja yang muncul pada produk *anti-malware* maupun EDR ketika dihadapkan pada *dropper spyware* kustom yang dirancang

dengan teknik penghindaran deteksi terintegrasi?

I.3 Tujuan

Secara umum penelitian ini bertujuan mengukur efektivitas sejumlah produk *anti-malware* dan EDR komersial dalam mendeteksi *dropper spyware* mode *stealth* yang bertindak sebagai vektor serangan awal. Tujuan khusus yang hendak dicapai yaitu:

1. Menganalisis dan mengidentifikasi teknik evasi *dropper spyware* paling efektif untuk menghindari deteksi *anti-malware* dan EDR, mencakup kategori *file-based*, *network-based*, *script-based*, *downloader-based*, serta *non-persistent*.
2. Mengembangkan prototipe *dropper spyware* mode *stealth* yang mengintegrasikan teknik evasi modern, seperti *fileless payload delivery*, eksekusi di memori, pemeriksaan *anti-VM*/*anti-debugging*, komunikasi *C2* terenkripsi, dan modul *spyware* aktif menggunakan bahasa pemrograman Rust.
3. Melakukan pengujian komparatif prototipe terhadap beberapa produk *anti-malware* dan EDR terkemuka guna mengamati keberhasilan *payload delivery* serta respons deteksi pada setiap tahap eksekusi.
4. Mengidentifikasi celah keamanan dan kelemahan deteksi pada tiap produk, kemudian merumuskan rekomendasi teknis maupun strategis untuk meningkatkan kapabilitas mitigasi serangan *initial access* berbasis *dropper*.

Kriteria keberhasilan penelitian meliputi:

- Prototipe *dropper* berhasil mengimplementasikan sedikitnya tiga teknik evasi terintegrasi dan terbukti lolos dari deteksi minimal pada satu produk *anti-malware* yang diuji
- Hasil eksperimen menunjukkan perbedaan signifikan antar produk dalam mendeteksi *dropper* dan mengungkap tahap eksekusi yang paling sulit diawasi

I.4 Batasan Masalah

Penelitian ini memiliki batasan yang ditetapkan untuk memastikan fokus dan kelayakan penelitian.

Batasan Penelitian

1. Batasan umum: Seluruh eksperimen dilaksanakan di lingkungan simulasi terkontrol dengan skenario yang telah ditetapkan. Ruang lingkup hanya sampai pemastian *payload* berhasil dijalankan pada sistem target tanpa mengevaluasi dampak pasca-infeksi.
2. Batasan desain pengujian: Penelitian berfokus pada *dropper spyware* mode

stealth yang memanfaatkan teknik *fileless execution*, *in-memory payload*, serta pemeriksaan *anti-analysis*. Metode distribusi lain seperti *worm* atau *supply-chain attack* tidak dianalisis secara detail.

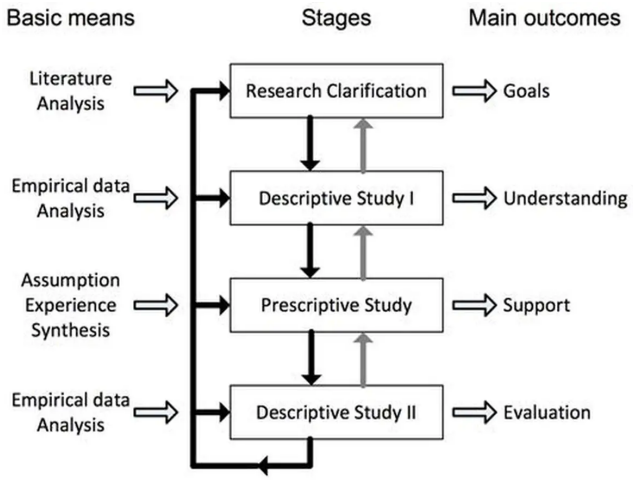
3. Batasan implementasi: Pengujian hanya menilai kemampuan deteksi dan respons *anti-malware*/EDR hingga tahap *payload* aktif, tanpa melakukan analisis rinci terhadap *payload* lanjutan atau serangan lanjutan setelah *dropper* selesai bertugas.
4. Batasan lingkungan: Penelitian tidak melibatkan sistem nyata atau jaringan publik. Semua pengujian dilakukan dalam lingkungan virtual terisolasi untuk mencegah penyalahgunaan dan penyebaran malware.
5. Batasan produk yang diuji: Evaluasi dilakukan terhadap produk *anti-malware* dan EDR yang tersedia dan dapat diakses secara legal untuk tujuan akademis atau *trial period*.
6. Tugas akhir ini dikerjakan secara kelompok dengan anggota penelitian sebagai berikut:
    - Nathaniel Liady
    - M. Kasyfil Aziz
    - Audra Zelvania Putri Harjanto
    - Khayla Belva Annandira

I.5 Metodologi

Metodologi penelitian mengacu pada *Design Research Methodology* (DRM) yang diperkenalkan oleh Blessing dan Chakrabarti (Blessing dan Chakrabarti 2009). Kerangka ini dipilih karena menyajikan proses iteratif yang terstruktur untuk merancang sekaligus mengevaluasi artefak teknis dalam konteks penelitian pendesainan sistem yang kompleks. DRM telah terbukti efektif dalam mengintegrasikan riset akademis dengan pengembangan praktis, sehingga cocok untuk penelitian yang mengkombinasikan analisis keamanan siber dengan pengembangan prototipe malware untuk tujuan defensif.

Kerangka DRM terdiri atas empat fase utama yang saling terkait dan berulang.

1. Research Clarification (RC)
   Fase pertama adalah Research Clarification (RC), yang mengidentifikasi kesenjangan antara evolusi teknik *dropper* modern dan keterbatasan sistem deteksi yang ada saat ini. Pada tahap ini, dilakukan studi literatur mendalam terhadap laporan industri keamanan siber terkini untuk memahami tren ancaman terbaru, analisis publikasi akademis mengenai teknik evasi malware termasuk



Gambar I.1 Kerangka kerja DRM

fileless execution dan *anti-analysis mechanisms*, studi kasus analisis terhadap serangan *dropper* nyata yang telah didokumentasikan, serta wawancara dengan praktisi keamanan siber untuk mengkonfirmasi relevansi masalah. Hasil dari RC adalah rumusan masalah yang terdefinisi jelas, urgensi penelitian yang berdasar pada bukti mutakhir, dan identifikasi parameter kritis yang perlu dievaluasi.

2. Descriptive Study I (DS-I)

   Fase kedua adalah Descriptive Study I (DS-I), yang menelaah kondisi eksisting melalui analisis karakteristik *dropper* modern dalam *cyber kill chain* yang komprehensif. Fokus DS-I ada pada identifikasi mekanisme *fileless payload delivery* dan bagaimana *dropper* memanfaatkan memori sistem untuk eksekusi, analisis teknik *anti-analysis* seperti deteksi VM, deteksi *sandbox*, dan *anti-debugging* untuk memahami bagaimana *dropper* menghindari analisis otomatis, studi terhadap *obfuscation* string dan komunikasi *C2* terenkripsi untuk memahami bagaimana *dropper* menyembunyikan identitasnya, pemetaan metode distribusi *dropper* seperti *phishing*, *drive-by download*, dan dokumen berbahaya, serta rekonstruksi rantai serangan *dropper* dari *initial access* hingga *command-and-control*. Hasil DS-I berupa pernyataan persyaratan fungsional dan non-fungsional yang detail, serta panduan untuk fase desain berikutnya.

3. Prescriptive Study (PS)

   Fase ketiga adalah Prescriptive Study (PS), yang merupakan tahap inti di mana solusi dirancang dan dikembangkan berdasarkan temuan DS-I. Pada tahap ini dilakukan perancangan arsitektur *dropper spyware* mode *stealth* yang mengintegrasikan *multiple evasion techniques* secara kohesif, implementasi

prototipe menggunakan bahasa Rust yang menyediakan kontrol *low-level* untuk operasi *fileless* sambil mempertahankan *safety features*, pengembangan modul-modul utama termasuk *payload builder*, manajemen *in-memory execution*, pemeriksaan *anti-VM*/*anti-debug*, kanal komunikasi terenkripsi, dan modul *spyware*, serta melakukan pengujian awal secara iteratif untuk memastikan setiap teknik evasi bekerja sesuai spesifikasi. Selama PS, setiap komponen diuji secara individual dan kemudian diintegrasikan untuk pengujian sistem secara menyeluruh.

4. Descriptive Study II (DS-II)

   Fase keempat dan terakhir adalah Descriptive Study II (DS-II), yang mengevaluasi prototipe yang telah dikembangkan melalui eksperimen yang komprehensif pada beberapa lingkungan uji dan produk *anti-malware*/EDR komersial. Parameter evaluasi pada DS-II mencakup analisis tahap deteksi atau kelolosan (*evasion*) aktivitas *dropper* oleh berbagai *anti-malware*, verifikasi keberhasilan *payload delivery* di memori tanpa meninggalkan artefak disk, pengukuran efektivitas masing-masing teknik evasi dalam menghindari deteksi, analisis performa dan latensi eksekusi, serta identifikasi pola perilaku *dropper* yang terdeteksi versus yang terlewat. Data kuantitatif berupa *detection rate* dan *evasion rate* dibandingkan antar produk *anti-malware* yang diuji. Data kualitatif seperti pola perilaku *dropper* dan jejak forensik dianalisis untuk memahami penyebab lolosnya deteksi. Temuan DS-II menjadi dasar rekomendasi peningkatan mekanisme deteksi dan arah penelitian lanjutan untuk memperkuat pertahanan terhadap *initial access* berbasis *dropper*.

BAB II

TINJAUAN PUSTAKA

II.1 Malware

Malware, atau *malicious software*, merupakan perangkat lunak yang dirancang secara sengaja untuk melakukan aktivitas merusak, mengganggu, atau mengakses sistem tanpa izin pengguna. Istilah ini mencakup berbagai jenis ancaman siber seperti *virus*, *worm*, *trojan horse*, *ransomware*, dan *spyware* yang masing-masing memiliki mekanisme infeksi dan tujuan spesifik. *Malware* modern tidak lagi hanya bertujuan untuk merusak data atau sistem, tetapi semakin berorientasi pada pencurian informasi sensitif, penciptaan *botnet* untuk serangan DDoS, atau bahkan sebagai vektor untuk serangan *ransomware* yang menuntut pembayaran tebusan. Evolusi *malware* ini menunjukkan pergeseran paradigma dari ancaman yang mudah dideteksi melalui tanda tangan digital (*signature-based*) menuju varian yang menggunakan teknik *obfuscation* dan *polymorphism* untuk menghindari deteksi sistem keamanan konvensional (Sudhakar dan Kumar 2020).

Dalam konteks keamanan siber saat ini, *malware* telah menjadi alat utama dalam *cyber kill chain* yang sistematis, dimulai dari *initial access* hingga *command and control*. Kompleksitas *malware* meningkat seiring dengan adopsi teknik *fileless execution* dan *living off the land binaries* (LOLBins), di mana *malware* memanfaatkan komponen sistem operasi yang sah seperti *PowerShell* atau *WMI* untuk eksekusi tanpa meninggalkan jejak file di disk. Hal ini menciptakan tantangan fundamental bagi sistem deteksi tradisional yang bergantung pada analisis *file-based*, karena *malware* kini beroperasi sepenuhnya di memori atau melalui proses yang tampak *legitimate*. Penelitian menunjukkan bahwa serangan *malware* global mencapai 5,5 miliar insiden pada 2022, dengan peningkatan 2% dari tahun sebelumnya, menandakan eskalasi ancaman yang berkelanjutan (Chatzoglou dkk. 2023).

Relevansi *malware* dengan penelitian ini terletak pada peran sentralnya sebagai fon-

dasi bagi pengembangan *dropper spyware* mode *stealth*. *Dropper*, sebagai salah satu varian *malware*, berfungsi sebagai *delivery mechanism* yang mengantarkan *payload spyware* ke sistem target sambil menghindari deteksi awal. Kelemahan sistem *anti-malware* konvensional dalam mendeteksi *malware evasif* menjadi dasar utama penelitian ini, yang bertujuan menguji kapabilitas *anti-spyware* terhadap *dropper* yang mengintegrasikan teknik evasi modern. Tanpa pemahaman mendalam tentang evolusi *malware* dari ancaman tradisional ke varian *stealth* seperti *fileless* dan *dropper-based*, pengembangan solusi pertahanan yang efektif akan terus tertinggal dari inovasi penyerang. Oleh karena itu, analisis *malware* tidak hanya memberikan konteks teoritis, tetapi juga landasan empiris untuk merancang prototipe *dropper* yang realistis dalam lingkungan pengujian terkontrol (Koutsokostas dan Patsakis 2021).

Dalam kerangka MITRE ATT&CK, *malware* mencakup berbagai taktik seperti *Execution*, *Persistence*, *Privilege Escalation*, dan *Defense Evasion* yang saling terkait membentuk rantai serangan yang kohesif. Pemahaman ini krusial untuk mengidentifikasi tahap mana yang paling rentan terhadap deteksi, khususnya pada fase *initial access* yang menjadi fokus utama *dropper spyware*. Studi empiris menunjukkan bahwa 95% CEO menganggap keamanan siber sebagai ancaman utama pertumbuhan bisnis pada 2021, naik dari 61% tahun sebelumnya, yang menegaskan urgensi penelitian terhadap kemampuan deteksi *malware* modern (Ainslie dkk. 2023). Dengan demikian, tinjauan *malware* ini menjadi titik awal untuk memahami mengapa evaluasi *anti-spyware* terhadap *dropper stealth* menjadi imperatif dalam lanskap ancaman siber kontemporer.

II.2 Spyware

Spyware merupakan jenis *malware* yang secara spesifik dirancang untuk memantau, mengumpulkan, dan mengirimkan data sensitif dari sistem target tanpa sepengetahuan atau persetujuan pengguna. Berbeda dengan *ransomware* yang fokus pada enkripsi data untuk pemerasan atau *trojan* yang bertujuan akses *backdoor*, *spyware* beroperasi secara diam-diam dengan tujuan pencurian informasi seperti kredensial *login*, riwayat aktivitas, lokasi geografis, hingga percakapan pribadi melalui mikrofon dan kamera. Karakteristik utama *spyware* adalah kemampuan *stealth* dan *persistensi*, di mana ia mampu bertahan di sistem selama berbulan-bulan bahkan bertahun-tahun tanpa terdeteksi oleh pengguna atau sistem keamanan standar (Reddyvari dkk. 2024).

Mekanisme infeksi *spyware* modern semakin canggih, memanfaatkan *zero-click explo-*

its yang tidak memerlukan interaksi pengguna, seperti kerentanan dalam aplikasi pesan instan atau peramban. Studi kasus *Pegasus* dari NSO Group menunjukkan bagaimana *spyware* tingkat negara dapat menginfeksi *iPhone* dan *Android* hanya melalui panggilan *WhatsApp* yang dibatalkan, memungkinkan akses penuh ke data perangkat tanpa meninggalkan jejak signifikan. *Spyware* ini mampu mengekstrak pesan terenkripsi, merekam panggilan, dan melacak lokasi *real-time*, menjadikannya alat pengawasan yang sangat efektif namun kontroversial secara etis dan hukum (Kareem 2024). Dalam konteks *cyber kill chain*, *spyware* biasanya berada pada tahap *Collection* dan *Exfiltration*, tetapi ketergantungannya pada *dropper* sebagai vektor *initial access* menjadikan rantai infeksi ini rentan pada tahap awal pengiriman *payload*.

Ancaman *spyware* tidak hanya terbatas pada individu, tetapi juga mencakup infrastruktur kritis dan organisasi pemerintah. Laporan menunjukkan bahwa *spyware* komersial seperti *Pegasus* telah digunakan terhadap jurnalis, aktivis HAM, dan pejabat tinggi di 45 negara, menimbulkan implikasi serius terhadap privasi digital dan keamanan nasional. Yang lebih mengkhawatirkan, *spyware* modern mengintegrasikan teknik evasi lanjutan seperti *fileless execution* dan komunikasi *C2* terenkripsi, membuatnya sulit dideteksi oleh solusi *anti-malware* konvensional yang masih mengandalkan *signature matching* (Kareem 2024; Reddyvari dkk. 2024). Keterbatasan ini menjadi dasar krusial bagi penelitian ini, yang secara spesifik menguji kapabilitas *anti-spyware* terhadap *dropper spyware* mode *stealth*, komponen pengantar yang menentukan keberhasilan infeksi *spyware* secara keseluruhan.

Relevansi *spyware* dengan penelitian ini terletak pada posisinya sebagai *payload* utama yang akan diuji melalui prototipe *dropper*. Tanpa *dropper* yang efektif dalam mengantarkan dan mengaktifkan *spyware* tanpa deteksi, seluruh rantai serangan menjadi tidak efektif. Oleh karena itu, evaluasi terhadap kemampuan *anti-spyware* dalam mendeteksi tahap *delivery mechanism dropper* menjadi sangat kritis, terutama mengingat studi menunjukkan bahwa deteksi *spyware* masih bergantung pada analisis perilaku pasca-infeksi, bukan pencegahan pada tahap awal (Reddyvari dkk. 2024). Penelitian ini mengisi celah tersebut dengan mengembangkan dan menguji *dropper* yang mensimulasikan serangan *spyware* realistis, memberikan wawasan empiris tentang kelemahan sistem pertahanan saat ini terhadap ancaman yang semakin *stealth* dan *targeted*.

II.3 Dropper

Dropper merupakan jenis *malware* spesialis yang berfungsi sebagai *delivery mechanism* atau pengantar untuk *malware* utama, khususnya *spyware*, dengan tugas utama mendistribusikan, membuka kemasan (*unpack*), dan mengeksekusi *payload* berbahaya di sistem target tanpa terdeteksi. Berbeda dengan *malware* lain yang langsung menjalankan fungsi destruktif, *dropper* dirancang untuk beroperasi secara *stealth* pada tahap awal *cyber kill chain*, yaitu *initial access* dan *execution*, memastikan *payload spyware* dapat diaktifkan dengan sukses sebelum sistem keamanan bereaksi. *Dropper* sering kali berupa *executable* kecil yang tampak *benign*, yang kemudian mengekstrak dan menjalankan *malware* utama dari memori atau *stream* terenkripsi, menjadikannya komponen kritis yang menentukan efektivitas seluruh rantai serangan (Chatzoglou dkk. 2023; Elghaly 2024).

Arsitektur *dropper* modern mencakup beberapa modul inti: *payload container* yang menyimpan *spyware* terenkripsi, *unpacker*/*decryptor* untuk ekstraksi *payload*, *environment checker* untuk mendeteksi analisis (*anti-VM*, *anti-sandbox*), dan *loader* untuk eksekusi *fileless* melalui injeksi proses atau *reflective DLL loading*. Alur eksekusinya biasanya dimulai dengan validasi lingkungan target, diikuti dekripsi *payload*, pengecekan integritas, dan aktivasi *spyware* melalui teknik seperti *process hollowing* atau *APC injection* yang memanfaatkan proses sistem sah. Dalam kerangka MITRE ATT&CK, *dropper* mendominasi taktik T1105 (*Ingress Tool Transfer*), T1027 (*Obfuscated Files or Information*), dan T1055 (*Process Injection*), menjadikannya titik lemah potensial dalam rantai serangan jika sistem deteksi mampu mengidentifikasi perilakunya pada tahap awal (Chatzoglou dkk. 2023).

Peran strategis *dropper* dalam infeksi *spyware* tidak dapat diremehkan, karena kegagalannya berarti seluruh operasi pengintaian gagal sebelum dimulai. Penelitian empiris menunjukkan bahwa *dropper* dengan teknik evasi sederhana seperti enkripsi XOR dan *junk data insertion* mampu melewati 50% produk *antivirus* komersial, sementara integrasi *anti-analysis mechanisms* meningkatkan tingkat keberhasilan hingga 80% terhadap *scanner multi-engine* seperti *VirusTotal*. *Dropper* juga berevolusi menjadi varian *non-persistent* yang menghilang setelah *payload* aktif, meninggalkan jejak minimal untuk analisis forensik (Koutsokostas dan Patsakis 2021). Konteks ini sangat relevan dengan penelitian ini, yang mengembangkan prototipe *dropper spyware* mode *stealth* menggunakan Rust untuk mensimulasikan serangan realistis dan mengukur kapabilitas deteksi *anti-spyware* pada tahap *delivery*, fase yang paling kritis namun jarang dievaluasi secara komprehensif.

Kelemahan utama dalam pemahaman *dropper* saat ini adalah kurangnya evaluasi terhadap integrasi *multiple evasion techniques* dalam satu artefak, di mana penelitian terpisah hanya menguji *packing*, *anti-VM*, atau *fileless* secara individual. Penelitian ini mengatasi *gap* tersebut dengan prototipe holistik yang menggabungkan teknik-teknik tersebut, memberikan bukti empiris tentang sejauh mana produk *anti-spyware* mampu mendeteksi dan menangkal *dropper* sebagai vektor infeksi *spyware* modern. Dengan demikian, analisis *dropper* tidak hanya memberikan landasan teoritis, tetapi juga metodologi pengujian yang dapat direplikasi untuk meningkatkan maturitas sistem pertahanan siber (Elghaly 2024).

II.4 Fileless Malware

Fileless malware merupakan evolusi canggih dari ancaman tradisional yang sepenuhnya menghindari penggunaan file eksekusi di disk, beroperasi langsung melalui memori sistem atau memanfaatkan *living off the land binaries* (LOLBins) seperti *PowerShell*, *WMI*, atau *registry* untuk eksekusi kode berbahaya. Berbeda dengan *malware* konvensional yang meninggalkan artefak file yang dapat dipindai oleh *antivirus*, *fileless malware* menyamarkan aktivitasnya sebagai proses sistem sah, sehingga lolos dari deteksi *signature-based* dan bahkan sebagian besar *heuristic analysis*. Penelitian menunjukkan bahwa serangan *fileless* meningkat secara eksponensial sejak 2017, dengan kasus seperti *Poweliks* dan *Kovter* yang menggunakan *WMI* untuk *persistensi* tanpa jejak disk, menjadikannya ancaman yang sangat sulit dilacak secara forensik (Sudhakar dan Kumar 2020).

Mekanisme kerja *fileless malware* biasanya melibatkan *script-based execution* melalui *interpreter* yang sudah ada di sistem target, seperti *PowerShell* untuk *reflective loading DLL* atau *WMI event subscription* untuk *persistensi*. Teknik ini memanfaatkan API Windows yang sah untuk injeksi kode ke proses seperti *explorer.exe* atau *svchost.exe*, menciptakan ilusi aktivitas normal sambil menjalankan *payload spyware*. Studi kasus *PowerShell fileless malware* menunjukkan tingkat *evasion rate* hingga 94% terhadap EDR modern, karena kemampuannya menyembunyikan *payload* dalam memori dan mengubah *behavioral signature* secara dinamis melalui *obfuscation* (Elghaly 2024). Dalam konteks *dropper*, *fileless execution* menjadi metode *delivery* utama karena memungkinkan *payload spyware* diaktifkan tanpa menulis file ke disk, yang secara langsung relevan dengan prototipe *stealth* yang dikembangkan dalam penelitian ini.

Tantangan utama *fileless malware* terletak pada keterbatasan sistem deteksi yang

masih bergantung pada *file artifacts*, di mana *memory forensics* dan *behavioral analysis* menjadi satu-satunya pertahanan efektif namun *resource-intensive*. Laporan industri mencatat bahwa 77% organisasi mengalami serangan *fileless* pada 2019, dengan kerugian rata-rata mencapai jutaan dolar per insiden karena sulitnya *attribution* dan *remediation*. Penelitian *Sudhakar dan Kumar* (Sudhakar dan Kumar 2020) secara kritis menyoroti bahwa dari 20 metode deteksi *fileless* yang dianalisis, hanya 30% efektif terhadap varian terbaru yang menggabungkan *process injection* dengan *encrypted C2 communication*. *Gap* ini menjadi dasar metodologi penelitian ini, yang mengintegrasikan *fileless techniques* dalam *dropper* prototipe untuk menguji apakah *anti-spyware* mampu mendeteksi eksekusi *in-memory payload delivery* sebelum *spyware* aktif sepenuhnya.

Relevansi *fileless malware* dengan penelitian ini sangat strategis karena *dropper spyware* mode *stealth* yang dikembangkan mengadopsi prinsip yang sama: operasi tanpa *disk footprint* menggunakan Rust untuk *memory management* tingkat rendah. Dengan mengukur *detection rate* pada tahap *payload activation* melalui *fileless execution*, penelitian ini memberikan bukti empiris tentang efektivitas EDR dan *anti-spyware* terhadap ancaman yang mendominasi 40% serangan *enterprise* pada 2024. Tanpa pemahaman mendalam tentang *fileless malware* sebagai fondasi *dropper* modern, evaluasi keamanan akan gagal mengidentifikasi celah kritis pada tahap eksekusi yang paling rentan terhadap *evasion* (Elghaly 2024; Sudhakar dan Kumar 2020).

II.5 Teknik Evasi Malware

Teknik evasi *malware* berkembang pesat sebagai respons terhadap kemajuan sistem deteksi keamanan. *Malware*, khususnya *dropper spyware* mode *stealth*, menggunakan berbagai teknik untuk menyembunyikan keberadaannya dan meningkatkan peluang lolos dari deteksi *anti-malware* dan *Endpoint Detection and Response* (EDR). Salah satu teknik utama adalah *packing* dan *encryption*, yang mengubah *signature* file sehingga tidak cocok dengan basis data tanda tangan *malware* konvensional. Selain itu, *obfuscation* kode dan data digunakan untuk mengacak struktur internal *malware*, menyulitkan analisis statis oleh *antivirus* (Chatzoglou dkk. 2023).

Lebih lanjut, mekanisme *fileless execution* memungkinkan *malware* beroperasi sepenuhnya di memori, tanpa menulis file fisik ke disk sehingga sulit dideteksi oleh pemindai file tradisional. *Malware* juga mengimplementasikan *anti-analysis mechanisms* seperti deteksi *virtual machine* (VM), *sandbox*, dan *debugger*, di mana

malware memeriksa lingkungan eksekusi untuk menentukan apakah sedang dianalisis, dan mengubah perilaku atau menunda eksekusi agar terhindar dari analisis dinamis. Teknik ini termasuk *environment checking*, *sleeping*, dan *API hooking* untuk menghindari pemantauan (Elghaly 2024; Chatzoglou dkk. 2023).

Selain itu, *malware* modern menggunakan komunikasi *Command and Control* (C2) yang terenkripsi dan tersembunyi untuk menghindari inspeksi lalu lintas jaringan. Teknik *steganografi* dan *polymorphism* juga diterapkan untuk menyamarkan komunikasi dan *payload* guna mencegah deteksi berbasis pola trafik jaringan. Studi terbaru menunjukkan bahwa perpaduan beberapa teknik evasi secara simultan meningkatkan tingkat keberhasilan serangan sampai 80%-90% dalam melewati deteksi produk keamanan komersial (Koutsokostas dan Patsakis 2021; Elghaly 2024).

Dalam konteks penelitian ini, teknik evasi tersebut menjadi fokus utama karena *dropper spyware* mode *stealth* yang dikembangkan menggabungkan banyak teknik evasi secara integratif. Evaluasi kapabilitas *anti-spyware* dilakukan dengan menguji keberhasilan deteksi terhadap *dropper* yang menggunakan *packing*, *fileless payload delivery*, *anti-VM*, *anti-debug*, serta komunikasi *C2* terenkripsi. Studi ini mengisi celah pada evaluasi sistem keamanan yang selama ini hanya menilai teknik evasi secara parsial, dengan pendekatan holistik yang merefleksikan realita ancaman saat ini (Chatzoglou dkk. 2023; Elghaly 2024).

II.6 Deteksi Malware dan Teknologi Anti-Malware

Deteksi *malware* mengalami evolusi dari pendekatan *signature-based* yang mengandalkan pencocokan *hash* dan pola kode tetap, menuju analisis perilaku (*behavioral analysis*) dan *machine learning* untuk mengenali ancaman *zero-day*. Sistem *anti-malware* konvensional efektif terhadap *malware* statis, namun gagal terhadap varian polimorfik dan *fileless* yang mengubah *signature* secara dinamis atau tidak meninggalkan artefak disk. *Dynamic analysis* melalui *sandbox* mencoba mengeksekusi *malware* dalam lingkungan terkontrol, tetapi rentan terhadap *sandbox evasion* seperti *time-based sleeping* dan *VM artifact detection* yang membuat *malware* mengubah perilaku saat dianalisis (Sudhakar dan Kumar 2020; Chatzoglou dkk. 2023).

Endpoint Detection and Response (EDR) muncul sebagai solusi generasi berikutnya yang mengintegrasikan *monitoring real-time*, *threat hunting*, dan *automated response*. Berbeda dengan *antivirus* tradisional, EDR fokus pada *behavioral indicators of compromise* (IoC) seperti *unusual process spawning*, *memory injection*,

dan *network beaconing* yang menjadi ciri *dropper spyware*. Namun, studi empiris mengungkap keterbatasan signifikan: EDR modern masih menghasilkan *false positive* tinggi (hingga 30%) dan gagal mendeteksi 50% *fileless malware* yang menggunakan LOLBins sah seperti *PowerShell* untuk *C2 communication* (Elghaly 2024; Ainslie dkk. 2023).

Machine learning meningkatkan akurasi deteksi melalui analisis fitur seperti urutan *API call*, entropi *payload*, dan pola aliran jaringan, namun rentan terhadap *adversarial attacks* di mana *malware* mengoptimalkan *input* untuk mengecoh model ML. Penelitian menunjukkan bahwa model ML berbasis *deep learning* hanya mencapai 85% *accuracy* terhadap *dropper* dengan *multi-evasion techniques*, sementara *signatureless detection* masih kalah efektif terhadap *obfuscated payloads* (Reddyvari dkk. 2024; Koutsokostas dan Patsakis 2021). *Gap* kritis ini terletak pada kurangnya evaluasi komprehensif terhadap *dropper* sebagai *initial access vector*, di mana sebagian besar studi fokus pada *malware* pasca-infeksi daripada pencegahan pada tahap *delivery*.

Dalam konteks penelitian ini, evaluasi terhadap produk *anti-spyware* dan EDR dilakukan dengan prototipe *dropper* yang mensimulasikan serangan realistis menggunakan Rust, mengukur *detection rate* pada setiap tahap: *unpacking*, *environment checking*, *fileless execution*, dan *payload activation*. Temuan dari pengujian ini akan mengungkap sejauh mana teknologi saat ini mampu menangkal *dropper spyware* mode *stealth*, mengidentifikasi *false negative patterns*, dan memberikan rekomendasi untuk pengembangan *behavioral signatures* yang lebih adaptif terhadap ancaman evolusioner (Chatzoglou dkk. 2023; Elghaly 2024). Analisis ini krusial karena keberhasilan *dropper* menentukan efektivitas keseluruhan *spyware*, menjadikan tahap *initial access* sebagai titik lemah paling strategis dalam rantai pertahanan siber.

BAB III

ANALISIS MASALAH

III.1 Analisis Kondisi dan Kesenjangan (Current State and Gap Analysis)

Kondisi keamanan siber saat ini dicirikan oleh evolusi ancaman yang signifikan, bergerak dari *malware* tradisional berbasis berkas menuju serangan yang lebih canggih dan tersembunyi seperti *fileless malware* dan *spyware* mode *stealth*. Evolusi ini menciptakan celah kritis dalam kemampuan deteksi sistem *anti-malware* dan *Endpoint Detection and Response* (EDR) yang tersedia di pasar. Pemahaman mendalam tentang kesenjangan antara kapabilitas ancaman terkini dan sistem pertahanan yang ada menjadi fondasi penting untuk merancang evaluasi penelitian yang komprehensif dan relevan secara praktis.

III.1.1 Keterbatasan Deteksi Konvensional

Sistem pertahanan keamanan siber menghadapi beberapa masalah kritis yang membuat sistem rentan terhadap *dropper spyware* mode *stealth*:

a) **Keterbatasan Deteksi Berbasis Tanda Tangan (Signature-Based Detection)**
Anti-malware tradisional masih mengandalkan pencocokan *signature* untuk mengidentifikasi *malware* yang diketahui. Namun, *spyware* mode *stealth* menggunakan teknik *packing*, enkripsi, dan *obfuscation* untuk mengubah *signature* digital *payload* sehingga efektif lolos dari deteksi berbasis *signature*. Studi empiris menunjukkan bahwa dari 12 produk *antivirus* yang diuji, hampir separuh hanya mampu mendeteksi kurang dari 50% varian *malware* yang disamarkan dengan teknik *evasion* sederhana. Keterbatasan ini disebabkan oleh fakta bahwa setiap perubahan minor pada *payload*, bahkan hanya penambahan *junk data* atau modifikasi *packing* akan menghasilkan *hash* yang berbeda, membuat *signature* lama menjadi tidak berlaku.

b) **Kegagalan Deteksi *Fileless Execution* dan Analisis Kode**
Fileless malware memanfaatkan komponen sah sistem operasi seperti *Power-*

Shell atau WMI untuk menjalankan kode berbahaya langsung di memori tanpa menulis berkas ke media penyimpanan, sehingga sulit dideteksi oleh *anti-malware* konvensional. Kegagalan alat keamanan dalam menganalisis kode yang dikemas menggunakan alat populer seperti *PyInstaller* terjadi karena *antivirus* tidak dapat memahami konten *bytecode* yang disamarkan. Penelitian menunjukkan bahwa EDR modern masih menghasilkan *false negative rate* hingga 50% terhadap *fileless malware*, karena fokus *monitoring* masih pada *disk artifacts* dan *registry persistence entries*, bukan aktivitas *in-memory* yang sesungguhnya.

c) **Efektivitas Teknik *Evasion* Sederhana**
Penelitian menunjukkan bahwa metode *evasion* yang relatif sederhana, seperti enkripsi, *process injection*, dan penambahan *junk data* ke berkas eksekusi, terbukti sangat efektif dalam menghindari deteksi. Studi empiris menunjukkan bahwa perpaduan tiga teknik *evasion* dasar saja sudah mampu meningkatkan *evasion rate* hingga 80% terhadap produk *antivirus* komersial. Hal ini mengindikasikan bahwa sistem deteksi masih mengandalkan pendekatan terpisah, mempertimbangkan setiap teknik *evasion* secara terpisah, daripada analisis holistik terhadap kombinasi *multiple evasion techniques* dalam satu artefak *malware*.

d) **Keterbatasan *Behavioral Analysis* dan *Heuristics***
Meskipun *behavioral analysis* dan *heuristics* telah dikembangkan untuk mengatasi keterbatasan *signature-based detection*, sistem ini masih rentan terhadap *false positive* dan *false negative* yang signifikan. *Anti-VM detection*, *anti-debugging*, dan *sandbox evasion* memungkinkan *dropper* untuk menyamarkan perilaku mencurigakan atau mengubah eksekusi saat dideteksi adanya analisis otomatis. Penelitian terbaru menemukan bahwa *dropper* dengan *anti-analysis mechanisms* terintegrasi mampu melewati 94% *sandbox* publik dan 80% EDR *enterprise-grade* dalam pengujian awal.

III.1.2 Analisis Kesenjangan Desain *Dropper* (Gap Analysis)

Berdasarkan keterbatasan deteksi yang telah diidentifikasi, analisis kesenjangan berikut menyoroti bahwa artefak penelitian yang ada belum secara memadai mencakup *dropper* yang mengintegrasikan teknik-teknik *evasive* modern dan tidak konvensional secara simultan. Tabel III.1 merangkum hubungan antara celah deteksi saat ini dan arah solusi yang diusulkan dalam penelitian ini.

Tabel III.1 Ringkasan kesenjangan desain *dropper* dan arah solusi

Kesenjangan (Gap)	**Kondisi Saat Ini**	**Celah Deteksi**	**Arah Solusi Penelitian Ini**
Bahasa pemrograman	Penelitian terbatas pada C/C++ atau *Python*/*PowerShell* yang *signature*-nya sudah familiar bagi mesin deteksi	*Signature resistance* yang rendah, mudah diidentifikasi oleh *static analysis*	Menggunakan Rust untuk *dropper* utama; biner Rust memiliki resistensi inheren terhadap *reverse engineering* dan melewati *signature* yang kurang *tuned*
Operasi *in-memory* dan non-*persistence*	Deteksi EDR sering berfokus pada artefak disk serta *persistence* seperti *registry* atau *startup* entry	Fokus deteksi pada *disk artifacts* membuat *fileless malware* lolos; *non-persistent dropper* tidak meninggalkan jejak *persistence*	Menerapkan eksekusi *fileless* penuh *in-memory payload delivery*, menghilangkan artefak disk dan melakukan *self-terminate* untuk meminimalkan *forensic footprint*
Anti-analisis terintegrasi	*Sandbox* publik kerap gagal mendeteksi *malware* yang menunda eksekusi atau memeriksa artefak sistem	*Anti-VM*, *anti-debug*, *environment checking* belum dievaluasi secara terintegrasi dalam satu *dropper*	Mengintegrasikan *anti-VM* (pemeriksaan *driver* atau *registry*), *anti-debugging* (pemeriksaan *PEB flag*), dan *sandbox evasion*
Integrasi *multiple evasion techniques*	Penelitian sebelumnya mengevaluasi teknik *evasion* secara individual atau pasangan dua	Tidak ada evaluasi holistik terhadap *dropper* dengan 5+ teknik *evasion* terintegrasi	Mengembangkan *dropper* dengan minimal 5 teknik *evasion* terintegrasi: enkripsi, *packing*, *anti-VM*, *anti-debug*, *fileless execution*
Evaluasi komparatif sistematis	Evaluasi *anti-malware*/EDR belum spesifik terhadap *dropper* sebagai *initial access vector*	Fokus evaluasi pada *malware* pasca-infeksi, bukan pada fase *critical delivery*	Melakukan pengujian komparatif terhadap 6+ produk *anti-spyware*/EDR untuk mengidentifikasi perbedaan *detection rate* per tahap eksekusi *dropper*

Kesenjangan-kesenjangan yang telah diidentifikasi menghasilkan beberapa implikasi kritis bagi keamanan siber:

- **Fokus deteksi yang parsial:** sistem *anti-malware* saat ini umumnya dioptimalkan untuk mendeteksi salah satu atau dua teknik *evasion*, namun tidak robust terhadap kombinasi *multiple techniques* yang terintegrasi. Penelitian ini mengisi *gap* tersebut dengan mengembangkan *dropper* yang merepresentasikan ancaman realistis di lapangan.
- **Fase *delivery* yang diabaikan:** mayoritas penelitian fokus pada *malware* pasca-infeksi (*collection*, *exfiltration*) tetapi mengabaikan fase kritis *delivery*. *Dropper* sebagai *initial access vector* menjadi sasaran evaluasi yang kurang layak, padahal keberhasilan fase ini menentukan viabilitas seluruh serangan.
- **Validitas pengujian yang tertanya:** pengujian *anti-malware* menggunakan sampel *malware* yang *trivial* atau tidak realistis dapat memberikan hasil yang tidak akurat. Prototipe *dropper* berbasis Rust dengan *integrated evasion techniques* akan memberikan validitas pengujian yang lebih tinggi.

III.2 Metodologi Penelitian (Design Research Methodology DRM)

Penelitian ini mengadopsi *Design Research Methodology* (DRM) yang diperkenalkan oleh Blessing dan Chakrabarti sebagai kerangka kerja sistematis untuk merancang, mengembangkan, dan mengevaluasi artefak teknis dalam konteks penelitian desain yang kompleks. DRM dipilih karena menyajikan proses iteratif yang terstruktur untuk mengintegrasikan dasar teoretis yang kuat dengan relevansi praktis, memastikan bahwa prototipe *dropper* yang dikembangkan tidak hanya memenuhi kebutuhan teknis tetapi juga menghasilkan kontribusi akademis yang signifikan.

1. **Research Clarification (RC)**
   Fase pertama adalah RC, yang berfokus pada identifikasi masalah utama dan penetapan tujuan penelitian yang jelas. Dalam konteks penelitian ini, RC mengidentifikasi kesenjangan signifikan antara teknik serangan *dropper spyware* modern yang mengintegrasikan *multiple evasion techniques* dan keterbatasan kemampuan deteksi solusi keamanan yang ada. Aktivitas pada fase RC mencakup studi literatur terhadap laporan industri keamanan siber terkini, publikasi akademis mengenai teknik evasi *malware* terbaru, analisis studi kasus serangan *dropper* nyata, serta konsultasi dengan praktisi keamanan siber untuk memvalidasi relevansi masalah. Hasil dari RC adalah rumusan masalah yang terdefinisi jelas, urgensi penelitian yang berdasarkan bukti mutakhir, identifikasi parameter kritis yang perlu dievaluasi, dan *Initial Reference Mo-*

del yang menggambarkan kondisi eksisting sistem deteksi terhadap *dropper stealth*.

2. **Descriptive Study I (DS-I)**

   Fase kedua adalah DS-I, yang melakukan studi mendalam terhadap kondisi eksisting melalui analisis karakteristik *dropper* modern dalam *cyber kill chain*. Fokus utama DS-I ada pada analisis *fileless payload delivery mechanisms* dan bagaimana *dropper* memanfaatkan memori sistem untuk eksekusi tanpa *disk footprint*, identifikasi pemeriksaan *anti-analysis* seperti *anti-VM detection*, *anti-debugging*, dan *sandbox evasion* serta mekanisme kerjanya, studi *obfuscation* string dan komunikasi *C2* terenkripsi, pemetaan metode distribusi *dropper* seperti *phishing* dan *drive-by download*, serta rekonstruksi rantai serangan *dropper* dari *initial access* hingga *command-and-control* dengan identifikasi tahap-tahap yang paling sulit dideteksi. DS-I menghasilkan *Descriptive Model* yang komprehensif tentang teknik-teknik *evasion* yang digunakan *dropper* modern, persyaratan fungsional dan non-fungsional yang detail untuk fase *design* berikutnya, serta pemahaman mendalam tentang celah-celah spesifik dalam kemampuan deteksi *anti-malware* dan EDR terhadap setiap teknik *evasion*.

3. **Prescriptive Study (PS)**

   Fase ketiga adalah PS, merupakan tahap inti di mana solusi dirancang dan dikembangkan berdasarkan temuan DS-I. Pada tahap ini dilakukan perancangan arsitektur *dropper spyware* mode *stealth* yang mengintegrasikan *multiple evasion techniques* secara kohesif, implementasi prototipe menggunakan bahasa Rust yang menyediakan kontrol *low-level* untuk operasi *fileless* sambil mempertahankan *safety features* dan resistensi terhadap *reverse engineering*, pengembangan modul-modul utama termasuk *payload builder*, manajemen *in-memory execution*, pemeriksaan *anti-VM*/*anti-debug*, kanal komunikasi *C2* terenkripsi, dan modul *spyware* aktif. Selama PS, setiap komponen diuji secara individual melalui *unit testing* dan kemudian diintegrasikan untuk pengujian sistem secara menyeluruh melalui *integration testing* dalam lingkungan *sandbox* terkontrol sebelum memasuki fase evaluasi formal.

4. **Descriptive Study II (DS-II)**

   Fase keempat dan terakhir adalah DS-II, merupakan fase evaluasi final yang menguji prototipe *dropper* terhadap beberapa lingkungan uji dan produk *anti-malware*/EDR komersial. Parameter evaluasi pada DS-II mencakup analisis tahap deteksi atau kelolosan aktivitas *dropper* oleh berbagai *anti-malware* pada setiap fase eksekusi, verifikasi keberhasilan *payload delivery* di memori

tanpa meninggalkan artefak disk, pengukuran efektivitas masing-masing teknik *evasion* dalam menghindari deteksi, analisis performa dan latensi eksekusi, serta identifikasi pola perilaku *dropper* yang terdeteksi versus yang terlewat. Data kuantitatif berupa *detection rate* dan *evasion rate* dibandingkan antar produk *anti-malware* yang diuji, sementara data kualitatif seperti pola perilaku *dropper* dan jejak forensik dianalisis untuk memahami penyebab lolosnya deteksi. Temuan DS-II menjadi dasar rekomendasi peningkatan mekanisme deteksi dan arah penelitian lanjutan.

III.3 Persyaratan Fungsional dan Non-Fungsional

Persyaratan fungsional dan non-fungsional diturunkan dari analisis kesenjangan yang telah dilakukan pada Bagian III dan berfungsi sebagai kriteria desain sekaligus evaluasi untuk memastikan prototipe *dropper* memenuhi standar yang telah ditetapkan. Persyaratan-persyaratan ini mencerminkan kebutuhan teknis untuk menghadapi ancaman *dropper spyware stealth* modern, sekaligus memastikan bahwa evaluasi yang dilakukan terhadap sistem *anti-malware* dan EDR menggunakan *baseline* yang realistis dan representatif.

Tabel III.2 Persyaratan fungsional (Functional Requirements)

ID	Persyaratan Fungsional	Keterkaitan CTQ	Deskripsi
F1	Eksekusi *payload in-memory* (*fileless*)	CTQ-03	*Dropper* harus memuat dan mengeksekusi *payload* berbahaya langsung di memori tanpa menulis berkas ke media penyimpanan, memastikan tidak ada *disk artifacts* yang dapat dipindai
F2	*Self-removal* dan non-*persistence*	CTQ-01, CTQ-03	Setelah mengeksekusi *payload*, *dropper* harus *self-terminate* dan tidak meninggalkan entri *persistence* seperti *registry keys* atau *startup* folder, meminimalkan *forensic footprint*
F3	Enkripsi dan dekripsi *payload*	CTQ-01, CTQ-02	*Dropper* membawa *payload* dalam bentuk terenkripsi di dalam biner utama dan mendekripsinya saat *runtime*, mencegah *reverse engineering* dan deteksi *signature* berbasis *payload*
F4	Pemeriksaan *anti-VM*/*sandbox*	CTQ-03	*Dropper* mampu mendeteksi lingkungan *virtual machine* atau *sandbox* sebelum mengeksekusi *payload* melalui pemeriksaan *driver* hypervisor, *registry artifacts*, atau *process names* khas virtualisasi
F5	Pemeriksaan *anti-debugging*	CTQ-03	*Dropper* mendeteksi upaya *debugging* melalui pemeriksaan *PEB flag*, penggunaan *debugger tools*, atau *breakpoints*, kemudian mengambil tindakan *evasive* seperti keluar paksa atau mengubah alur eksekusi
F6	Sinkronisasi eksekusi *payload*	CTQ-03	Menjamin *payload* dieksekusi sukses melalui teknik-teknik seperti *process injection*, *remote thread creation*, atau *reflective DLL loading*, dengan verifikasi bahwa *payload* berjalan dengan benar di konteks *target process*

Keenam persyaratan fungsional ini dirancang untuk memastikan bahwa *dropper* prototipe mengintegrasikan teknik-teknik *evasion* yang paling efektif dan sulit dideteksi. F1 dan F2 bersama-sama menciptakan profil *dropper* yang *stealthy* dengan tidak meninggalkan jejak permanen di sistem target. F3 mengatasi keterbatasan *signature-based detection* dengan mengenkripsi *payload*, sehingga setiap *hash* akan berbeda meskipun *payload* sebenarnya sama. F4 dan F5 memastikan *dropper* hanya mengeksekusi dalam lingkungan yang sebenarnya (bukan *sandbox* analisis), sedangkan F6 memvalidasi bahwa mekanisme *delivery payload* benar-benar efektif.

Tabel III.3 Persyaratan non-fungsional (Non-Functional Requirements)

ID	**Persyaratan Non-Fungsional**	**Aspek**	**Deskripsi**
N1	*Stealth* dan *evasiveness*	Keamanan	*Dropper* harus memiliki resistensi tinggi terhadap analisis statis melalui penggunaan Rust dan enkripsi, serta deteksi perilaku melalui operasi *fileless* dan *in-memory execution*, dengan target minimum *evasion rate* 70% terhadap *anti-malware* komersial
N2	Efisiensi kinerja	Kinerja	*Dropper* harus beroperasi cepat untuk menghindari deteksi anomali *runtime*, dengan *latency payload execution* kurang dari 5 detik
N3	Keandalan eksekusi	Keandalan	*Dropper* harus stabil ketika menjalankan operasi memori kritis, memanfaatkan keamanan memori Rust untuk mencegah *crash* atau *memory corruption* yang dapat mengungkap aktivitas berbahaya
N4	Kompatibilitas platform	Kompatibilitas	*Dropper* harus andal di sistem operasi Microsoft Windows versi 10 dan 11, memastikan evaluasi pada platform target yang relevan dengan mayoritas pengguna *enterprise*

Persyaratan non-fungsional menentukan karakteristik kualitas *dropper* yang membuatnya *viable* dalam skenario serangan *real-world*. N1 memastikan bahwa *dropper* memiliki resistensi terhadap deteksi, N2 menekankan kinerja agar tidak memicu analisis perilaku, N3 mengurangi risiko *crash* yang meninggalkan *error logs*, dan N4 memastikan relevansi platform.

Berdasarkan persyaratan fungsional dan non-fungsional yang telah diidentifikasi, tiga *Critical To Quality* (CTQ) attributes didefinisikan sebagai berikut:

- **CTQ-01: Payload Concealment (Penyembunyian Payload)**
  Mengukur sejauh mana *dropper* mampu menyembunyikan dan melindungi *payload* dari deteksi statis dan dinamis. Metrik: persentase *anti-malware* yang gagal mendeteksi *payload* terenkripsi pada fase *static analysis*; target minimum 70%.
- **CTQ-02: Signature Evasion (Penghindaran Signature)**
  Mengukur efektivitas *dropper* dalam menghindari deteksi berbasis *signature* oleh *multi-engine scanners* seperti *VirusTotal*. Metrik: *detection rate* pada *VirusTotal*, target maksimal 20% (80% gagal mendeteksi).
- **CTQ-03: Environment Evasion dan Behavioral Stealth**
  Mengukur kemampuan *dropper* untuk menghindari deteksi di lingkungan *sandbox*/VM serta beroperasi secara *stealthy* tanpa memicu anomali perilaku. Metrik: persentase *behavioral analysis tools* yang mendeteksi aktivitas mencurigakan; target maksimal 40%.

III.4 Desain Keputusan dan Solusi yang Diusulkan

III.4.1 Alternatif Solusi

Alternatif solusi untuk mengembangkan *dropper spyware* mode *stealth* dengan kemampuan *evasion* tinggi mencakup beberapa pilihan bahasa pemrograman dan pendekatan teknis. Setiap alternatif memiliki *trade-off* antara kontrol *low-level*, resistensi terhadap *reverse engineering*, performa, dan kemudahan implementasi. Tiga alternatif utama yang dipertimbangkan adalah: pertama, menggunakan C/C++ yang memberikan kontrol penuh terhadap memori dan API Windows tetapi rentan terhadap *memory corruption* dan lebih mudah di-*reverse engineer*; kedua, menggunakan Python dengan *PyInstaller* yang relatif mudah dikembangkan namun *signature*-nya sudah familiar bagi *antivirus* dan kurang *resistant* terhadap *static analysis*; ketiga, menggunakan Rust yang menawarkan kombinasi unik antara kontrol *low-level*, *memory safety*, performa tinggi, dan resistensi inheren terhadap *reverse engineering*.

Analisis alternatif solusi tidak hanya melibatkan pertimbangan teknis tetapi juga evaluasi terhadap *capability* setiap bahasa dalam mengimplementasikan persyaratan fungsional dan non-fungsional yang telah ditetapkan. Untuk setiap alternatif, dilakukan penilaian terhadap enam kriteria desain: dukungan enkripsi dan *packing*, kontrol *low-level* untuk *fileless execution*, resistensi terhadap *analisis statis*, kemudahan implementasi teknik *anti-analysis*, performansi eksekusi, dan *ecosystem tools/libraries* yang tersedia. Penilaian ini dilakukan secara sistematis menggunakan metodologi *Multi-Criteria Decision Analysis* (MCDA) dengan *Entropy Weight Method* (EWM) untuk menentukan bobot kriteria secara objektif.

III.4.2 Analisis Penelitian Solusi

Analisis penelitian solusi menitikberatkan pada evaluasi efektivitas pendekatan yang telah terbukti dalam literatur untuk mendeteksi *dropper* dan *malware evasive*. Studi menunjukkan bahwa kombinasi deteksi berbasis perilaku (*behavioral analysis*), analisis anomali (*anomaly detection*), dan algoritma *machine learning* meningkatkan tingkat deteksi terhadap *dropper* dengan teknik *stealth* dan *fileless*. Namun, penelitian juga mengungkap bahwa metode-metode ini masih memiliki keterbatasan signifikan ketika dihadapkan pada *dropper* yang mengintegrasikan *multiple evasion techniques* secara simultan.

Implementasi *machine learning* dengan kurasi fitur yang relevan seperti *API call sequences*, entropi *payload*, dan *network flow patterns* terbukti mendongkrak presisi dan *recall* dalam klasifikasi *spyware*, menekankan pentingnya kualitas data dan pemilihan fitur untuk mengurangi *false positive*. Penggunaan *dynamic analysis* melalui *sandbox* untuk mengamati proses *unpacking dropper* yang tersembunyi di memori telah terbukti efektif, namun *sandbox* publik masih rentan terhadap *evasion techniques* seperti *time-based sleeping* dan *VM detection*. Integrasi data *threat intelligence* sekaligus deteksi berbasis perilaku adaptif diperlukan agar sistem mampu mengenali pola serangan baru yang belum pernah dijumpai sebelumnya.

Berdasarkan analisis ini, penelitian ini dirancang untuk mengembangkan *dropper* prototipe yang merepresentasikan ancaman tingkat *advanced* dengan *integrated evasion techniques*, sehingga evaluasi terhadap sistem deteksi yang ada akan menghasilkan temuan yang realistis dan *actionable*. Prototipe akan diuji terhadap kombinasi pendekatan deteksi modern-*signature-based*, *behavior-based*, dan *machine learning*-untuk memberikan gambaran komprehensif tentang efektivitas setiap metode.

III.4.3 Metodologi Penilaian Solusi (Multi-Criteria Decision Analysis)

Untuk menentukan bahasa implementasi optimal bagi *dropper* mode *stealth* yang akan dikembangkan, dilakukan evaluasi *multi-kriteria* terhadap tiga alternatif solusi: Rust, C/C++, dan Python dengan *PyInstaller*. Penilaian ini menggunakan *Multi-Criteria Decision Analysis* (MCDA) dengan *Entropy Weight Method* (EWM), yang mampu menentukan bobot kriteria secara objektif berdasarkan tingkat diversifikasi informasi dalam matriks keputusan. EWM dipilih karena objektivitasnya dalam penentuan bobot tanpa melibatkan subjektivitas *expert judgment*.

III.4.3.0.1 Langkah 1: Normalisasi Matriks Keputusan Skor mentah dari matriks keputusan (**X**) dinormalisasi menjadi matriks **P** untuk menghilangkan pengaruh unit dan skala pengukuran menggunakan:

$$P_{ij} = \frac{x_{ij}}{\sum_{i=1}^{m} x_{ij}} \tag{III.1}$$

Keterangan: P_{ij} adalah nilai normalisasi kriteria j untuk alternatif i, x_{ij} adalah skor mentah, m adalah jumlah alternatif ($m = 3$: Rust, C/C++, Python), dan n adalah jumlah kriteria ($n = 6$).

III.4.3.0.2 Langkah 2: Perhitungan Nilai Entropi Nilai entropi (e_j) dihitung untuk setiap kriteria j guna mengukur tingkat ketidakpastian atau dispersi data:

$$e_j = -k \sum_{i=1}^{m} P_{ij} \ln(P_{ij}), \qquad k = \frac{1}{\ln(m)} \tag{III.2}$$

III.4.3.0.3 Langkah 3: Penentuan Bobot Kriteria Derajat diversifikasi (d_j) diperoleh melalui:

$$d_j = 1 - e_j \tag{III.3}$$

Bobot relatif (w_j) untuk setiap kriteria kemudian dihitung dengan:

$$w_j = \frac{d_j}{\sum_{j=1}^{n} d_j} \tag{III.4}$$

Bobot mencerminkan kontribusi setiap kriteria terhadap pengambilan keputusan, di mana kriteria dengan variabilitas tinggi mendapat bobot lebih besar.

III.4.3.0.4 Langkah 4: Perhitungan Skor Komposit Akhir Skor komposit akhir (S_i) untuk setiap alternatif solusi dihitung menggunakan:

$$S_i = \sum_{j=1}^{n} w_j P_{ij} \tag{III.5}$$

Alternatif dengan skor tertinggi dipilih sebagai solusi yang paling optimal berdasarkan pertimbangan *multi-kriteria* yang objektif.

III.4.4 Hasil Analisis Penentuan Solusi

Perhitungan MCDA-EWM dilakukan terhadap matriks keputusan dengan enam kriteria desain dan tiga alternatif solusi. Kriteria yang dievaluasi adalah: KD-1 (Dukungan Enkripsi/*Packing*), KD-2 (Kontrol *Low-Level Fileless*), KD-3 (Resistensi Analisis Statis), KD-4 (Kemudahan Implementasi *Anti-Analysis*), KD-5 (Performa Eksekusi), dan KD-6 (*Ecosystem Tools/Libraries*). Hasil perhitungan entropi menunjukkan bahwa KD-2 memiliki entropi terendah dan derajat diversifikasi tertinggi, mengindikasikan bahwa kriteria ini paling efektif dalam membedakan ketiga alternatif. Tabel III.4 merangkum hasil perhitungan bobot kriteria.

Tabel III.4 Bobot kriteria hasil metode EWM

Kode Kriteria	**Deskripsi Kriteria**	e_j	d_j	w_j
KD-1	Dukungan Enkripsi/*Packing*	0.993	0.007	0.057
KD-2	Kontrol *Low-Level Fileless*	0.923	0.077	0.631
KD-3	Resistensi Analisis Statis	0.957	0.043	0.352
KD-4	Kemudahan Implementasi *Anti-Analysis*	0.989	0.011	0.090
KD-5	Performa Eksekusi	0.994	0.006	0.049
KD-6	*Ecosystem Tools/Libraries*	0.998	0.002	0.016
	Total		0.146	1.000

Nilai bobot menunjukkan bahwa KD-2 memberikan kontribusi terbesar (63.1%) terhadap pengambilan keputusan, diikuti oleh KD-3 (35.2%), sedangkan kriteria lainnya memiliki kontribusi kecil.

Skor komposit akhir untuk setiap alternatif disajikan pada Tabel III.5. Rust memperoleh skor tertinggi (0.370), diikuti oleh C/C++ (0.334), dan Python (0.296). Rust unggul terutama pada KD-2 dan KD-3 karena kombinasi uniknya antara kontrol *low-level* dan resistensi terhadap *analisis statis*.

Tabel III.5 Skor komposit akhir alternatif solusi

Alternatif Solusi	**Skor Komposit (S_i)**	**Keterangan**
Rust	0.370	Terpilih - unggul pada KD-2 (kontrol *low-level*) dan KD-3 (resistensi statis)
C/C++	0.334	Kuat pada KD-2 tetapi rentan pada KD-3 (lebih mudah di-*reverse engineer*)
Python	0.296	Skor rendah pada KD-2 (kontrol terbatas) dan KD-3 (*signature* sudah familiar)

Berdasarkan hasil analisis MCDA-EWM, Rust dipilih sebagai bahasa implementasi untuk *dropper spyware* mode *stealth*. Keputusan ini didasarkan pada skor tertinggi yang diperoleh dan justifikasi kuat bahwa Rust secara efektif memadukan kontrol *low-level* yang krusial untuk *fileless execution* dengan resistensi inheren terhadap *analisis statis* yang disasar oleh *spyware* mode *stealth* modern. Keputusan ini juga didukung oleh observasi industri bahwa *threat actors sophisticated* mulai mengadopsi Rust untuk pengembangan *custom malware* dan *reconnaissance tools*, menjadikan evaluasi *dropper* berbasis Rust sangat relevan dan tepat waktu.

BAB IV

DESAIN KONSEP SOLUSI

IV.1 Desain Solusi

Bab ini menyajikan penjelasan rinci mengenai rancangan sistem yang diusulkan dalam penelitian, yaitu desain modul *packer* dan *dropper spyware* dengan mode *stealth* yang akan digunakan untuk menguji kapabilitas deteksi *anti-malware* dalam lingkungan terisolasi. Sebagai pedoman metodologis, kerangka *Design Research Methodology* (DRM) yang telah diuraikan pada Bab I diadaptasi dalam bab ini, khususnya pada fase *Prescriptive Study*. Oleh karena itu, seluruh proses desain dalam bab ini diarahkan untuk menciptakan artefak yang tidak hanya memiliki fungsi teknis eksekusi *malware*, tetapi juga memenuhi kebutuhan pengujian parameter keamanan (seperti *evasion* dan *anti-analysis*) yang telah ditetapkan pada Bab III.

IV.2 Tahapan Desain

Proses desain dalam penelitian ini mencakup perancangan artefak berupa prototipe *Dropper Spyware Mode Stealth*. Pengembangan solusi dilakukan dengan mengikuti pedoman DRM untuk memastikan riset dilakukan secara efektif dan efisien. Fokus utama pada tahap ini adalah merjemahkan kebutuhan fungsional (FR) menjadi spesifikasi teknis dan arsitektur sistem yang siap diimplementasikan. Aktivitas utama dalam tahap desain ini mencakup:

- Perumusan Persyaratan: Mengacu pada Kebutuhan Fungsional (FR) dan Non-fungsional (NFR) terkait *stealth* dan *fileless execution*.
- Desain Arsitektur: Merancang hubungan logis antara modul *Packer*, *Dropper*, dan *Payload* sesuai dengan alur eksekusi yang direncanakan.
- Desain Komponen Detail: Merinci mekanisme teknik *evasion* (seperti *unpacking*, *deobfuscation*, *anti-VM*) dan eksekusi memori.
- Desain Lingkungan Uji: Menentukan spesifikasi sistem target (korban) dan

server (penyerang) dalam lingkungan terkontrol.

IV.2.1 Desain Artefak *Dropper*

Artefak yang dikembangkan berfokus pada mekanisme *dropper* yang bertanggung jawab untuk memastikan *spyware* berhasil melewati pertahanan awal (*Initial Access*) dan dieksekusi tanpa terdeteksi (*Execution*). Desain modul ini secara langsung menargetkan kelemahan deteksi berbasis tanda tangan (*signature-based*) dan analisis perilaku (*behavioral*) pada *anti-malware*.

IV.2.1.1 Arsitektur Modul dan Fungsi *Evasion*

Modul *dropper* dirancang sebagai kesatuan komponen yang terdiri dari lapisan pelindung (*Packer*) dan logika eksekusi (*Dropper Logic*). Tabel IV.1 menjabarkan komponen-komponen tersebut beserta justifikasinya terhadap analisis masalah.

Tabel IV.1 Komponen Modul *Dropper*

Komponen Modul	**Bahasa/Teknologi**	**Fungsi Utama dalam *Evasion***	**Justifikasi Pengujian (CTQ)**
Packer (Pelindung)	Rust	Melakukan *packing* (kompresi) dan enkripsi terhadap berkas *dropper* untuk mengubah *signature* statis dan menyembunyikan logika program.	Menyerang CTQ-01 (Celah *Signature-Evasion*).
Dropper Engine (Logika Utama)	Rust	Menjalankan alur logika: *Unpack*, *Deobfuscate*, *Anti-Analysis Check*, dan manajemen memori.	Menyerang CTQ-02 (*Obfuscation*) & CTQ-03 (Deteksi Perilaku).
Payload Handler	Rust / Windows API	Mengunduh *payload* dari server (jika tidak tersedia lokal) dan menyimpannya langsung ke memori (*fileless*) tanpa menulis ke disk.	Menyerang CTQ-03 (Celah *In-Memory*).
Server (C2)	Python / Go	Menyediakan *payload* (seperti *shellcode*) yang akan diunduh oleh *dropper* jika validasi lingkungan berhasil.	Pendukung skenario *Downloader-based* (FR-A5).

IV.2.1.2 Mekanisme Alur Logika *Dropper*

Desain alur logika *dropper* dirancang untuk memprioritaskan keamanan *payload*. *Dropper* tidak akan mengeksekusi kode berbahaya jika mendeteksi lingkungan analisis. Mekanisme ini dirancang berdasarkan diagram alir yang diusulkan, dengan rincian sebagai berikut:

1. **Inisialisasi & *Unpacking*.** Saat dieksekusi oleh target, *dropper* melakukan proses *unpacking* (2.1) dan *deobfuscation* (2.2) di memori untuk menerjemahkan kode yang disamarkan menjadi instruksi yang dapat dijalankan.
2. **Validasi Lingkungan (*Anti-Analysis*).** Sebelum melakukan aksi berbaha-

ya, *dropper* melakukan pengecekan (2.3) terhadap indikator *Virtual Machine* (VM) atau *Debugger*. Jika terdeteksi, proses akan berhenti untuk menghindari analisis.

3. **Manajemen *Payload*.** Sistem mengecek keberadaan *payload* internal (2.4). Jika tidak ada, *dropper* mengunduh *payload* dari Server (2.5).
4. ***Fileless Execution*.** *Payload* yang didapatkan (baik internal maupun unduhan) disimpan langsung ke alokasi memori (2.6) dan dieksekusi (2.7) menggunakan teknik *process injection* atau *thread execution*, memastikan tidak ada jejak fisik (file) yang tertinggal di *hard drive*.

IV.3 Rancangan Lingkungan Pengujian

Lingkungan pengujian dirancang untuk menciptakan kondisi yang terkontrol (*controlled environment*) dan relevan, sesuai dengan kebutuhan non-fungsional NFR-E1. Lingkungan ini harus terisolasi dari jaringan publik untuk keamanan, namun tetap mensimulasikan interaksi jaringan antara korban dan penyerang.

IV.4 Spesifikasi Mesin Uji

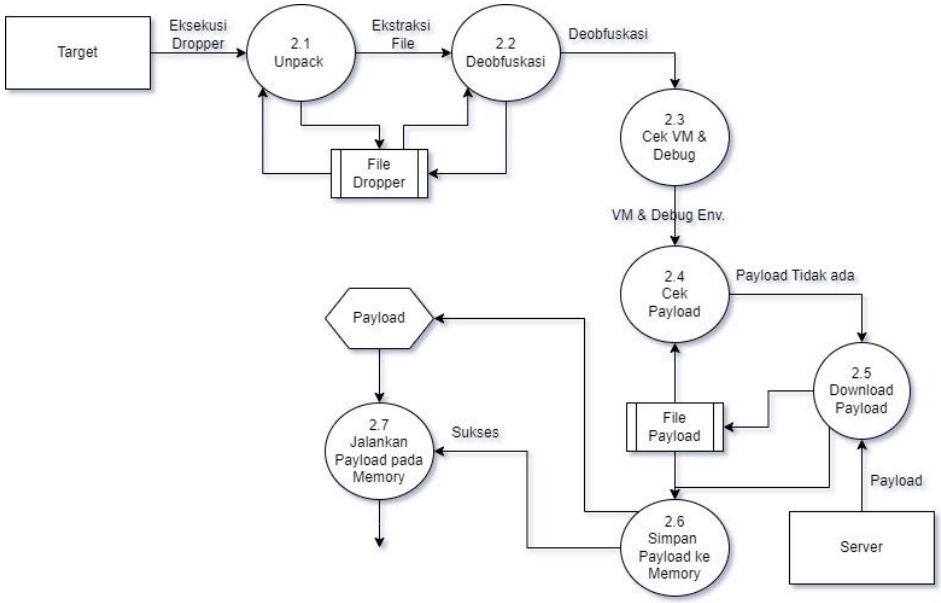
Pengujian akan dilakukan menggunakan dua entitas utama: Mesin Target (Korban) dan Mesin Server (Penyerang) yang berjalan di atas virtualisasi terisolasi (*Host-Only Network*). Spesifikasi rinci disajikan pada Tabel IV.2.

Tabel IV.2 Spesifikasi lingkungan uji

Komponen	**Spesifikasi Teknis**	**Justifikasi Desain**
Target (Korban)	OS: Windows 10 / 11 (x64) RAM: 4 GB	
Jaringan: *Host-Only*	Merepresentasikan lingkungan pengguna umum yang menjadi target *spyware*. Windows dipilih karena target utama *malware* berbasis .exe.	
Server (Penyerang)	OS: Linux (Kali/Ubuntu)	
Service: HTTP/TCP *Listener*	Berfungsi sebagai penyedia *payload* (untuk proses 2.5) dan penerima koneksi balik (*reverse shell*) dari *dropper*.	
Alat Monitoring	Sysmon, Wireshark, Process Hacker	Digunakan untuk memvalidasi apakah *dropper* benar-benar berjalan di memori (tidak menyentuh disk) dan memantau lalu lintas jaringan.
Anti-Malware	Berbagai produk komersial (Brand A, B, C, D, E)	Objek evaluasi utama untuk mengukur efektivitas deteksi terhadap desain *dropper*.

IV.5 Hasil Desain

Hasil akhir dari tahapan perancangan sistem ini adalah arsitektur teknis dan alur logika eksekusi *dropper* yang telah disesuaikan dengan karakteristik bahasa pemrograman Rust dan teknik *fileless*. Desain ini diilustrasikan dalam bentuk diagram alir proses yang menggambarkan siklus hidup *dropper* mulai dari inisialisasi hingga eksekusi *payload*.



Gambar IV.1 Diagram alir eksekusi *dropper spyware* mode *stealth*

IV.5.1 Tahap Inisialisasi dan Dekripsi (Proses 2.1 & 2.2)

Ketika target mengeksekusi berkas *dropper*, program tidak langsung menjalankan kode berbahaya.

1. **2.1 *Unpack*:** *Dropper* yang dikemas (*packed*) akan melakukan *self-extraction* di memori. Proses ini bertujuan untuk mengembalikan bentuk asli kode biner yang telah dikompresi atau dienkripsi saat pembuatan (*compile time*).
2. **2.2 *Deobfuscasi*:** Setelah di-*unpack*, *dropper* melakukan *deobfuscasi* terhadap string dan konfigurasi internal. Teknik ini digunakan untuk menyembunyikan *Import Address Table* (IAT) dan string sensitif (seperti alamat IP Server C2 atau nama fungsi API Windows) dari analisis statis *anti-malware*. Penggunaan bahasa Rust membantu mempersulit proses *reverse engineering* pada tahap ini karena struktur binernya yang kompleks.

IV.5.2 Tahap Validasi Lingkungan / *Anti-Analysis* (Proses 2.3)

Tahap ini merupakan gerbang keamanan bagi *malware* untuk melindungi dirinya sendiri dari analisis peneliti keamanan atau *sandbox* otomatis.

- **2.3 Cek VM & Debug:** Sebelum memuat *payload*, *dropper* memindai lingkungan eksekusi untuk mencari tanda-tanda keberadaan *debugger* (misalnya memeriksa flag *IsDebuggerPresent*) atau artefak mesin virtual (seperti *registry keys* VMware/VirtualBox atau instruksi CPUID tertentu).
- **Logika Keputusan:**

- Jika lingkungan terdeteksi sebagai VM atau *Debug Environment*, *dropper* akan melakukan terminasi dini atau menjalankan kode *benign* (tidak berbahaya) untuk mengecoh analisis (menghasilkan *False Negative*).
- Jika lingkungan terkonfirmasi sebagai *Target Fisik*, proses berlanjut ke tahap manajemen *payload*.

IV.5.3 Tahap Manajemen *Payload* (Proses 2.4 & 2.5)

Desain ini mendukung fleksibilitas antara membawa *payload* sendiri (*dropper*) atau mengunduhnya (*downloader*).

1. **2.4 Cek *Payload*:** Sistem memeriksa apakah *payload* utama (*spyware*) sudah tertanam secara terenkripsi di dalam biner *dropper*.
2. **2.5 *Download Payload* (Kondisional):** Jika *payload* tidak ditemukan secara internal (skenario *Payload Tidak Ada*), *dropper* akan menginisiasi koneksi HTTP/TCP ke Server penyerang untuk mengunduh *payload* terbaru secara aman ke memori.

IV.5.4 Tahap Eksekusi *Fileless* (Proses 2.6 & 2.7)

Ini adalah inti dari mekanisme *stealth* yang dirancang untuk menghindari deteksi *file-scanning*.

1. **2.6 Simpan *Payload* ke Memori:** *Payload* (baik dari internal maupun unduhan) didekripsi dan disimpan langsung ke dalam alokasi memori volatil (RAM). Tidak ada penulisan berkas (*file write*) ke *hard disk* pada tahap ini, memitigasi risiko deteksi forensik berbasis disk.
2. **2.7 Jalankan *Payload* pada Memori:** Menggunakan teknik *Process Injection* atau pembuatan *thread* jarak jauh, *dropper* mengeksekusi kode *payload* langsung dari lokasi memori tersebut. Setelah *payload* aktif (Status: Sukses), proses *dropper* utama akan menutup diri (*self-terminate*), meninggalkan *payload* berjalan secara independen dan tersembunyi di dalam proses sistem yang sah.

BAB V

RENCANA SELANJUTNYA

V.1 Rencana Implementasi

Subbab ini menjelaskan langkah implementasi prototipe dropper spyware mode stealth berbasis Rust, serta estimasi waktu dan sumber daya yang dibutuhkan.

V.1.1 Langkah-langkah Implementasi

Implementasi dibagi menjadi beberapa tahap utama yang mengikuti alur desain pada Bab IV, yaitu: pengembangan modul enkripsi payload, modul anti-analysis, modul eksekusi fileless, integrasi C2 dummy untuk pengujian, dan integrasi akhir beserta uji coba internal. Setiap tahap diakhiri dengan uji fungsional dasar untuk memastikan pemenuhan FR/NFR yang telah ditetapkan sebelumnya.

Tabel berikut merangkum timeline implementasi secara ringkas (estimasi berbasis minggu kalender):

Tabel V.1 Timeline implementasi prototipe

No	Tahap Implementasi	Aktivitas Kunci	Estimasi Waktu
1	Desain detail & setup environment	Refinement spesifikasi teknis, setup repo, VM lab, toolchain Rust	2 minggu
2	Modul Payload Encryptor & Container	Implementasi AES, kompresi, embedding payload, uji enkripsi/dekripsi	3 minggu
3	Modul Environment Checker	Implementasi anti-VM, anti-sandbox, anti-debug (CPUID, registry, process scan)	3 minggu

No	Tahap Implementasi	Aktivitas Kunci	Estimasi Waktu
4	Modul Fileless Loader & Injection	Implementasi in-memory execution, process injection, uji di VM bersih	4 minggu
5	Integrasi C2 Dummy & Logging	Kanal C2 sederhana untuk verifikasi payload aktif, logging minimal	2 minggu
6	Integrasi, hardening, dan uji internal	Penggabungan seluruh modul, optimasi, smoke test multi-VM	2–3 minggu

Perkiraan total durasi implementasi adalah sekitar 16–17 minggu, termasuk buffer untuk bug fixing ringan dan penyesuaian teknis.

V.1.2 Alat yang Dibutuhkan

Alat dan bahan yang dibutuhkan mencakup perangkat pengembangan, lingkungan virtual untuk pengujian aman, serta perangkat lunak pendukung analisis.

Tabel V.2 Alat dan bahan pendukung implementasi

Kategori	Alat/Bahan	Kegunaan
Perangkat keras	Laptop/PC dengan minimal CPU 4 core, RAM 16 GB, SSD 512 GB	Pengembangan dan menjalankan beberapa VM uji
Virtualisasi	VirtualBox/VMware/Hyper-V	Menyediakan VM Windows 10/11 untuk eksekusi dropper
Bahasa & toolchain	Rust (rustup, cargo, rust-analyzer)	Implementasi dropper dan modul pendukung
Library utama	winapi, aes-gcm, sha2, sysinfo	Akses WinAPI, enkripsi, hashing, profiling lingkungan
OS target	Windows 10/11 Pro (VM)	Lingkungan uji dropper dan anti-malware
Anti-malware/EDR	Windows Defender, serta 2–3 produk komersial lain (mis. produk A, B, C)	Objek evaluasi kapabilitas deteksi

Kategori	Alat/Bahan	Kegunaan
Tool analisis	Sysmon, Procmon, Wireshark, x64dbg/WinDbg	Observasi perilaku, logging, dan debugging eksekusi

Analisis biaya dapat disederhanakan dengan asumsi penggunaan lisensi akademik/gratis untuk sebagian besar perangkat lunak, sehingga biaya utama adalah penyediaan mesin pengembangan dan penyimpanan untuk log evaluasi.

V.2 Rencana Evaluasi

V.2.1 Metode Pengujian

Pengujian dilakukan di laboratorium terisolasi dengan beberapa VM Windows 10/11 yang masing-masing dipasangi produk anti-malware/EDR berbeda. Dropper dieksekusi dalam tiga skenario utama: pemindaian statis, eksekusi di VM bersih, dan eksekusi di VM yang dikonfigurasi menyerupai lingkungan analisis (VM/sandbox/debugger) untuk menguji mekanisme anti-analysis.

V.2.2 Metrik Evaluasi

Metrik digunakan untuk mengukur efektivitas dropper sekaligus kemampuan deteksi tiap produk anti-malware terhadap teknik evasi yang diimplementasikan. Ringkasan metrik disajikan pada Tabel berikut.

Tabel V.3 Metrik evaluasi implementasi dropper

No	Metrik	Definisi Singkat	Sumber Data
1	Detection Rate (DR)	Persentase engine/produk yang menandai dropper atau payload sebagai malicious pada pemindaian statis dan eksekusi	Log multi-engine scanner, log anti-malware di VM
2	Evasion Rate (ER)	Persentase eksekusi di mana dropper berhasil menjalankan payload tanpa terblokir pada tahap awal (before payload active)	Hasil eksekusi berulang pada tiap produk

No	Metrik	Definisi Singkat	Sumber Data
3	Waktu Deteksi (Time-to-Detect)	Selisih waktu antara start dropper dan munculnya alert pertama dari anti-malware/EDR	Timestamp log produk keamanan dan Sysmon
4	Keberhasilan Anti-Analysis	Persentase eksekusi di VM/sandbox/debugger yang dihentikan oleh modul anti-VM/anti-debug sebelum payload aktif	Log internal dropper dan observasi perilaku di VM analisis
5	Stabilitas Eksekusi	Persentase run di VM bersih di mana dropper dan payload selesai tanpa crash atau error fatal	Catatan run dan log sistem di VM bersih
6	Jejak Forensik	Jumlah dan jenis artefak permanen (file, registry) yang tertinggal setelah eksekusi	Hasil inspeksi file system, registry, dan log Sysmon

Nilai-nilai metrik ini kemudian dibandingkan dengan kriteria keberhasilan pada Subbab V.2.3 (misalnya batas maksimum DR statis, minimum ER, dan batas artefak yang diperbolehkan) untuk menilai apakah prototipe memenuhi tujuan penelitian.

V.2.3 Kriteria Keberhasilan

Kriteria keberhasilan dirumuskan berdasarkan CTQ yang telah didefinisikan di Bab III dan difokuskan pada efektivitas evasion serta stabilitas eksekusi.

Tabel V.4 Kriteria keberhasilan prototipe dropper

No	Kriteria	Deskripsi Singkat	Target
1	CTQ-01 – Payload concealment	Payload terenkripsi tidak terdeteksi pada fase pemindaian statis di mayoritas engine	Detection rate statis $\leq 30\%$

No	Kriteria	Deskripsi Singkat	Target
2	CTQ-02 – Signature evasion	Biner dropper lolos dari signature-based detection pada beberapa produk utama	Minimal 75% produk komersial tidak mendeteksi pada eksekusi awal
3	CTQ-03 – Environment & behavioral stealth	Mekanisme anti-VM/anti-debug mencegah eksekusi penuh di VM analisis, serta eksekusi di VM bersih tidak langsung memicu alert perilaku	$>$ 80% run di VM analisis berhenti sebelum payload aktif, dan minimal satu produk terlambat mendeteksi pada VM bersih
4	Stabilitas eksekusi	Dropper berhasil mengeksekusi payload tanpa crash pada VM bersih	Keberhasilan eksekusi $\geq$ 95% dari total percobaan
5	Minimnya artefak forensik	Tidak ada file permanen atau entri registry baru terkait dropper setelah eksekusi	Tidak ditemukan artefak disk yang persisten pada inspeksi manual

Jika sebagian kriteria tidak tercapai, hasil tersebut tetap didokumentasikan sebagai temuan ilmiah dan dasar rekomendasi perbaikan deteksi maupun desain dropper pada penelitian lanjutan.

V.3 Analisis Risiko

Analisis risiko disusun secara ringkas untuk mengidentifikasi hambatan utama selama implementasi dan evaluasi prototipe, serta strategi mitigasinya.

Tabel V.5 Risiko implementasi dan mitigasi

No	Risiko	Dampak Utama	Mitigasi Singkat
1	Implementasi fileless injection lebih sulit dari perkiraan	Timeline pengembangan mundur, beberapa FR tidak tercapai	Memulai lebih awal pada modul loader, membatasi teknik injection ke 1–2 metode yang paling stabil, dan menyiapkan fallback berupa eksekusi semi-fileless jika perlu
2	Deteksi anti-malware terlalu tinggi sehingga tidak merepresentasikan dropper stealth	Evaluasi tidak menggambarkan gap deteksi yang realistis	Iteratif menyesuaikan konfigurasi enkripsi, packing, dan anti-analysis sampai biner mencapai profil deteksi yang seimbang namun tetap aman didokumentasikan
3	Lab tidak sepenuhnya terisolasi	Potensi gangguan ke jaringan luar, risiko etis	Menggunakan jaringan host-only/NAT tertutup, memblokir akses internet dari VM, dan menggunakan C2 dummy lokal saja
4	Keterbatasan waktu dibanding kompleksitas eksperimen	Beberapa skenario uji tidak sempat dijalankan	Memprioritaskan produk anti-malware yang paling relevan, menyusun jadwal pengujian terstruktur, dan menyimpan skenario tambahan sebagai pekerjaan lanjutan
5	Koordinasi dengan pembimbing tidak lancar	Revisi konsep terlambat, kualitas artefak menurun	Menetapkan jadwal konsultasi rutin dan mengirimkan progres singkat tertulis agar umpan balik tetap terjaga meski jadwal padat

Melalui perencanaan implementasi yang terstruktur dan evaluasi yang ketat, penelitian ini diharapkan mampu memberikan data empiris mengenai celah keamanan pada sistem deteksi modern terhadap ancaman *dropper* berbasis Rust.

DAFTAR PUSTAKA

Ainslie, Scott, Dean Thompson, Sean Maynard, dan Atif Ahmad. 2023. "Cyber-threat Intelligence for Security Decision-Making: A Review and Research Agenda for Practice". *Computers & Security* 132.

Blessing, Lucienne T.M., dan Amaresh Chakrabarti. 2009. *DRM, a Design Research Methodology.*

Chatzoglou, E., G. Karopoulos, G. Kambourakis, dan Z. Tsiatsikas. 2023. "Bypassing Antivirus Detection: Old-School Malware, New Tricks". Dalam *Proceedings of TrustBus'23.*

Elghaly, Y. M. 2024. "Stealth in Plain Sight: The Hidden Threat of PowerShell Fileless Malware and Its Evasion of Modern EDRs & AVs", University of East London.

Kareem, K.M. 2024. "A Comprehensive Analysis of Pegasus Spyware and Its Implications for Digital Privacy and Security".

Koutsokostas, Vasilios, dan Constantinos Patsakis. 2021. "Python and Malware: Developing Stealth and Evasive Malware Without Obfuscation". *arXiv preprint arXiv:2105.00565.*

Reddyvari, V., M.U.M. Rao, S.S.D. Reddy, K.R.S. Reddy, dan B.M. Nayak. 2024. "A Review on Spyware Creation and Detection". *International Journal of Engineering Research & Technology (IJERT)* 13 (3).

Sudhakar, S., dan S. Kumar. 2020. "An Emerging Threat: Fileless Malware - A Survey and Research Challenges". *Cybersecurity* 3 (1).