EVALUASI KAPABILITAS DETEKSI ANTI-SPYWARE TERHADAP PACKER SPYWARE MODE STEALTH

Proposal Tugas Akhir

Oleh

Nathaniel Liady
18222114

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

Desember 2025

LEMBAR PENGESAHAN

EVALUASI KAPABILITAS DETEKSI ANTI-SPYWARE TERHADAP PACKER SPYWARE MODE STEALTH

Proposal Tugas Akhir

Oleh

Nathaniel Liady
18222114

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

Prof. Dr. Ir. Suhardi, M.T.

NIP. 123456789

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR KODE

BAB I

PENDAHULUAN

I.1 Latar Belakang

Ancaman siber saat ini telah mengalami evolusi signifikan, bergeser dari *malware* tradisional berbasis *file* menuju serangan yang lebih canggih dan tersembunyi, seperti *fileless malware* dan *spyware*. Evolusi ini menciptakan tantangan mendasar bagi sistem *anti-malware* konvensional, karena pertahanan yang mengandalkan deteksi berbasis tanda tangan (*signature-based*) menjadi tidak efektif. *Fileless malware* secara khusus memanfaatkan *utility* sistem operasi yang sah, seperti PowerShell dan Windows Management Instrumentation (WMI), untuk menjalankan kode berbahaya langsung di memori (*in-memory*) tanpa meninggalkan jejak *file* pada *disk*.

Kelemahan sistem pertahanan ini diperkuat oleh hasil studi empiris yang menguji kapabilitas deteksi produk keamanan komersial. Penelitian telah menunjukkan bahwa teknik *evasion* sederhana seperti enkripsi dan injeksi proses terbukti sangat efektif dalam menghindari deteksi. Bahkan, dalam sebuah studi di tahun 2023, sejumlah *anti-malware* yang diuji hanya mampu mendeteksi sebagian kecil dari varian *malware* yang disamarkan. Temuan ini menegaskan bahwa terdapat kerentanan substansial terhadap *malware* lama yang dimodifikasi dengan trik penyamaran baru, sementara *scanner anti-virus* juga sering gagal menganalisis kode yang dikemas.

Penelitian ini memfokuskan diri pada pengembangan dan pengujian modul *Packer Spyware Mode Stealth*. *Packer* ini dirancang untuk mencapai keberhasilan *Initial Access* dengan menerapkan teknik *obfuscation* tingkat tinggi yang langsung menyerang kelemahan inti *anti-malware: Signature-Evasion* dan *In-Memory Execution*. Karena *spyware* modern memanfaatkan teknik penghindaran analisis dan deteksi baik dalam bentuk statis maupun dinamis, penelitian ini berupaya menghasilkan artefak yang sulit teridentifikasi.

Dari berbagai temuan tersebut, terlihat jelas bahwa terdapat *gap* signifikan pada kemampuan deteksi *signature-based* dan *behavior-based* terhadap teknik *packing*, *obfuscation*, dan *fileless execution*. Gap inilah yang menjadi fokus utama penelitian ini, yang bertujuan untuk secara empiris menguji dan menganalisis kemampuan deteksi solusi keamanan yang tersedia di pasar terhadap *spyware* kustom.

I.2 Rumusan Masalah

Berdasarkan latar belakang di atas, maka rumusan masalah dalam penelitian ini adalah:

1. Bagaimana kapabilitas sistem *anti-spyware* dalam mendeteksi aktivitas tersembunyi dari perilaku *spyware mode stealth* pada lingkungan uji terkontrol?
2. Bagaimana pendekatan paling efektif untuk mengembangkan *packer spyware stealth* yang mampu menghindari deteksi *anti-malware*
3. Bagaimana teknik penyamaran (*obfuscation*) dan metode eksekusi berbasis memori (*in-memory execution*) mempengaruhi kemampuan deteksi produk *anti-spyware* saat ini?
4. Apa saja celah keamanan dan kelemahan spesifik yang ditemukan pada produk *anti-spyware* dan EDR ketika dihadapkan pada serangan *spyware* kustom yang dirancang untuk menghindari deteksi?

I.3 Tujuan

Secara umum, tujuan dari pelaksanaan tugas akhir ini adalah untuk mengukur dan mengevaluasi efektivitas produk *anti-spyware* dan EDR komersial dalam mendeteksi *spyware mode stealth* yang dikembangkan dengan teknik-teknik penghindaran deteksi modern.

Secara spesifik, tujuan yang ingin dicapai adalah:

1. Menganalisis dan mengidentifikasi teknik-teknik evasi yang paling efektif digunakan oleh *spyware* untuk menghindari deteksi dari *anti-spyware* dan EDR.
2. Mengembangkan sebuah prototipe *spyware mode stealth*, termasuk *reverse shell fileless PowerShell*, yang menggabungkan teknik-teknik evasif yang telah diidentifikasi.
3. Melakukan pengujian komparatif prototipe *spyware* pada sejumlah produk *anti-spyware* dan EDR komersial untuk mengevaluasi tingkat keberhasilan dan kegagalan deteksi.
4. Mendokumentasikan dan mempublikasikan celah keamanan yang ditemukan

pada produk *anti-spyware*, serta menyusun rekomendasi mitigasi untuk pengembangan sistem pertahanan yang lebih adaptif dan tangguh.

kriteria keberhasilan dari pelaksanaan tugas akhir ini adalah:

- Prototipe *spyware* berhasil dikembangkan dengan setidaknya dua teknik evasif (misalnya, enkripsi dan obfuscation) dan mampu menghindari deteksi oleh salah satu produk *anti-spyware* yang diuji.
- Hasil pengujian menunjukkan bahwa ada perbedaan signifikan dalam tingkat deteksi antara format skrip dan *anti-spyware* yang berbeda.

Kedua kriteria tersebut menjadi indikator utama untuk menilai efektivitas penelitian, serta menunjukkan sejauh mana solusi yang ditawarkan mampu mengungkap kelemahan sistem keamanan siber saat ini. Pencapaian terhadap kriteria ini diharapkan dapat menjadi dasar pertimbangan dalam peningkatan kapabilitas deteksi *anti-spyware* dan EDR di masa depan

I.4 Batasan Masalah

Batasan masalah dalam pelaksanaan tugas akhir adalah sebagai berikut:

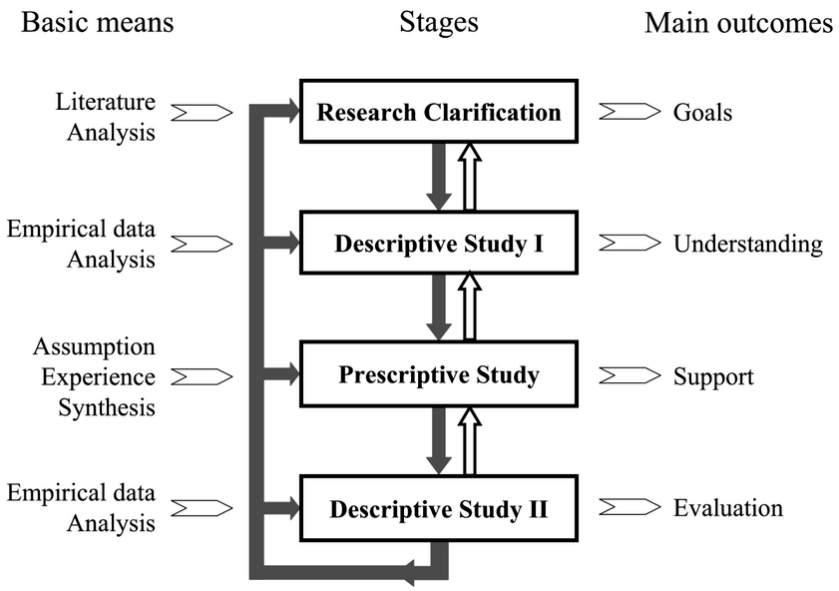
1. Penelitian ini hanya berfokus pada pengujian kapabilitas deteksi *anti-spyware* terhadap aktivitas awal (*Initial Access*) yang dilakukan oleh *spyware mode stealth* pada sistem operasi *desktop*.
2. Evaluasi hanya mencakup perangkat anti *spyware* yang dipilih sesuai kriteria penelitian, tidak membandingkan seluruh produk anti malware yang ada.
3. Aktivitas *malware* yang diujikan dibatasi secara ketat pada tahap penyusupan, enkripsi *payload*, *fileless execution*, dan upaya *persistence*.
4. 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 yang digunakan adalah *Design Research Methodology* (DRM) dikenalkan oleh Blessing dan Chakrabarti (2009).[1] DRM dibuat dengan tujuan supaya riset dilakukan dengan lebih efektif dan efisien. Metodologi ini terdiri dari

1. L. T. M. Blessing & A. Chakrabarti, Design Research Methodology, 2009.

empat tahap utama yaitu Research Clarification (RC), Descriptive Study I (DS-I), Perspective Study (PS), dan Descriptive Study II (DS-II). Berikut ini adalah gambaran dari kerangka kerja DRM.



Gambar I.1 *Design Research Methodology Framework*

1. *Research Clarification (RC)*
   Fase ini akan dimulai dengan identifikasi masalah utama: adanya celah yang signifikan antara teknik serangan siber modern, khususnya *spyware mode stealth*, dan kemampuan deteksi solusi keamanan yang ada. Pengumpulan data awal akan dilakukan melalui tinjauan literatur komprehensif, termasuk laporan industri, artikel akademis, dan berita, untuk memahami lanskap ancaman dan teknik evasif yang digunakan untuk menghindari deteksi *anti-spyware* dan EDR. Hasil dari fase ini adalah rumusan masalah yang jelas dan terperinci, yang akan menjadi landasan untuk seluruh penelitian.
2. *Descriptive Study I (DS-I)*
   Pada fase ini, analisis mendalam terhadap masalah yang telah dirumuskan akan dilakukan dengan mengumpulkan data dan informasi yang relevan. Ini mencakup analisis rinci mengenai teknik-teknik evasi canggih, seperti penggunaan *PowerShell* dan obfuscation, untuk memahami bagaimana ancaman ini bekerja dan mengapa mereka sulit dideteksi. Selain itu, studi-studi terdahulu yang telah menguji bypass *antivirus* akan dikaji untuk mendapatkan wawasan mengenai metode dan potensi hasil. Sebagai bagian penting dari fase ini, alur kerja atau arsitektur teknis dari serangan *spyware* akan disusun, yang akan menjelaskan fase-fase seperti *Initial Access, Establish Foothold,*

Persistence, Data Collection, dan *Exfiltration*. Diagram alur yang telah ada akan menjadi representasi visual dari arsitektur ini.

3. *Perspective Study (PS)*

   Fase ini merupakan inti dari DRM, di mana solusi untuk masalah yang telah didefinisikan akan dirancang dan dikembangkan. Artefak yang akan dibuat adalah prototipe *spyware mode stealth* dengan fungsi-fungsi spesifik seperti *keylogging* dan *screen capture*. Desain prototipe akan mengintegrasikan teknik evasi yang telah diidentifikasi pada fase sebelumnya, seperti enkripsi *payload* dan penggunaan PowerShell untuk menjalankan kode langsung di memori. Berbagai modul, termasuk Packer untuk menyamarkan *payload*.

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

   Fase terakhir ini bertujuan untuk menguji dan mengevaluasi efektivitas solusi yang telah dikembangkan. Serangkaian pengujian eksperimental akan dilakukan dalam lingkungan virtual yang terisolasi untuk mengukur tingkat deteksi berbagai produk *anti-spyware* terhadap prototipe *spyware*. Hasil pengujian akan dianalisis untuk mengidentifikasi teknik evasi mana yang paling efektif dan mengapa produk keamanan tertentu gagal atau berhasil dalam mendeteksi ancaman. Analisis ini akan memvalidasi temuan awal dan memberikan kontribusi nyata pada pemahaman tentang kerentanan sistem keamanan modern, yang akan menjadi dasar untuk rekomendasi perbaikan dan penelitian lebih lanjut.

BAB II

STUDI LITERATUR

Bab ini membahas landasan teori dan kajian literatur yang relevan untuk mendukung penelitian ini. Studi literatur dilakukan untuk memahami konsep, teori, dan teknologi yang mendasari penelitian, termasuk *information gathering, social engineering, cyber security, Anti-Spyware, malware*, dan *spyware*. Selain itu, kajian terhadap penelitian terdahulu yang berkaitan juga dilakukan untuk mengidentifikasi solusi yang sudah ada, celah penelitian, serta pendekatan yang dapat diadopsi atau dikembangkan lebih lanjut. Hasil dari studi literatur ini akan menjadi dasar dalam merumuskan solusi desain yang diusulkan dalam penelitian ini.

II.1 *Information Gathering*

Information gathering merupakan tahap awal yang sangat penting dalam proses pengujian keamanan siber maupun kegiatan intelijen digital. Tahap ini bertujuan untuk mengumpulkan informasi sebanyak mungkin mengenai target, baik berupa sistem, jaringan, organisasi, maupun individu, dengan tujuan memahami permukaan serangan yang tersedia. Verma dkk. (2021) menjelaskan bahwa kegiatan *information gathering* dilakukan untuk memetakan aset dan mengidentifikasi potensi kerentanan sebelum serangan atau pengujian keamanan dilakukan. Secara umum, proses ini terbagi menjadi dua pendekatan utama, yaitu pasif dan aktif. Pengumpulan informasi secara pasif dilakukan tanpa melakukan interaksi langsung dengan sistem target, misalnya dengan memanfaatkan data publik dari mesin pencari, basis data domain, atau sumber intelijen terbuka (*Open Source Intelligence/OSINT*). Sebaliknya, pendekatan aktif melibatkan aktivitas langsung seperti *port scanning, service enumeration*, atau *banner grabbing* untuk memperoleh data teknis yang lebih spesifik, namun metode ini berisiko lebih tinggi untuk terdeteksi oleh sistem keamanan target.

Teknik *information gathering* secara tradisional terdiri atas tiga tahap utama, yaitu *footprinting, scanning*, dan *enumeration*. Pada tahap *footprinting*, peneliti mengumpulkan informasi dasar seperti alamat IP, nama domain, sistem operasi, serta teknologi yang digunakan. Tahap *scanning* bertujuan untuk menemukan *host* aktif dan *port* terbuka, sementara *enumeration* melibatkan eksplorasi lebih dalam terhadap layanan, pengguna, atau konfigurasi sistem yang dapat dimanfaatkan dalam tahap berikutnya. Seiring berkembangnya teknologi, berbagai alat bantu seperti Nmap, Wireshark, The Harvester, Netcraft, dan Metagoofil banyak digunakan untuk mendukung proses pengumpulan informasi ini. Studi Verma dkk. (2021) juga menyoroti pentingnya kombinasi antara OSINT dan pemindaian aktif untuk meningkatkan efektivitas deteksi potensi risiko, meskipun penggunaan metode ini harus dibatasi dalam ruang lingkup yang legal dan etis.

Selain aspek teknis, penelitian terbaru menekankan pentingnya aspek etika dan hukum dalam kegiatan *information gathering*. Pengumpulan data secara berlebihan atau tanpa izin dapat melanggar privasi individu maupun regulasi keamanan informasi. Oleh karena itu, para peneliti dianjurkan untuk melakukan kegiatan ini dalam lingkungan laboratorium yang terisolasi, dengan batasan yang jelas dan persetujuan dari pihak terkait. Di sisi lain, hasil pengumpulan informasi juga harus dikelola dengan prinsip *responsible disclosure*, yaitu melaporkan temuan yang berpotensi sensitif kepada pihak yang berwenang tanpa menyebarkannya secara publik. Dengan demikian, *information gathering* tidak hanya berfungsi sebagai tahapan teknis untuk mendukung pengujian keamanan, tetapi juga sebagai fondasi penting dalam membangun kesadaran, kebijakan, dan strategi pertahanan siber yang lebih komprehensif.

II.2 *Social Engineering*

Social engineering adalah teknik manipulasi psikologis yang mengeksploitasi kelemahan manusia, bukan kerentanan teknis pada sistem keamanan. Dalam lanskap siber yang terus berkembang, *social engineering* menjadi salah satu ancaman paling signifikan dan efektif. Sejak tahun 2021, serangan *social engineering* telah meningkat baik dari segi volume maupun kecanggihan, dengan penjahat siber dan kelompok terorganisir mengeksploitasi bias kognitif dan emosi manusia untuk menipu. Ini menjadikan faktor manusia sebagai mata rantai terlemah dalam keamanan siber

Penelitian terbaru mengidentifikasi berbagai metode serangan *social engineering*,

mulai dari yang sederhana hingga yang sangat canggih. *Phishing, spear phishing*, dan *whaling* adalah serangan yang menggunakan email atau pesan palsu yang disesuaikan untuk menipu individu atau target tingkat tinggi. Serangan *phishing* di media sosial juga dapat menjangkau audiens yang lebih luas daripada email konvensional. Selain itu, *pretexting* adalah ketika penyerang menciptakan skenario palsu untuk mendapatkan informasi sensitif atau akses, sering kali dengan menyamar sebagai rekan kerja atau figur otoritas. Ada juga metode *baiting* yang menggunakan umpan (seperti file berbahaya) atau manipulasi suara untuk memancing korban agar mengambil tindakan yang membahayakan. Penyerang juga dapat melakukan *impersonation*, yaitu menyamar sebagai individu yang dikenal atau dipercaya, sebuah taktik yang lebih mudah dilakukan di media sosial karena melimpahnya informasi korban. Untuk meningkatkan presisi dan skalabilitas, penjahat siber kini juga memanfaatkan teknologi canggih seperti kecerdasan buatan (AI) dan *deepfake* untuk membuat serangan *social engineering* lebih meyakinkan.

Berbagai studi telah mengidentifikasi beberapa faktor yang membuat individu rentan terhadap serangan *social engineering*. Kesadaran keamanan yang rendah adalah salah satu faktor utama. Penyerang juga mengeksploitasi emosi seperti ketakutan, urgensi, rasa ingin tahu, dan kepercayaan untuk mendorong korban membuat keputusan yang salah. Kepercayaan berlebihan pada figur otoritas juga membuat korban lebih mudah dimanipulasi. Serangan-serangan ini memiliki dampak yang signifikan, termasuk kerugian finansial, kerusakan reputasi, dan hilangnya data. Studi kasus skema penipuan keuangan yang menargetkan Google dan Facebook menunjukkan bahwa organisasi dengan sistem keamanan yang kuat pun tidak kebal terhadap *social engineering*.

II.3 *Cyber Security*

Cyber security merupakan disiplin ilmu dan praktik yang berfokus pada perlindungan sistem komputer, jaringan, data, serta perangkat digital dari berbagai ancaman yang dapat mengganggu kerahasiaan, integritas, dan ketersediaan informasi. Menurut Ainslie dkk. (2023), keamanan siber tidak hanya menjadi isu teknis, melainkan juga tantangan strategis yang berpengaruh terhadap pengambilan keputusan di tingkat organisasi. Dalam konteks modern, setiap keputusan bisnis harus mempertimbangkan potensi risiko siber, karena ancaman digital kini dapat berdampak langsung pada keberlanjutan operasional dan reputasi perusahaan. Oleh karena itu, *cyber security* mencakup kombinasi aspek teknologi, manusia, dan kebijakan organisasi yang bekerja secara terpadu untuk mencegah, mendeteksi, dan merespons

insiden keamanan secara efektif.

Komponen utama dalam *cyber security* meliputi perlindungan data dan privasi, pengelolaan risiko, deteksi serta respons terhadap insiden, hingga penerapan *Cyber Threat Intelligence (CTI)* yang berfungsi untuk mengidentifikasi pola serangan dan memberikan wawasan bagi pengambilan keputusan keamanan Ainslie dkk. (2023). Selain itu, munculnya ancaman baru seperti fileless malware turut memperluas ruang lingkup keamanan siber. Berdasarkan penelitian Sudhakar dan Kumar (2020), *fileless malware* beroperasi langsung di memori tanpa menyimpan file berbahaya di sistem, sehingga sulit dideteksi oleh *antivirus* tradisional yang berbasis tanda tangan. Ancaman ini menunjukkan bahwa sistem pertahanan harus bergeser dari deteksi berbasis file menuju pendekatan berbasis perilaku dan analisis memori.

Dalam penerapannya, pendekatan holistik menjadi kunci keberhasilan manajemen keamanan siber. FLECO, sebuah kerangka kerja yang dikembangkan oleh Domínguez-Dorado dkk. (2024), menekankan pentingnya integrasi antara aspek teknologi, tata kelola, dan budaya organisasi untuk membangun sistem keamanan yang berkelanjutan. Pendekatan ini membantu organisasi dalam mengukur kesiapan keamanan siber dan memperkuat koordinasi lintas departemen agar setiap unit memahami tanggung jawabnya dalam menjaga keamanan digital. Dengan demikian, *cyber security* tidak hanya berfungsi untuk merespons ancaman yang terjadi, tetapi juga sebagai strategi proaktif yang melibatkan seluruh komponen organisasi dalam menciptakan ketahanan siber yang adaptif dan menyeluruh.

II.4 *Anti-Spyware*

Sistem *anti-spyware* modern menghadapi tantangan signifikan dalam lanskap keamanan siber yang terus berevolusi, beralih dari malware tradisional menuju serangan stealth yang canggih. Menurut Sudhakar dan Kumar (2020), fileless malware beroperasi langsung di memori, menjadikannya sulit dideteksi oleh antivirus tradisional yang berbasis tanda tangan. Spyware merupakan ancaman yang beroperasi tersembunyi, dirancang untuk memantau dan mengumpulkan informasi pengguna secara rahasia, dan dikategorikan sebagai ancaman cyber espionage. Ancaman ini terus berkembang: vektor serangan meluas hingga menyasar fitur modern seperti asisten suara smartphone dan spyware disebarkan melalui injeksi pada aplikasi palsu. Selanjutnya, Ainslie dkk. (2023) menekankan bahwa keamanan siber bukan hanya isu teknis, tetapi tantangan strategis yang memengaruhi pengambilan keputusan di tingkat organisasi, sehingga pentingnya *Cyber Threat Intelligence (CTI)* semakin

krusial.

Kelemahan deteksi pada sistem *anti-spyware* timbul dari metode evasi canggih yang memanfaatkan celah arsitektur keamanan. E. dkk. (2023) menjelaskan bahwa spyware menggunakan teknik *packing* dan *obfuscation* untuk mengubah tanda tangan digitalnya, secara efektif menghindari deteksi berbasis tanda tangan. Bahkan, Koutsokostas dan Patsakis (2021) menunjukkan bahwa malware dapat memanfaatkan *obfuscation bytecode* Python karena kegagalan alat keamanan dalam memprosesnya. Lebih lanjut, ancaman *fileless* mengandalkan skrip bawaan sistem operasi, terutama PowerShell, untuk evasi, karena eksekusi langsung di memori (*in-memory*) menghindari deteksi berbasis file tradisional. Kareem (2024) menggarisbawahi bahwa spyware tingkat lanjut menunjukkan kapabilitas *zero-click* dan memerlukan mekanisme deteksi yang jauh lebih kompleks.

Mengingat kompleksitas ancaman yang ada, strategi *anti-spyware* harus beralih dari deteksi pasif menuju pendekatan yang lebih proaktif dan holistik. Dalam konteks pengembangan alat offensive security, Kerkour (2021) memilih bahasa pemrograman modern seperti Rust sebagai pilihan unggulan karena unggul dalam menciptakan tool yang tangguh dan sulit dilacak. Koutsokostas dan Patsakis (2021) serta Elghaly dan M. (2024) sama-sama menekankan bahwa inti dari penguatan *anti-spyware* adalah adopsi pendekatan deteksi berbasis perilaku (*behavior-based*) untuk mengenali anomali aktivitas sistem, alih-alih hanya menandai *file*. Selain itu, Domínguez-Dorado dkk. (2024) menyoroti bahwa manajemen keamanan siber harus berfokus pada integrasi aspek teknologi, tata kelola, dan budaya organisasi, didukung oleh kerangka kerja holistik seperti FLECO.

II.4.1 *Endpoint Detection and Response (EDR)*

Endpoint Detection and Response (EDR) merupakan solusi keamanan siber yang sangat penting dalam strategi pertahanan modern, dirancang untuk mengatasi keterbatasan *anti-malware* konvensional yang terlalu mengandalkan deteksi berbasis tanda tangan (*signature-based*). EDR berfokus pada pendekatan deteksi berbasis perilaku (*behavior-based*) dan analisis aktivitas, menjadikannya garis pertahanan krusial terhadap ancaman *stealth* dan *fileless malware*. Kebutuhan akan EDR meningkat karena *fileless malware* beroperasi langsung di memori (*in-memory*), sering memanfaatkan *script* bawaan sistem operasi seperti PowerShell untuk menjalankan kode berbahaya, yang secara efektif menghindari deteksi berbasis *file*. E. dkk. (2023) menguatkan bahwa EDR harus mengatasi teknik *obfuscation* dan *packing spyware* yang efektif melawan mesin *anti-malware* lama. EDR menjadi vital da-

lam menghadapi ancaman canggih seperti *spyware* dengan kapabilitas *zero-click*, dan Ainslie dkk. (2023) menekankan bahwa EDR merupakan komponen kunci dalam pengambilan keputusan keamanan strategis. Oleh karena itu, EDR berfungsi untuk melengkapi *anti-malware* tradisional, menyediakan kemampuan analisis perilaku dan forensik, serta merupakan komponen inti dalam penguatan ketahanan siber organisasi secara keseluruhan.

II.4.2 *Signatured-Based Anti-Virus*

Deteksi berbasis tanda tangan (*signature-based*) merupakan metode dasar dan tradisional yang digunakan oleh sebagian besar produk *anti-virus (AV)* sejak awal kemunculannya. Metode ini bekerja dengan membandingkan *hash* kriptografi atau pola urutan byte unik (disebut sebagai tanda tangan atau *signature*) dari *file* yang di*scan* dengan basis data ekstensif yang berisi *signature malware* yang sudah dikenal. Jika terjadi kecocokan (*match*), *file* tersebut diklasifikasikan sebagai *malware* dan akan dikarantina atau dihapus. Meskipun metode ini sangat cepat dan memiliki akurasi 100% dalam mendeteksi *malware* yang *signature*-nya telah terdaftar, kelemahan mendasarnya adalah sifatnya yang reaktif; *anti-virus* harus memiliki signature terlebih dahulu, yang berarti metode ini tidak efektif terhadap ancaman baru (*zero-day threats*). Lebih lanjut, *signature-based detection* mudah dihindari oleh *malware* modern melalui teknik penyamaran seperti enkripsi, *packing*, dan *obfuscation*, yang mengubah *signature file* tanpa mengubah fungsionalitas intinya. Kegagalan ini memaksa sistem keamanan untuk bergeser menuju pendekatan yang lebih proaktif, seperti analisis perilaku dan pembelajaran mesin.

II.4.3 *Behavior-Based Anti-Virus*

Deteksi berbasis perilaku (*behavior-based*) adalah pendekatan yang lebih proaktif dan modern, dikembangkan untuk mengatasi kelemahan utama metode berbasis tanda tangan (*signature-based*). Metode ini tidak bergantung pada *signature* yang sudah dikenal, melainkan memantau dan menganalisis tindakan atau pola perilaku yang mencurigakan yang dilakukan oleh suatu program saat *runtime*. Program keamanan akan memonitor serangkaian aktivitas sistem, seperti upaya untuk memodifikasi *registry sistem*, mencoba mengakses dan mengenkripsi *file* sensitif, atau meluncurkan proses sistem yang sah (misalnya, PowerShell atau WMI) dengan parameter yang tidak biasa. Jika suatu program menunjukkan urutan tindakan yang menyerupai *malware* (seperti *fileless execution* atau *persistence*), program tersebut akan ditandai atau dihentikan. Meskipun deteksi berbasis perilaku efektif dalam mengidentifikasi *malware* baru dan *fileless malware*, metode ini juga me-

miliki tantangan. *Malware* canggih sering kali dirancang untuk meniru perilaku proses yang sah (*living-off-the-land*) atau menunda eksekusi berbahaya, sehingga *behavior-based detection* rentan terhadap *false positives* (program sah yang salah dideteksi) atau *evasion* yang kompleks. Oleh karena itu, pendekatan ini sering diintegrasikan dengan *machine learning* dan analisis *in-memory* untuk meningkatkan akurasi dan mengurangi *false positives*.

II.4.4 *Entropy-Based Anti-Virus*

Entropy adalah konsep yang digunakan dalam teori informasi untuk mengukur tingkat keacakan atau ketidakpastian data yang terkandung di dalam sebuah *file* atau segmen kode. Nilai *entropy* biasanya berkisar dari 0 hingga 8; di mana nilai yang mendekati 8 menunjukkan data yang sangat acak dan tidak terstruktur, mendekati *noise* murni. Nilai *entropy* yang tinggi ini merupakan indikator penting yang digunakan oleh *anti-spyware* dan alat analisis statis (*static analysis*) untuk menduga adanya teknik *evasion* yang canggih, terutama *packing* dan enkripsi. Ketika *payload malware* dienkripsi atau dikompresi oleh *packer*, data asli yang terstruktur diubah menjadi data yang tampak acak. Oleh karena itu, *anti-spyware* menggunakan ambang batas *entropy* yang tinggi (misalnya, di atas 7.0) sebagai bendera merah (*red flag*) untuk menandai *file* sebagai mencurigakan (*suspicious*), yang mengindikasikan bahwa sebagian besar isi file tersebut tidak dapat dibaca atau dianalisis secara statis. Meskipun entropy tidak dapat memastikan secara pasti bahwa *file* tersebut adalah *spyware*, ia sangat efektif dalam mengidentifikasi adanya upaya penyembunyian (*concealment*) yang merupakan karakteristik utama dari *Packer Spyware Mode Stealth*.

II.5 *Malware*

Malware merupakan salah satu ancaman utama dalam dunia keamanan siber yang terus berevolusi dari generasi ke generasi. Secara umum, *malware* adalah perangkat lunak berbahaya yang dirancang untuk menyusup, merusak, atau mencuri data dari sistem komputer tanpa sepengetahuan pengguna. Sudhakar dan Kumar (2020) menjelaskan bahwa evolusi malware telah bergeser dari bentuk tradisional berbasis *file* menuju *fileless malware* yang beroperasi sepenuhnya di memori. Jenis *malware* ini tidak meninggalkan jejak *file* di sistem, sehingga sulit dideteksi oleh antivirus berbasis tanda tangan. *Fileless malware* sering memanfaatkan komponen sah dari sistem operasi, seperti *Windows Management Instrumentation (WMI)* dan PowerShell, untuk melancarkan serangan tanpa menulis *file* berbahaya ke *disk*. Teknik

ini memungkinkan pelaku untuk melakukan aksi seperti *reconnaissance*, pencurian data, dan persistensi tanpa terdeteksi oleh solusi keamanan konvensional.

E. dkk. (2023) menambahkan bahwa dalam "permainan kucing dan tikus" antara pembuat *malware* dan pembuat *antivirus*, berbagai teknik penghindaran deteksi terus berkembang. Teknik-teknik seperti *code obfuscation, polymorphism, packing*, dan *process injection* digunakan untuk mengubah struktur dan perilaku kode agar tidak mudah dikenali. Studi mereka menunjukkan bahwa dari 16 sampel *malware* yang diujikan dengan tujuh teknik penghindaran klasik, hanya sebagian kecil antivirus yang mampu mendeteksi lebih dari separuh variasi *malware* tersebut. Hal ini menegaskan bahwa bahkan *malware* "lama" dengan trik penyamaran baru masih mampu menembus sistem deteksi modern. Lebih jauh lagi, peneliti juga menemukan bahwa penggunaan model pembelajaran mesin (ML) untuk deteksi *malware* masih rentan terhadap serangan *adversarial*, di mana pembuat *malware* dapat menghasilkan varian baru yang tampak seperti program sah.

Koutsokostas dan Patsakis (2021) berfokus pada pengembangan *malware stealth* berbasis Python yang mampu menghindari deteksi tanpa menggunakan *obfuscation*. Mereka menemukan bahwa keterbatasan pada mesin deteksi statis, seperti VirusTotal dan sandbox analisis dinamis, dapat dimanfaatkan untuk menciptakan *malware* yang "bersih" dari hasil pemindaian puluhan antivirus. Studi tersebut mengungkapkan bahwa PyInstaller—alat populer untuk membungkus program Python menjadi *executable* dapat dimodifikasi agar menghasilkan *malware* yang tidak terdeteksi karena kelemahan inheren dalam cara antivirus memproses bytecode Python. Selain itu, mereka menemukan bahwa sandbox publik sering kali gagal mendeteksi *malware* yang menunda eksekusi, mendeteksi lingkungan virtual, atau memeriksa artefak sistem sebelum beraksi.

Berdasarkan literatur tersebut, tren utama dalam penelitian *malware* modern menyoroti bahwa ancaman kini tidak hanya berasal dari varian baru, tetapi dari kemampuan *malware* untuk beradaptasi terhadap mekanisme pertahanan yang ada. Dengan kombinasi teknik *living-off-the-land*, penghindaran berbasis memori, dan eksploitasi terhadap celah dalam sistem analisis otomatis, *malware* modern semakin sulit diidentifikasi. Oleh karena itu, studi-studi ini menegaskan perlunya pendekatan deteksi berbasis perilaku dan kecerdasan buatan yang mampu mengenali pola aktivitas abnormal alih-alih bergantung semata pada tanda tangan statis. Dengan demikian, penelitian mengenai *malware* tidak hanya penting untuk memahami sifat serangan, tetapi juga menjadi dasar dalam merancang sistem pertahanan yang lebih adaptif

dan tangguh terhadap ancaman siber generasi baru.

II.5.1 *Fileless Malware*

Fileless execution adalah teknik serangan siber canggih di mana kode berbahaya dijalankan secara langsung di dalam memori sistem (RAM) tanpa perlu menulis atau menyimpan *file* yang dapat dideteksi ke *disk* (*hard drive*). Metode ini sering dimanfaatkan oleh *fileless malware* atau *spyware* untuk menghindari deteksi berbasis tanda tangan (*signature-based*) dan forensik digital tradisional yang berfokus pada analisis *file system*. *Fileless execution* umumnya dicapai dengan mengeksploitasi alat (*utility*) atau fitur bawaan sistem operasi yang sah, seperti PowerShell, *Windows Management Instrumentation (WMI)*, atau dengan menyuntikkan kode langsung ke proses yang sudah ada dan terpercaya (*process injection*). Karena tidak ada *file* berbahaya yang disimpan, serangan ini sangat sulit dilacak dan dideteksi oleh *anti-virus* konvensional, memaksa sistem keamanan untuk beralih ke deteksi berbasis perilaku dan analisis memori.

II.5.2 *Obfuscation* pada *Malware*

Obfuscation adalah teknik penyembunyian (*concealment*) yang digunakan untuk memanipulasi kode *malware* atau *payload* agar menjadi sulit untuk dipahami, dianalisis, atau dideteksi oleh alat keamanan statis maupun dinamis. Menurut E. dkk. (2023) *code obfuscation* adalah teknik klasik yang terbukti masih sangat efektif untuk menghindari deteksi *anti-virus* modern. Tujuan utamanya adalah untuk mengubah tanda tangan (*signature*) kode, *string*, dan alur program, sehingga *anti-virus* berbasis *signature* kesulitan mencocokkannya dengan basis data yang ada.

Dalam implementasinya, *obfuscation* sering dilakukan melalui berbagai metode. Misalnya, dalam pengembangan *malware* berbasis script seperti Python, *obfuscation* dapat terjadi ketika kode sumber dikemas menjadi bentuk *bytecode* terkompilasi menggunakan alat seperti PyInstaller (banyak *anti-virus* gagal menganalisis *bytecode* ini, sehingga menghasilkan *false negative*). Selain itu, *obfuscation* diterapkan pada *Dropper* untuk menyamarkan skrip yang dieksekusi melalui *utility* sistem seperti PowerShell yang secara langsung menargetkan kelemahan *script monitoring* pada EDR dan AV. Kesuksesan *obfuscation* memaksa peneliti dan anti-malware untuk beralih dari analisis statis ke analisis perilaku (*behavior-based detection*) dan teknik *deobfuscation* yang mahal.

II.6 *Spyware*

Spyware merupakan salah satu bentuk *malware* yang dirancang untuk memantau, mengumpulkan, dan mengirimkan informasi pengguna tanpa izin atau kesadaran mereka. Perangkat lunak ini biasanya berjalan secara tersembunyi di latar belakang sistem dan dapat merekam aktivitas pengguna, seperti penekanan tombol (*keylogging*), riwayat peramban, data *login*, serta *file* sensitif. Dalam literatur keamanan siber, *spyware* sering dikategorikan sebagai ancaman yang bersifat *stealth*, karena kemampuannya untuk beroperasi tanpa menimbulkan indikasi mencolok bagi pengguna maupun sistem keamanan. Menurut Koutsokostas dan Patsakis (2021), kemampuan *stealth* seperti ini muncul karena *malware* modern, termasuk *spyware*, memanfaatkan teknik penghindaran analisis dan deteksi baik dalam bentuk statis maupun dinamis. Mereka menunjukkan bahwa banyak *antivirus* gagal mengenali kode berbahaya yang dikemas menggunakan alat seperti PyInstaller karena keterbatasan dalam menganalisis *bytecode Python*. Hal ini menyebabkan sebagian besar *multi-engine scanner*, termasuk VirusTotal, dapat memberikan hasil "bersih" terhadap *file* yang sebenarnya mengandung komponen *spyware*.

E. dkk. (2023) dalam studi "*Bypassing Antivirus Detection: Old-school Malware, New Tricks*" menguatkan temuan tersebut dengan menyoroti bagaimana teknik klasik seperti *code obfuscation, packing*, dan *process injection* masih sangat efektif untuk menghindari deteksi *antivirus* modern. Mereka menguji berbagai varian *malware*, termasuk *spyware*, terhadap beberapa produk *antivirus* dan menemukan bahwa sebagian besar sistem deteksi hanya mampu mengenali sebagian kecil varian yang dimodifikasi. Temuan ini menunjukkan bahwa banyak mesin *antivirus* masih mengandalkan pencocokan tanda tangan (*signature-based detection*), yang tidak mampu mengidentifikasi pola perilaku baru dari *spyware* yang berevolusi. Selain itu, metode penghindaran berbasis lingkungan—seperti deteksi sandbox atau penundaan eksekusi (*delayed execution*) membuat spyware semakin sulit teridentifikasi melalui analisis dinamis tradisional.

Dari sisi karakteristik perilaku, *spyware* modern cenderung memanfaatkan teknik *living-off-the-land*, yaitu memanfaatkan fungsi atau layanan sah dari sistem operasi seperti PowerShell, WMI, atau API Windows untuk melaksanakan aksinya tanpa mengunduh *file* berbahaya tambahan. Pendekatan ini menjadikan *spyware* semakin sulit dilacak, karena aktivitasnya tampak seperti proses sistem yang normal. Koutsokostas dan Patsakis (2021) menegaskan bahwa pola serangan semacam ini tidak hanya menunjukkan kelemahan sistem deteksi *antivirus*, tetapi juga menyoroti per-

lunya pendekatan baru berbasis perilaku (*behavior-based detection*) yang mampu mengenali anomali aktivitas sistem, bukan sekadar menandai *file* berbahaya.

Secara keseluruhan, *spyware* merupakan evolusi dari malware tradisional menuju ancaman yang lebih canggih, tersembunyi, dan adaptif. Dengan kemampuan memanfaatkan celah pada sistem deteksi statis maupun dinamis, *spyware* modern menjadi tantangan utama dalam bidang keamanan siber. Oleh karena itu, penelitian dan pengembangan sistem pertahanan di masa depan perlu berfokus pada integrasi antara analisis perilaku, pembelajaran mesin, dan *threat intelligence* untuk mendeteksi aktivitas mencurigakan secara proaktif. Pendekatan ini diharapkan dapat menutup celah yang selama ini dimanfaatkan oleh *spyware* untuk beroperasi tanpa terdeteksi di berbagai *platform*, baik *desktop* maupun *mobile*.

II.6.1 *Spyware Mode Stealth*

Spyware mode stealth merujuk pada evolusi ancaman *spyware* yang secara khusus dirancang untuk beroperasi secara tersembunyi dan menghindari deteksi sistem keamanan siber. Ancaman ini dikategorikan dalam lingkup yang lebih luas yaitu *cyber espionage* dan merupakan evolusi dari malware tradisional.

Kemampuan *spyware* untuk beroperasi secara *stealth* sangat bergantung pada eksploitasi kelemahan dalam mekanisme deteksi. E. dkk. (2023) menjelaskan bahwa *spyware* secara aktif menggunakan teknik *packing* dan *obfuscation* untuk mengubah tanda tangan digitalnya, efektif menghindari deteksi berbasis tanda tangan. Sudhakar dan Kumar (2020) dan Elghaly dan M. (2024) sama-sama menyoroti bahwa ancaman *fileless* merupakan mekanisme *stealth* kunci, di mana *spyware* beroperasi langsung di memori, sering memanfaatkan PowerShell untuk menjalankan kode berbahaya, menghindari deteksi berbasis *file*. Koutsokostas dan Patsakis (2021) menambahkan bahwa kegagalan alat keamanan dalam memproses *bytecode* Python juga dieksploitasi untuk menyamarkan *script* berbahaya. EDR menghadapi tantangan besar karena harus memantau proses sistem yang sah ini untuk mengidentifikasi aktivitas mencurigakan.

II.6.2 *Packer Spyware Mode Stealth*

Packer merupakan alat atau teknik perangkat lunak yang berfungsi sebagai lapisan perlindungan awal dengan mengemas (*wrap*) *payload malware* agar menjadi sulit dideteksi dan dianalisis oleh sistem keamanan seperti *anti-virus (AV)* maupun EDR (*Endpoint Detection & Response*). Tujuan utama *packer* adalah mencapai *evasion*

dengan mengubah *signature file* sebelum eksekusi. *Packer* mencapai tujuan ini dengan melakukan beberapa proses, termasuk kompresi (*compression*) untuk mengurangi ukuran *file*, enkripsi untuk mengacak kode dan data, serta *obfuscation* untuk menyamarkan struktur kode agar tidak mudah dicocokkan dengan basis data *signature AV*. Keberhasilan *packer* dalam mengubah *signature file* secara efektif menjadikannya taktik utama untuk lolos pada tahap pemeriksaan statis (*pre-execution*).

Dalam konteks *spyware mode stealth*, *packer* sering diimplementasikan sebagai *Loader single-stage* yang bertanggung jawab untuk menjalankan *runtime unpacking*. Pada *runtime* di sistem target, sebuah stub kecil di dalam *file* yang sudah di-*pack* akan mendekode atau mendekripsi *payload* asli secara langsung di memori (RAM). Dengan melakukan proses *unpacking* dan eksekusi di memori tanpa menulis *payload* yang sudah didekripsi ke *disk*, *packer* mendukung teknik *fileless execution*. Hal ini secara signifikan meningkatkan stealth karena menghindari analisis *file system* dan *signature-based detection*. Secara keseluruhan, *Packer Spyware Mode Stealth* bertujuan mengeksploitasi celah *anti-malware* ganda: pertama, dengan memanipulasi *signature file*, dan kedua, dengan menjalankan kode berbahaya hanya di memori (*in-memory*) melalui alat sistem yang sah seperti PowerShell, menjadikannya sangat sulit dilacak oleh *anti-virus* konvensional.

II.6.3 *Payload* pada *Spyware*

Payload merujuk pada komponen inti atau muatan dari suatu *malware* yang membawa fungsi berbahaya yang sebenarnya, yaitu tujuan akhir dari serangan. *Payload* dipandang sebagai *mission* atau tugas utama yang harus diselesaikan setelah *malware* berhasil menginfiltrasi sistem. Dalam konteks *spyware*, *payload* memiliki peran spesifik untuk pengumpulan intelijen digital. Fungsi utamanya meliputi *Data Collection* dan Eksfiltrasi. Sebelum dieksekusi, *payload* sering kali disembunyikan dan dienkripsi oleh lapisan *packer*. Dalam skenario *fileless execution*, *payload* sering dijalankan di memori (*in-memory*) melalui *utility* sistem yang sah, seperti PowerShell atau WMI, sehingga sulit dideteksi oleh *anti-virus* konvensional. Setelah pertahanan awal dilewati, *payload* akan didekripsi dan dieksekusi oleh *loader packer* langsung di memori untuk memulai *mission* tanpa jejak *file* di *disk*.

II.7 Studi Sebelumnya

Studi sebelumnya diperlukan untuk memahami perkembangan penelitian terkait mekanisme penyembunyian data (*packer*), teknik eksekusi tersembunyi (*stealth*), serta berbagai metode evasi dan deteksi yang digunakan dalam keamanan siber. Kajian

ini memberikan gambaran mengenai pendekatan, metode, serta keterbatasan penelitian terdahulu yang menjadi dasar dalam merumuskan ruang lingkup dan kontribusi penelitian ini.

Tabel II.1 Studi Sebelumnya (Packer Spyware dan Mekanisme Evasi)

No	Judul Penelitian (Penulis)	Metode	Hasil Utama	Keterbatasan
1	*Python and Malware: Developing Stealth and Evasive Malware Without Obfuscation* (Koutsokostas dan Patsakis (2021))	Perancangan sistem dan eksperimen	Mengembangkan *malware* Python *stealth* yang menghindari deteksi dengan mengeksploitasi kelemahan *multi-engine scanners* dalam memproses *bytecode* Python, menunjukkan efektivitas *packer stealth*.	Fokus utama pada evasi *static analysis* dan keterbatasan *bytecode* Python; kurang membahas teknik *in-memory* dan *behavioral evasion* yang lebih luas.
2	*Bypassing Antivirus Detection: Old-School Malware, New Tricks* (E. dkk. (2023))	Eksperimen komparatif dan analisis teknis	Menunjukkan bahwa teknik lama seperti *packing* dan *obfuscation* tetap efektif melawan AV/EDR modern; hampir separuh mesin yang diuji gagal mendeteksi varian *malware* yang disamarkan.	Fokus pada kerentanan metode *signature-based* umum; tidak mengusulkan solusi arsitektur deteksi baru.
3	*Stealth in Plain Sight: The Hidden Threat of PowerShell Fileless Malware...* (Elghaly dan M. (2024))	Analisis teknis dan eksperimen bypass	Mengkaji bagaimana *malware fileless* memanfaatkan *PowerShell* untuk eksekusi kode langsung di memori, secara efektif menghindari EDR dan AV modern.	Fokus pada lingkungan PowerShell; kurang membahas mekanisme *packer* kustom dan *bytecode evasion*.
4	*An Emerging Threat: Fileless Malware – A Survey and Research Challenges* (Sudhakar dan Kumar (2020))	Studi literatur	Menjelaskan teknik penyembunyian dan eksfiltrasi *fileless* yang dieksploitasi oleh *spyware*, termasuk penggunaan PowerShell, *registry abuse*, dan *memory-resident payload*.	Bersifat survei dan komprehensif; tidak mencakup implementasi uji coba *packer* kustom atau pengukuran kapabilitas deteksi.

BAB III

ANALISIS MASALAH

III.1 Analisis Kondisi Saat Ini

Kondisi keamanan siber saat ini ditandai dengan evolusi ancaman yang signifikan, bergerak dari *malware* tradisional berbasis *file* menuju serangan yang lebih canggih dan tersembunyi, seperti *fileless malware* dan *spyware mode stealth*. Evolusi ini menciptakan celah kritis dalam kemampuan deteksi sistem *anti-spyware (AS)* dan *Endpoint Detection and Response (EDR)* yang masih berpegangan pada mekanisme pertahanan konvensional.

Secara konseptual, serangan *spyware* modern yang berfokus pada *Initial Access* dan *Packer* melibatkan beberapa komponen utama: Target (sistem yang diserang), *Packer/Dropper* (Artefak *spyware* yang disamarkan), *Anti-Malware/EDR* (Sistem pertahanan), dan *Server* (Pengumpul informasi, seperti yang digambarkan dalam alur *General Flow*).

III.1.1 Masalah Kerentanan dan Ketertinggalan *Anti-Malware*onvensional

Kondisi keamanan siber saat ini ditandai dengan evolusi ancaman yang signifikan, bergerak dari *malware* tradisional berbasis *file* 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 (AV)* dan *Endpoint Detection and Response (EDR)* yang masih berpegangan pada mekanisme pertahanan konvensional.

III.1.2 Keterbatasan Deteksi Berbasis Tanda Tangan (*Signature-Based*)

Anti-malware tradisional masih sangat mengandalkan pencocokan tanda tangan (*signature-based detection*). *Spyware mode stealth* menggunakan teknik *Packer* (seperti yang dijelaskan dalam konsep arsitektur serangan) untuk melakukan kompresi, enkripsi,

dan *obfuscation* pada *payload*. Teknik ini secara efektif mengubah tanda tangan digital (*signature*) *spyware*, sehingga membuatnya lolos dari deteksi *signature-based* karena dianggap sebagai *file* baru atau tidak dikenal.

III.1.3 Gagal Menganalisis Code *Fileless Execution*

Situasi ini diperparah dengan temuan bahwa banyak alat keamanan gagal tidak dapat mendeteksi *fileless malware*. *Fileless malware* memanfaatkan komponen sah dari sistem operasi (seperti PowerShell dan WMI) untuk menjalankan kode berbahaya langsung di memori tanpa menulis *file* ke disk, sehingga sulit dideteksi oleh *antivirus* konvensional.

III.1.4 Efektivitas Teknik Evasi Sederhana

Meskipun serangan semakin canggih, penelitian menunjukkan bahwa metode evasi yang relatif sederhana, seperti enkripsi, injeksi proses, dan penambahan data sampah (*junk data*) ke file eksekusi, terbukti sangat efektif dalam menghindari deteksi. Bahkan, dalam sebuah studi E. dkk. (2023), hampir separuh dari 12 mesin *antivirus* yang diuji hanya mampu mendeteksi kurang dari setengah varian *malware* yang disamarkan.

III.1.5 Kurangnya Deteksi Proaktif Terhadap Arsitektur Serangan

Kurangnya deteksi yang efektif oleh solusi keamanan menciptakan celah besar yang dieksploitasi oleh *Advanced Persistent Threats (APTs)*. Sistem pertahanan saat ini terlalu reaktif, berfokus pada *file* yang sudah terinstal, bukan pada deteksi perilaku evasif dari *Packer* di fase *Initial Access* dan *Establish Foothold*.

III.1.6 Gap Analysis Celah Deteksi *Packer Spyware Mode Stealth*

Berdasarkan kondisi saat ini dan studi literatur, sebuah Analisis Kesenjangan (*Gap Analysis*) dirumuskan untuk menyoroti perbedaan antara kondisi ideal deteksi dan realitas sistem *anti-spyware* saat ini.

Tabel III.1 Kebutuhan fungsional (Functional Requirements)

ID	*Critical to Quality(CTQ)*	Kondisi Saat ini	*Gap*	Kondisi Ideal
CTQ-01	Deteksi *Packer* pada *Initial Access*	Deteksi berfokus pada *signature file* yang mudah di-*bypass* oleh teknik enkripsi dan *obfuscation*	Celah *Signature-Evasion*	*Packer* harus menggunakan enkripsi unik dan *obfuscation* untuk *payload*.
CTQ-02	Gagal menganalisis kode *payload* ter-*obfuscated*.	Gagal menganalisis skrip yang ter-*obfuscated*	Celah *Obfuscation*	Prototipe harus dirancang dengan bahasa yang menghasilkan *binary* resisten (*low-level Rust*).
CTQ-03	Akurasi Deteksi *Fileless Execution*	Deteksi rendah karena *payload* berjalan langsung di memori (via PowerShell/WMIC)	Celah *In-Memory* dan *Behavioral*	*Packer* harus mampu mengintegrasikan *loader in-memory* (*single-stage*) yang *stealth*.

III.2 Analisis Kebutuhan

Berdasarkan *gap analysis* (CTQ) di atas, diperlukan suatu artefak uji yang mampu mensimulasikan serangan *spyware mode stealth* secara terkontrol. Oleh karena itu, pada bagian berikut disusun kebutuhan fungsional dan nonfungsional dari prototipe *packer spyware*.

III.2.1 Kebutuhan Fungsional

Kebutuhan fungsional mendefinisikan kapabilitas yang harus dimiliki oleh *Packer Spyware* untuk menguji celah keamanan (CTQ).

Tabel III.2 Kebutuhan fungsional (Functional Requirements)

ID	Kebutuhan	CTQ Terkait
FR-01	Prototipe harus mampu mengenkripsi *payload* dan menyembunyikan *signature* asli.	CTQ-01
FR-02	Prototipe harus mampu melakukan pengemasan (*packing*) dan *obfuscation* tingkat tinggi pada *binary* akhir.	CTQ-02
FR-03	Prototipe harus mampu menjalankan *payload* secara *fileless* melalui PowerShell.	CTQ-03
FR-04	Prototipe harus memiliki kemampuan *Initial Access* dan *Persistence* dasar.	CTQ-01, CTQ-03
FR-05	Prototipe harus mampu mengirimkan sinyal keberhasilan *Initial Access* ke *Server* pengumpul data.	CTQ-01

III.2.2 Kebutuhan Nonfungsional

Kebutuhan nonfungsional berfokus pada bagaimana sistem pengujian harus bekerja untuk memastikan hasil yang valid dan andal.

Tabel III.3 Kebutuhan fungsional (Functional Requirements)

ID	Kebutuhan	CTQ Terkait	
NFR-01	*Executable* yang dihasilkan harus kecil dan *reliable* di Windows 11.	Keandalan & Validitas	CTQ-01, CTQ-02, CTQ-03
NFR-02	Lingkungan pengujian harus mencakup setidaknya empat produk *anti-malware/EDR* komersial yang berbeda.	Validitas komparatif	CTQ-01, CTQ-02, CTQ-03
NFR-03	Implementasi harus menggunakan *code safety (low-level Rust)* untuk memastikan *debugging* yang efisien.	Audibilitas	CTQ-01, CTQ-03

III.3 Analisis Pemilihan Solusi

Setelah masalah dianalisis, langkah selanjutnya adalah melaksanakan evaluasi berbagai solusi yang dapat mewujudkan tujuan penelitian. Analisis ini mencakup penilaian pendekatan teknis dan konseptual yang paling ideal sesuai dengan kebutuhan sistem dan kriteria desain yang telah ditetapkan. Tujuan utama dari penelitian ini adalah merancang dan mengimplementasikan *Packer Spyware Mode Stealth* untuk menguji kapabilitas deteksi *anti-malware*, sehingga dapat memvalidasi teknik *evasion* dan mengungkap kelemahan sistem keamanan konvensional.

III.3.1 Alternatif Solusi

Dalam upaya mencari solusi yang paling optimal untuk mengatasi permasalahan yang teridentifikasi, subbab ini akan menguraikan berbagai konsep implementasi alternatif untuk setiap Kebutuhan Fungsional (FR) dan Kebutuhan Non Fungsional (NFR) modul *packer spyware mode stealth*. Setiap alternatif ini akan dijelaskan secara ringkas mengenai pendekatannya dan bagaimana ia berpotensi memenuhi kebutuhan yang bersangkutan.

Tabel III.4 Kebutuhan fungsional (Functional Requirements)

Kode	**Solusi Bahasa pemrograman**	**Pendekatan Utama *(Evasion)***
S-01	Rust	Bahasa sistem yang menghasilkan *binary* statis, unggul dalam kontrol memori dan *low-level manipulation*.
S-02	C/C++	Bahasa sistem tradisional yang menawarkan kontrol penuh atas API Windows dan *memory injection*.
S-03	Python (PyInstaller)	Bahasa *scripting*. Mengandalkan *obfuscation bytecode* dan *packer wrapper*.

III.3.2 Rust

Rust adalah bahasa pemrograman sistem yang akan digunakan untuk mengembangkan *Packer*. Bahasa ini dipilih karena mampu menghasilkan *binary* yang sangat efisien dan terkompilasi menjadi *native code*. Kontrol tingkat rendah yang disediakan Rust memungkinkan implementasi manipulasi memori dan panggilan API Windows

secara langsung, yang sangat penting untuk teknik *fileless execution* (FR-A3). *Packer* yang dibangun dengan Rust dapat mengenkripsi *payload* (FR-A1) dan memiliki resistensi tinggi terhadap analisis statis (*reverse engineering*), karena minimnya jejak *runtime* yang mudah dideteksi (Potensi *Stealth*).

III.3.3 C/C++

C/C++ adalah bahasa pemrograman sistem tradisional yang merupakan standar umum untuk pengembangan alat *low-level*. Penggunaan bahasa ini memungkinkan pengembang untuk memiliki kontrol penuh atas manajemen memori dan akses langsung ke API Windows, yang esensial untuk implementasi *memory injection* dan *fileless execution* (FR-A3). *Packer* yang dikembangkan dengan C/C++ dapat mengkompilasi *payload* menjadi *binary native* yang kuat, menjadikannya alternatif yang efektif untuk menyamarkan *payload* dan melakukan enkripsi tingkat lanjut (FR-A1).

III.3.4 Python (PyInstaller)

Python adalah bahasa *scripting* tingkat tinggi yang menawarkan kemudahan dan kecepatan dalam pengembangan *payload* dan logika enkripsi sederhana (FR-A1). Ketika dikombinasikan dengan alat *packer* seperti PyInstaller, *script* Python dapat dibungkus menjadi *executable* Windows. Solusi ini memanfaatkan *obfuscation bytecode* dan *packer wrapper* sebagai lapisan pertahanan awal terhadap deteksi *signature-based*. Namun, solusi ini bergantung pada mekanisme *wrapper* yang seringkali kurang stealth dibandingkan kompilasi *low-level* murni.

III.3.5 Analisis Penentuan Solusi

Untuk memastikan solusi yang dihasilkan dapat menyelesaikan masalah yang telah diidentifikasi sebelumnya, kriteria desain ditetapkan dengan mempertimbangkan hasil identifikasi masalah dan tujuan solusi. Kriteria ini berfungsi sebagai acuan dalam merancang, mengembangkan, dan mengevaluasi solusi secara sistematis. Tabel III.5 akan menjelaskan kriteria-kriteria desain tersebut yang akan menjadi dasar dalam menilai keberhasilan artefak yang dikembangkan dalam penelitian ini.

Tabel III.5 Kriteria Desain Packer Spyware

Kode	Kriteria Desain	Keterkaitan dengan Evasion & Non-Functional Requirement
KD-1	Dukungan Enkripsi/Packing	Kemampuan *native* bahasa dalam mengimplementasikan enkripsi *payload* dan *packing* kode sumber secara ringkas, mendukung teknik *code obfuscation* dan *payload concealment*.
KD-2	Kontrol *Low-Level* & *Fileless*	Mendukung pemanggilan Windows API, manipulasi memori, dan eksekusi *fileless* untuk meningkatkan *stealth* serta mengurangi artefak forensik.
KD-3	Resistensi Analisis Statis	Meningkatkan kesulitan *reverse engineering* pada *binary*; bahasa yang kuat membantu keamanan, integritas, dan *reliability payload* terhadap analisis statis.
KD-4	Potensi Stealth & Footprint	Ukuran *binary* kecil dan minim dependensi *runtime* eksternal sehingga mengurangi jejak deteksi AV/EDR pada pengujian *stealth*.
KD-5	Kompleksitas Implementasi & Safety	Effort waktu dan tingkat kesulitan implementasi (termasuk keamanan memori dan *debugging*) berpengaruh langsung pada efisiensi pengembangan artefak.
KD-6	Kompatibilitas Sistem Target	Kemampuan menghasilkan *executable* yang stabil dan kompatibel pada Windows, memastikan *reliability* selama pengujian *evasion*.

Tabel menyajikan rangkuman hasil evaluasi dari setiap alternatif solusi berdasarkan kriteria desain yang telah ditentukan, menggunakan skala penilaian 1 hingga 9. Skor 1 menunjukkan bahwa suatu alternatif solusi memiliki tingkat kesesuaian yang sangat rendah terhadap kriteria desain. Skor 9 mencerminkan bahwa solusi tersebut sangat memenuhi kriteria yang ditetapkan dan memiliki potensi tinggi untuk menyelesaikan permasalahan secara efektif.

BAB IV

DESAIN KONSEP SOLUSI

Ilustrasikan desain konsep solusi dalam bentuk model konseptual dan penjelasan secara ringkas, beserta perbedaannya dengan sistem saat ini. Ilustrasi harus dapat dibandingkan (*before* and *after*). Karena masih berupa proposal, bab ini hanya berisi gambar desain konsep solusi tersebut dan penjelasan perbandingannya dengan gambar sistem yang ada saat ini (yang tergambar di awal Bab III).

BAB V

RENCANA SELANJUTNYA

Jelaskan secara detail langkah-langkah rencana selanjutnya, hal-hal yang diperlukan atau akan disiapkan, dan risiko dan mitigasinya, yang meliputi:

1. Rencana implementasi, termasuk alat dan bahan yang diperlukan, lingkungan, konfigurasi, biaya, dan sebagainya.
2. Desain pengujian dan evaluasi, misalnya metode verifikasi dan validasi.
3. Analisis risiko dan mitigasi, misalnya tindakan selanjutnya jika ada yang tidak berjalan sesuai rencana.

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* 132:103352. https://doi.org/10.1016/j.cose.2023.103352.

Domínguez-Dorado, Manuel, Francisco J. Rodríguez-Pérez, Jesús Galeano-Brajones, Jesús Calle-Cancho, dan David Cortés-Polo. 2024. “FLECO: A tool to boost the adoption of holistic cybersecurity management”. *Software Impacts* 19:100614. https://doi.org/10.1016/j.simpa.2024.100614.

E., Chatzoglou, Kambourakis G., Karupoulos G., dan Tsiatsikas Z. 2023. “Bypassing antivirus detection: old-school malware, new tricks”. Disiniasi dari file ”2023 Bypassing antivirus detection old-school malware, new tricks.pdf”.

Elghaly dan Yehia M. 2024. “Stealth in Plain Sight: The Hidden Threat of PowerShell Fileless Malware and Its Evasion of Modern EDRs & AVs”. Disiniasi dari file ”2024 Stealth in Plain Sight The Hidden Threat of PowerShell Fileless Malware and Its Evasion of Modern EDRs AVs.pdf”.

Kareem. 2024. “A Comprehensive Analysis of Pegasus Spyware and Its Implications for Digital Privacy and Security”. Disiniasi dari file ”A Comprehensive Analysis of Pegasus Spyware and Its Implications for Digital Privacy and Security.pdf”.

Kerkour, Sylvain. 2021. *Black Hat Rust: Applied offensive security with the Rust programming language.* Self-published.

Koutsokostas, V, dan C Patsakis. 2021. “Python and Malware: Developing Stealth and Evasive Malware Without Obfuscation”. Disiniasi dari file ”2021 Python and Malware Developing Stealth and Evasive Malware Without Obfuscation.pdf”.

Sudhakar, Unknown, dan Sushil Kumar. 2020. “An emerging threat Fileless malware: a survey and research challenges”. *Cybersecurity* 3 (1): 1. https://doi.org/10.1186/s42400-019-0043-x.

Verma, V., S. K. Dubey, T. Khan, dan Pandey H. Prem. 2021. “The Study on Information Gathering”. Disiniasi dari file ”2021 Cyber Security The Study on Information Gathering.pdf”.