Journal of Applied Informatics and Computing (JAIC)
Vol.x, No.x, Juli 201x, pp. xx~xx
e-ISSN: 2548-6861

Rancang Bangun Sistem Deteksi Intrusi Berbasis *Machine Learning* dan Akuisisi Bukti Digital Menggunakan *Framework* PLC-SEIFF untuk Mitigasi Serangan *Stealth Injection* pada Lingkungan OpenPLC

Fuad Restu Rhomadhon [1]*, Hamdani Arif [2]**
* Teknik Informatika, Politeknik Negeri Batam
** Rekayasa Keamanan Siber, Politeknik Negeri Batam
fuad.4332201059@students.polibatam.ac.id [1], hamdaniarif@polibatam.ac.id [2]

Article Info

Article history:

Received …
Revised …
Accepted …

Keyword:

Deep Isolation Forest,Digital Forensics,Intrussion Detection System,OpenPLC,Stealth Injection.

ABSTRACT

Industrial Control Systems (ICS), particularly Programmable Logic Controllers (PLCs), are increasingly vulnerable to stealth program injection attacks that manipulate internal logic without altering physical outputs. Traditional network-based intrusion detection systems often fail to identify these host-level anomalies, while reactive forensic approaches risk losing volatile memory evidence. This study proposes a novel cyber-physical defense mechanism integrating a Deep Isolation Forest (DIF) machine learning model with the PLC-SEIFF (Security Incident Forensics Framework) for automated digital evidence acquisition. Implemented on an OpenPLC v3 environment simulating a cyber-physical water tank, the DIF model analyzes raw execution time (scan cycle) and physical state correlations to detect logic hijacking via GDB hot-patching. During the internal evaluation phase, the proposed DIF model achieved an accuracy of 99.05% and an F1-Score of 0.9077. Furthermore, during the testing phase with unseen data, the model maintained a robust overall accuracy of 95.00% and a Macro Average F1-Score of 87.43%, successfully capturing all malicious intrusions (100% Recall for the anomaly class) with zero false negatives.Upon anomaly detection, the automated SEIFF integration rapidly acquired volatile memory dumps and network logs, revealing the injected hex payload (`b0 01 90`) and state inconsistencies as definitive forensic evidence. This research demonstrates a highly resilient, real-time intrusion detection and automated forensic solution for mitigating advanced cyber threats in modern ICS environments.

I. PENDAHULUAN

Sistem Kendali Industri (*Industrial Control Systems* atau ICS) merupakan komponen vital dalam operasional infrastruktur kritis negara, mencakup sektor pembangkit listrik, manufaktur, hingga pengolahan air bersih. Seiring dengan meluasnya implementasi paradigma Industri 4.0, terjadi konvergensi yang masif antara Teknologi Operasional (OT) di lapangan dengan jaringan Teknologi Informasi (IT) korporat. Meskipun integrasi ini secara signifikan meningkatkan efisiensi dan visibilitas operasional, ia sekaligus menghilangkan batasan isolasi tradisional (*air-gap*) dan memperluas permukaan serangan siber secara drastis. Berbagai data insiden keamanan di dunia nyata membuktikan bahwa serangan siber yang menargetkan komponen ICS, khususnya perangkat *Programmable Logic Controller* (PLC), tidak lagi sekadar mengganggu komunikasi data, melainkan mampu memanipulasi logika kendali hingga menyebabkan kerusakan fisik yang katastrofik pada aset industri [1].

Salah satu ancaman paling canggih dan berbahaya dalam ekosistem OT saat ini adalah teknik *Stealth Program Injection* atau injeksi kode logika secara senyap. Serangan jenis ini umumnya dilakukan oleh aktor ancaman yang memiliki tingkat persistensi tinggi atau ancaman dari dalam (*insider threat*) yang berhasil mendapatkan akses tingkat

e-ISSN: 2548-6861

tinggi ke lingkungan eksekusi PLC [2]**.** Berbeda dengan serangan konvensional yang membanjiri jaringan dengan paket data palsu, peretas dalam skenario ini menyisipkan kode berbahaya langsung ke dalam memori kerja (*volatile memory*) PLC tanpa mengubah program asli yang terlihat oleh operator di *Engineering Station* . Teknik manipulasi ini menciptakan "ilusi operasional", di mana layar *Human Machine Interface* (HMI) menampilkan status indikator yang normal dan stabil, sementara proses fisik aktual di lapangan sedang dieksploitasi atau dirusak secara perlahan, meniru karakteristik serangan legendaris Stuxnet [3].

Sistem pertahanan jaringan tradisional seperti *Network Intrusion Detection System* (NIDS) sering kali tidak berdaya dalam menghadapi taktik *Stealth Program Injection*. Hal ini dikarenakan lalu lintas protokol industri konvensional (seperti Modbus TCP) yang mengalir di jaringan sering kali tetap terlihat sah dan tidak melanggar aturan berbasis tanda tangan (*signature-based*) [4]**.** Oleh karena itu, diperlukan suatu mekanisme deteksi berbasis *host* sebagai lini pertahanan terakhir (*last line of defense*) yang memantau parameter deterministik internal PLC, salah satunya adalah *scan cycle time* (waktu siklus pemindaian). PLC bekerja dengan mengeksekusi siklus logika secara terus-menerus dan periodik, mulai dari membaca input fisik, menjalankan program pengguna, hingga memperbarui output [5]**.** Injeksi instruksi atau kode asing di memori RAM, sekecil apa pun ukurannya, secara matematis pasti akan memicu deviasi atau fluktuasi waktu eksekusi logika (*jitter*) yang dapat diekstraksi sebagai fitur anomali [6].

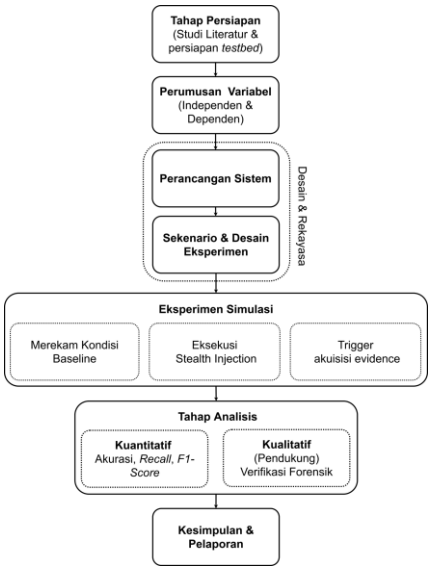
Meskipun pemantauan terhadap anomali waktu siklus dan metrik fisik operasional dapat mendeteksi adanya kejanggalan, pemodelan data ICS memiliki kompleksitas yang tinggi karena sifat hubungannya yang non-linier dan multi-dimensi. Pendekatan statistik sederhana atau penggunaan algoritma deteksi anomali tradisional seperti *Standard Isolation Forest* (IF) sering kali menghasilkan tingkat alarm palsu (*False Positive*) yang tinggi akibat fluktuasi alami beban kerja CPU PLC, yang berujung pada kelelahan peringatan (*alert fatigue*) bagi operator. Untuk mengatasi keterbatasan tersebut, penggunaan *Deep Isolation Forest* (DIF) menjadi solusi mutakhir. DIF memanfaatkan representasi fitur acak berbasis jaringan saraf tiruan (*Neural Network*) untuk memproyeksikan metrik operasional siber dan fisik PLC ke dalam ruang dimensi non-linier yang lebih tinggi, sehingga mampu mengisolasi titik pencilan serangan *stealth injection* dengan tingkat akurasi jauh lebih presisi dibandingkan metode linier standar [7].

Tantangan kritis dalam ketahanan siber (*cyber-resilience*) infrastruktur industri tidak berhenti pada fase deteksi anomali, melainkan juga pada kemampuan respons insiden dan investigasi forensik digital. Ketika sebuah serangan *Stealth Program Injection* terdeteksi, penyerang sering kali langsung melakukan pembersihan jejak (*cleanup process*), atau bukti digital pada RAM PLC akan hilang seketika jika perangkat terpaksa dimatikan atau di-*reboot* oleh sistem proteksi fisik. Penanganan forensik tradisional yang bersifat manual dan reaktif memiliki jeda waktu (*time gap*) yang fatal sehingga sering kehilangan bukti berharga. Oleh sebab itu, diperlukan integrasi kerangka kerja forensik otomatis seperti PLC-SEIFF (*Programmable Logic Controller Security Incident Forensics Framework*). Dengan kerangka kerja ini, sesaat setelah mesin kecerdasan buatan memberikan sinyal peringatan anomali, sistem akan langsung mengeksekusi akuisisi data memori *volatile* (*core dump*), tangkapan paket jaringan, dan log operasional secara *real-time* sebelum bukti tersebut dimanipulasi atau terhapus [8].

Berdasarkan permasalahan yang telah dipaparkan, penelitian ini bertujuan untuk merancang, membangun, dan menguji secara komprehensif sebuah Sistem Deteksi Intrusi (*Cyber-Physical* IDS) berbasis algoritma *Deep Isolation Forest* yang terintegrasi langsung dengan modul akuisisi forensik otomatis menggunakan *Framework* PLC-SEIFF. Implementasi dan simulasi pengujian dilakukan pada lingkungan OpenPLC v3 yang menjalankan skenario pemodelan fisik tangki air (*Water Tank Simulation*) terkendali. Penelitian ini diharapkan dapat memberikan kontribusi signifikan berupa solusi keamanan proaktif yang tidak hanya andal dalam mendeteksi ancaman manipulasi logika internal PLC secara *real-time* dengan meminimalisir alarm palsu, melainkan juga mampu mengamankan bukti digital yang valid dan utuh (*Chain of Custody*) demi kebutuhan analisis forensik pasca-insiden di lingkungan industri modern.

II. METODE

Metode yang digunakan dalam penelitian ini adalah metode eksperimental berbasis simulasi pada lingkungan terkendali (*testbed*). Pendekatan ini dipilih untuk mengevaluasi efektivitas sistem deteksi dan forensik tanpa mempertaruhkan stabilitas infrastruktur industri di dunia nyata [9]**.**



Gambar 1. Alur Tahapan Penelitian

Seperti Terlihat pada Gambar 1, alur metodologi dalam penelitian ini terdiri dari tujuh tahapan utama, yang membentang dari fase persiapan awal hingga tahap simulasi, analisis data, dan penarikan kesimpulan. Rangkaian tahapan eksperimental ini dirancang secara sistematis dengan merujuk pada kaidah pengujian *testbed* rekayasa komputer serta prinsip desain eksperimen kuantitatif mutakhir [10][11][12]**.** Melalui struktur ini, manipulasi terhadap variabel ancaman siber dapat dikontrol dan dievaluasi dampaknya secara presisi..

Lingkungan pengujian dirancang menggunakan arsitektur "Dua Realitas" (*Dual-Reality*) terisolasi yang secara tegas memisahkan sudut pandang operasional dari mesin eksekusi utama. Arsitektur ini diimplementasikan melalui sistem operasi *Host* berbasis Windows yang berfungsi sebagai *Engineering Station* sekaligus *Human Machine Interface* (HMI), serta Mesin Virtual (VM) berbasis Ubuntu Server yang bertindak sebagai pengontrol lapangan menjalankan *runtime* OpenPLC v3. Proses fisik disimulasikan menggunakan model tangki air (*Water Tank Simulation*) yang dikembangkan dengan *Python* dan berkomunikasi dua arah dengan PLC melalui protokol Modbus TCP pada *port* 502 dan 2605. Pemisahan arsitektur ini krusial untuk menciptakan skenario ilusi visual, di mana manipulasi logika pada memori pengontrol dapat disembunyikan (*blindspot*) dari antarmuka pemantauan operator yang sah.

Dalam mekanisme pertahanannya, alur kerja sistem bertumpu pada pemantauan korelasi antara metrik siber dan fisik secara kontinu. Teknik pengambilan sampel dalam penelitian ini menggunakan metode polling sistematis berkelanjutan yang disinkronkan dengan siklus eksekusi (scan cycle) OpenPLC. Justifikasi pemilihan teknik ini didasarkan pada karakteristik data waktu rill (time-series) pada lingkungan Industrial Control Systems (ICS), di mana pelestarian urutan temporal dan korelasi siklik antar parameter siber serta fisik sangat krusial untuk memastikan model Deep Isolation Forest mampu mengenali deviasi mikrosekon dari serangan stealth injection secara akurat.Data yang diekstraksi meliputi waktu siklus pemindaian (*scan cycle time*), tingkat fluktuasi latensi komputasi (*jitter*), serta inkonsistensi status *register* Modbus.

Variabel-variabel tersebut digunakan sebagai dataset untuk melatih model algoritma *Deep Isolation Forest* (DIF) agar mampu mengisolasi anomali mikrosekon yang ditimbulkan oleh interupsi *hot-patching* memori. Ketika model DIF menghasilkan skor keputusan (*anomaly score*) negatif yang melampaui ambang batas, sistem secara otomatis akan memicu kerangka kerja forensik PLC-SEIFF. Skrip otomasi ini langsung melakukan akuisisi memori *volatile* (*core dump*) dan mencetak *snapshot* status fisik (*Chain of Custody* via *hash* SHA-256) sesaat sebelum penyerang memiliki kesempatan untuk menghapus jejak eksploitasi.

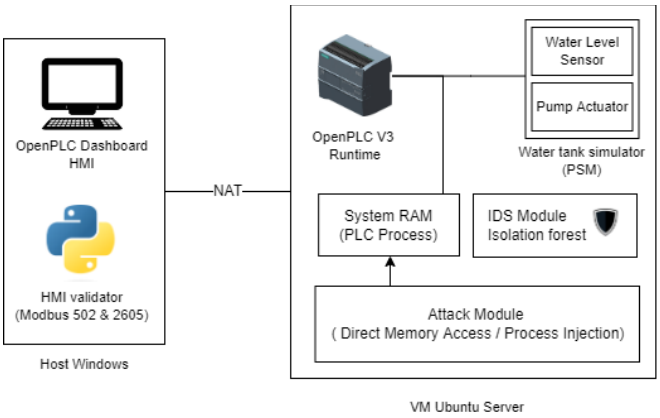
Pemilihan 6 parameter utama dalam dataset didasarkan pada pendekatan arsitektur *Cyber-Physical System* (CPS), yang memadukan metrik kinerja internal perangkat (siber) dengan variabel proses fisik lapangan. Keenam parameter tersebut yang mencakup waktu eksekusi (exec_time_ns), interval komunikasi (interval_ms), deviasi CPU jitter (jitter_std_ms), serta variabel fisik proses seperti perubahan level air (delta_level), status aktuator pompa (delta_pump), dan register kontrol I/O diperlukan karena serangan *stealth memory injection* bekerja secara senyap tanpa mengubah lalu lintas paket jaringan konvensional. Kombinasi fitur hibrida ini memberikan visibilitas penuh bagi model *Deep Isolation Forest* untuk mendeteksi anomali laten baik dari sisi komputasi host maupun dari anomali fisik tangki air.

III. PERANCANGAN DAN IMPLEMENTASI SISTEM

A. Arsitektur Lingkungan Eksperimen

Penelitian ini mengusulkan pendekatan "Dua Realitas" (Dual-Reality) untuk memisahkan antara sudut pandang operator (pengguna yang sah) dan sistem operasi inti yang rentan terhadap manipulasi. Arsitektur ini dibangun menggunakan virtualisasi dengan rincian sebagai berikut:

- *Host OS* (Windows): Berfungsi sebagai *Engineering Station* dan *Human Machine Interface* (HMI) yang menampilkan dasbor operasional kepada pengguna melalui web *browser*.
- *Guest OS* (Ubuntu Server 24.04 VM): Bertindak sebagai lingkungan lapangan terisolasi yang menjalankan *runtime* OpenPLC v3 dan skrip simulator fisik (*Python*).
- *Modbus TCP Bridge*: Komunikasi antara sistem kontrol dan simulator fisik dijembatani melalui protokol Modbus TCP, di mana OpenPLC mendengarkan pada *Port* 502, sedangkan umpan balik status fisik dari simulator diterima melalui *Port* 2605.



Gambar 2.Arsitektur Topologi sistem

B. Pemodelan Sistem

Sebagai studi kasus, penelitian ini memodelkan sistem pengisian tangki air otomatis (*Water Tank Simulation*). Logika kontrol diprogram menggunakan bahasa *Structured*

e-ISSN: 2548-6861

Text (ST) berstandar IEC 61131-3. Aturan dasar dari sistem fisik ini adalah: jika volume air turun di bawah 20%, PLC akan mengirimkan instruksi untuk menyalakan pompa air (`%ML0 = TRUE`). Sebaliknya, jika level air telah mencapai 80% dari kapasitas tangki (`%ML1028 >= 80`), pompa harus dimatikan secara otomatis. Pemodelan ini mensimulasikan sistem *loop* tertutup yang sangat sensitif terhadap perubahan nilai register kontrol seperti tertuang pada Gambar 3 Berikut.

```
PROGRAM prog0
  VAR
    Level AT %IW0 : INT;
    Pump AT %QX0.0 : BOOL;
    exec_time_internal AT %ML1028 : ULINT;
    exec_time_modbus AT %ML0 : ULINT;
  END_VAR
  exec_time_modbus := exec_time_internal;
  IF Level < 200 THEN
    Pump := TRUE;
  ELSIF Level > 800 THEN
    Pump := FALSE;
  END_IF;
END_PROGRAM
```

Gambar 3. Potongan Logika *structured text*

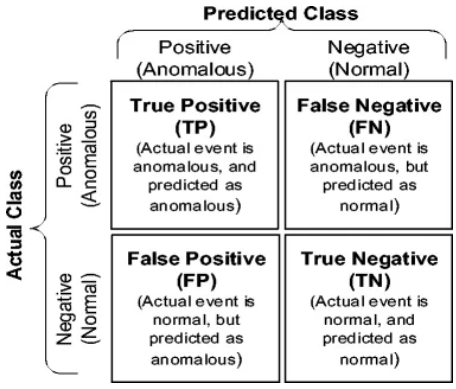
C. Sekenario Serangan Stealth Memory Injection

Untuk menguji keandalan sistem pertahanan, model ancaman (threat model) yang diterapkan adalah serangan penetrasi tingkat lokal, di mana penyerang (*insider*) diasumsikan telah berhasil mendapatkan akses terminal ke OS Ubuntu (lingkungan OpenPLC) [13]. Penyerang menggunakan GNU Debugger (GDB) untuk melakukan teknik *hot-patching* pada RAM secara real-time. Serangan ini secara spesifik mencari alamat memori fungsi `write_dig_out` dan menimpa instruksi asli (misalnya mengubah opcode menjadi `b0 01 90`). Hal ini menciptakan kondisi Logic Hijacking, di mana pompa akan terus menyala sehingga menyebabkan tangki meluap (*overflow*), namun HMI di layar Host Windows tetap menampilkan status pompa yang normal seperti logika pada gambar 3 (*blindspot*).

D. Implementasi Deteksi Deep Isolation Forest (DIF)

Menghadapi serangan senyap yang tidak mengubah struktur paket jaringan Modbus, sistem ini mengandalkan pemantauan perilaku parameter *host* dan fisik. Model *Machine Learning Deep Isolation Forest* (DIF) digunakan untuk memproses matriks fitur yang dikumpulkan oleh layanan pemantau (IDS). Fitur utama yang diekstraksi meliputi korelasi fisik (delta_level, delta_pump) dan korelasi komputasi, khususnya waktu siklus pemindaian program (exec_time_ns) dan tingkat latensinya (*jitter*). Algoritma DIF memproyeksikan fitur-fitur non-linier ini ke dalam jaringan saraf tiruan acak, lalu mempartisi ruang data. Saat GDB menghentikan sesaat proses PLC untuk menyuntikkan *payload* ke dalam memori, siklus pemindaian PLC akan mengalami perlambatan (*jitter* tinggi) selama beberapa milidetik. DIF akan mendeteksi anomali mikrosekon ini sebagai *outlier* dengan skor keputusan (anomali *score*) negatif ekstrem [7].

Sebelum model ini diimplementasikan, pengujian perbandingan dilakukan secara ketat (*benchmarking*) dengan dua algoritma varian lainnya, yaitu *Standard Isolation Forest* (IF) linier dan *Extended Isolation Forest* (EIF). Kinerja dari ketiga model tersebut dievaluasi dan divalidasi menggunakan pemetaan *Confusion Matrix* terhadap dataset uji.



Gambar 4. Tabel Confusion Matrix

Dari contoh tabel confusion matrix pada Gambar 5, pengujian difokuskan pada kalkulasi empat parameter statistik utama, yaitu, Akurasi untuk mengukur persentase prediksi sistem yang benar secara keseluruhan; Presisi (*Precision*) untuk menekan angka alarm palsu (*False Positive*); Sensitivitas (*Recall*) untuk memastikan tidak ada serangan berbahaya yang lolos (*False Negative*); serta F1-Score sebagai nilai rata-rata harmonik antara Presisi dan *Recall*. Metrik *F1-Score* menjadi parameter penentu paling krusial dalam pemilihan model akhir (DIF) karena karakteristik dataset serangan *Stealth Injection* yang sangat tidak seimbang (*highly imbalanced*), di mana insiden anomali jauh lebih sedikit dibandingkan durasi operasional normal, parameter tersebut dihitung dengan menggunakan rumus berikut:

1) *Precission:*

Precision ialah pengukuran akurasi positif prediksi dari model dengan membagi jumlah True Positives dengan jumlah total prediksi positif (TP + false positives (FP)).

$$Precission = \frac{TP}{TP + FP}$$

2) *Recall:*

Dikenal sebagai sensitivitas atau true positive rate,dihitung dengan membagi jumlah true *positives* (TP) dengan jumlah total kasus positif (TP + *false negatives* (FN)).

$$Recall = \frac{TP}{TP + FN}$$

3) *Accuracy:*
Yaitu rasio prediksi benar terhadap jumlah sampel, dihitung sebagai (TP + TN) / Total.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

4) *F1-Score:*
Ialah metrik gabungan dari Precision dan Recall menjadi satu nilai.

$$F1\text{-}Score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$$

E. Otomatisasi Kerangka Kerja PLC SEIFF

Saat fungsi ML mengklasifikasikan status sebagai *Anomalous*, sistem tidak hanya membunyikan peringatan, melainkan langsung memicu skrip seiff_acquire.py untuk mengamankan bukti digital sebelum penyerang sempat membersihkan jejak (*cleanup process*). Integrasi kerangka kerja PLC-SEIFF otomatis ini melakukan tiga tahap akuisisi kritis [8]:

1) *Akuisisi Memori Volatile:*
Mengeksekusi perintah gcore ke Process ID (PID) OpenPLC untuk menyimpan seluruh isi RAM (*core dump*) yang sedang berjalan menjadi berkas *binary*. Berkas ini krusial untuk menemukan jejak injeksi hex string.

2) *Snapshot Cyber-Physical:*
Melakukan penarikan data sinkron ganda antara register Modbus di sisi PLC (Port 502) dan di sisi Simulator (Port 2605) untuk membuktikan adanya ilusi/manipulasi status operasional, kemudian menyimpannya ke dalam format JSON.

3) *Chain of Custody*
Mengamankan integritas seluruh bukti digital (evidence vault) yang terkumpul dengan menghasilkan nilai hash kriptografi SHA-256 secara instan. Nilai hash ini memastikan keabsahan hukum (legal validity) artefak forensik untuk investigasi lebih lanjut.

IV. HASIL DAN PEMBAHASAN

Pengembangan penelitian ini dilakukan secara bertahap mulai dari perancangan simulasi Open PLC, Simulasi *Stealth Injection,* Pengambilan data latih dan data uji, pelatihan dan pengujian model, lalu integrasi dengan HMI dan akuisisi bukti digital.

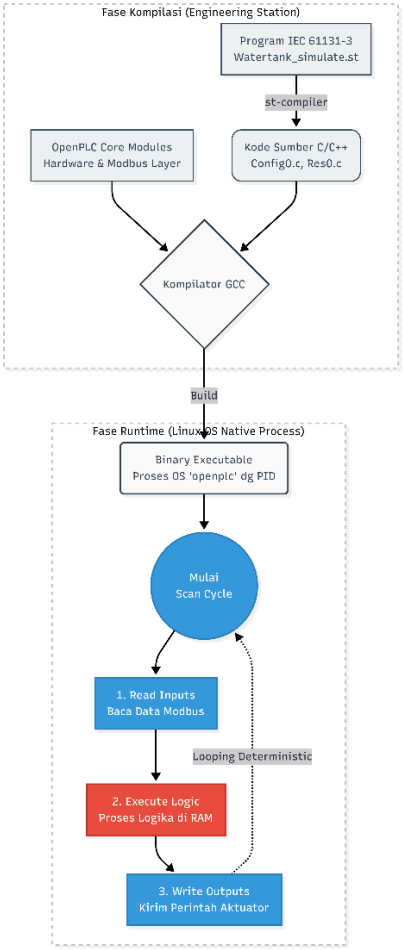
A. Arsitektur dan Logika Alur Kerja OpenPLC

Seluruh OpenPLC v3 merupakan perangkat lunak pengendali sumber terbuka yang dirancang sesuai dengan standar industri IEC 61131-3 [14]. Berbeda dengan simulator perangkat lunak konvensional, OpenPLC melakukan penerjemahan bahasa logika menjadi instruksi mesin tingkat rendah (*low-level*). Proses ini dimulai ketika program Structured Text (ST), seperti Watertank_simulate.st, diunggah ke dalam dashboard antarmuka web. Program tersebut tidak dieksekusi secara langsung melalui mesin virtual bawaan (*interpreter*), melainkan diterjemahkan terlebih dahulu ke dalam bahasa C/C++ menggunakan modul st-compiler.

Kode sumber C/C++ hasil terjemahan tersebut kemudian digabungkan dengan lapisan inti sistem OpenPLC (*Hardware Layer* dan Modbus *Server*) dan dikompilasi secara utuh menggunakan GNU *Compiler Collection* (GCC). Hasil akhir dari kompilasi ini adalah sebuah berkas biner (binary executable) bernama `openplc` yang berjalan sebagai proses tunggal (*single process*) secara mandiri atau asli (*native*) di atas kernel sistem operasi Linux Ubuntu dengan *Process* ID (PID) tertentu.

Proses eksekusi Logika *Structured Text* (ST) ini Menjadi Krusial untuk dianalisis, karena kebenaran semantic dari kode ST sangat menentukan validitas operasional dari biner yang dihasilkan [15].

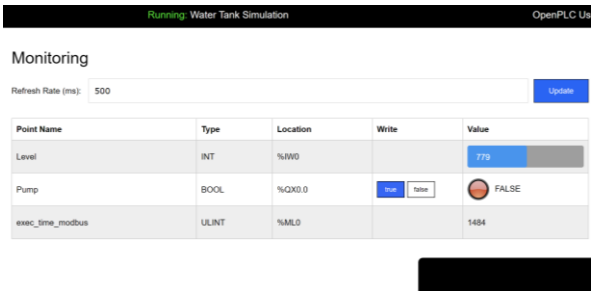
Pada Gambar 5, dapat dilihat algoritma atau cara kerja dari system OpenPLC yang berjalan di Linux.



Gambar 5.*Logic Flow* OpenPLC

e-ISSN: 2548-6861

Pembangunan simulasi dilakukan dengan mengkompilasi *structured text* ke tab program di dashboard OpenPLC dan membuat program *Hardware* Python on Linux (PSM) untuk menjalankan simulasi, sehingga setelah OpenPLC diinyalakan , akan muncul seperti gambar 6 berikut:



Gambar 6. Tab Monitoring Dashboard OpenPLC

B. *Sekenario Serangan Stealth Injection*

Skenario serangan dalam penelitian ini dirancang secara mendalam untuk mensimulasikan ancaman penetrasi tingkat lanjut (advanced persistent threat atau APT) yang bersumber dari dalam lingkungan operasional (insider threat). Fokus utama dari pemodelan ancaman (threat model) ini bukanlah eksploitasi jalur komunikasi jaringan Modbus TCP yang bersifat reaktif dan mudah dikenali oleh NIDS konvensional, melainkan manipulasi memori kerja (volatile memory) pada lapisan eksekusi biner runtime OpenPLC secara langsung dan senyap.

1) *Vektor Akses dan Eskalasi Hak Istimewa (Local Reconnaissance)*: Aktor ancaman diasumsikan telah berhasil melewati perimeter keamanan fisik atau jaringan teknologi informasi (TI) korporat, dan memperoleh akses kontrol terminal interaktif (perintah shell Bash melalui SSH atau konsol lokal) pada sistem operasi Ubuntu Server 24.04 yang menampung runtime OpenPLC v3. Langkah pertama yang dilakukan oleh penyerang adalah melakukan pemindaian internal untuk mengidentifikasi Process ID (PID) aktif dari biner eksekusi pengontrol melalui pencarian berbasis nama proses seperti pada Gambar 7 Berikut.

```
root@fuad-srv:/home/fuad# pgrep openplc
783
2217
root@fuad-srv:/home/fuad#
```

Gambar 7. Identifikasi ID Proses

2) Interupsi Proses dan Pemetaan Fungsi Aktuator (Process Hooking): Setelah memetakan PID target, penyerang memanfaatkan utilitas GNU Debugger (GDB) dengan hak akses administrator (root) untuk menempelkan (attach) debugger ke dalam proses openplc yang sedang berjalan melalui instruksi gdb -p [PID]. Eksekusi perintah ini secara mekanis akan mengirimkan sinyal interupsi kernel (SIGSTOP) yang membekukan seluruh alur instruksi scan cycle OpenPLC selama beberapa milidetik. Di dalam konsol interaktif GDB, penyerang melakukan deassemblasi (disassembly) terhadap fungsi penulisan biner keluaran digital bernama `write_dig_out` untuk membaca struktur instruksi bahasa mesin (assembly) asli serta seperti pada gambar 8 memetakan alamat memori virtualnya.

```
(gdb) disassemble write_dig_out
Dump of assembler code for function _Z13write_dig_outi:
   0x000063792ddc7775 <+0>:     endbr64
   0x000063792ddc7779 <+4>:     push   %rbp
   0x000063792ddc777a <+5>:     mov    %rsp,%rbp
   0x000063792ddc777d <+8>:     sub    $0x860,%rsp
   0x000063792ddc7784 <+15>:    mov    %edi,-0x854(%rbp)
   0x000063792ddc778a <+21>:    mov    %fs:0x28,%rax
   0x000063792ddc7793 <+30>:    mov    %rax,-0x8(%rbp)
   0x000063792ddc7797 <+34>:    xor    %eax,%eax
   0x000063792ddc7799 <+36>:    movabs $0xf00390000000100,%rax
   0x000063792ddc77a3 <+46>:    movabs $0x3290010000,%rdx
   0x000063792ddc77ad <+56>:    mov    %rax,-0x840(%rbp)
   0x000063792ddc77b4 <+63>:    mov    %rdx,-0x838(%rbp)
```

Gambar 8. Pemetaan alamat memori

3) *Injeksi Kode dan Pembongkaran Struktur Hex (Hot-Patching):* Fase inti dari serangan stealth program injection ini dilakukan dengan teknik hot-patching memori, yaitu memodifikasi byte instruksi langsung pada RAM tanpa menghentikan atau mematikan layanan OpenPLC secara permanen.

```
(gdb) set {char [3]}(_Z13write_dig_outi + 276) = {0xb0, 0x01, 0x90}
(gdb) x/3i _Z13write_dig_outi + 276
   0x58f6ca20d889 <_Z13write_dig_outi+276>:     mov    $0x1,%al
   0x58f6ca20d88b <_Z13write_dig_outi+278>:     nop
   0x58f6ca20d88c <_Z13write_dig_outi+279>:     test   %al,%al
(gdb) set {char [5]}(_Z15updateBuffersInv + 50) = {0x90, 0x90, 0x90, 0x90, 0x90}
(gdb) x/5i _Z15updateBuffersInv + 50
   0x58f6ca20de43 <_Z15updateBuffersInv+50>:    nop
   0x58f6ca20de44 <_Z15updateBuffersInv+51>:    nop
   0x58f6ca20de45 <_Z15updateBuffersInv+52>:    nop
   0x58f6ca20de46 <_Z15updateBuffersInv+53>:    nop
   0x58f6ca20de47 <_Z15updateBuffersInv+54>:    nop
(gdb) quit
```

Gambar 9 Injeksi fungsi `write_dig_out` & `updateBuffersIn`

Penyerang menimpa 3 byte instruksi pada alamat memori virtual fungsi `write_dig_out & updateBuffersIn` yang telah dipetakan sebelumnya dengan payload hex string spesifik, yaitu `b0 01 90`. Dapat dilihat Pada Gambar 9, Rincian pembongkaran semantik dari struktur hex tersebut adalah sebagai berikut:

- `b0 01`: Merupakan instruksi bahasa mesin x86/x64 untuk `mov al, 0x01`. Instruksi ini secara paksa memanipulasi register CPU al (komponen keluaran digital) agar selalu bernilai logika 1 (TRUE / ON), mengabaikan akumulasi kalkulasi logika pengkondisian tangki air yang sebenarnya di dalam program Structured Text.
- `90`: Merupakan instruksi nop (*No Operation*). Penggunaan instruksi kosong ini berfungsi sebagai pengisi ruang (*padding*) struktural agar panjang byte baru yang diinjeksikan tetap seimbang dengan panjang instruksi asli yang ditimpa, sehingga mencegah terjadinya kerusakan memori atau pemutusan proses (*segmentation fault*).

4) *Dampak Siber-Fisik dan Kebutaan Visual (Logic Hijacking & Blindspot)*: Dampak dari injeksi ini memicu kondisi pembajakan logika (*Logic Hijacking*). Sejak siklus pemindaian berikutnya dijalankan, OpenPLC akan terus-menerus mengirimkan perintah ke *Physical Simulator* bahwa status pompa air wajib aktif (ON / menyala), menyebabkan air terus mengisi tangki melampaui batas aman 80% hingga terjadi tumpahan fisik (*overflow*).

Karakteristik kepalsuan (*stealthiness*) dari serangan ini terletak pada pemisahan lapisan data. Karena manipulasi terjadi tepat di ujung memori fungsi eksekusi keluaran fisik, variabel logika internal yang dibaca oleh protokol komunikasi Modbus TCP pada *Port* 502 tidak mengalami perubahan modifikasi berkas proyek asli. Akibatnya, paket data Modbus yang dikirimkan ke stasiun operator tetap membawa data normal palsu, menyebabkan *dashboard* HMI mengalami kebutaan visual (*blindspot*) dan terus menampilkan visualisasi pompa dalam status mati (OFF) atau *normal*.

C. Pengambilan Dataset Training Dan Testing

Efektivitas dari arsitektur *Machine Learning* tanpa pengawasan (*unsupervised*) sangat bergantung pada kemurnian data dasar operasional (*baseline data*). Oleh karena itu, pengumpulan dataset dalam penelitian ini dilakukan melalui dua fase eksperimen yang terisolasi dan sistematis.

1) *Fase Perekaman Data Latih:* Fase pertama difokuskan pada pengumpulan matriks fitur siber-fisik dalam kondisi sistem yang murni dan sehat, tanpa adanya intervensi serangan. Proses Water Tank Simulator dijalankan selama durasi siklus penuh secara berulang-ulang, di mana sistem secara otomatis menyalakan pompa saat volume air di bawah 20% dan mematikannya saat mencapai 80%. Skrip deteksi latar belakang diatur untuk merekam nilai fitur (seperti `exec_time_ns` dan `delta_level`) pada setiap siklus komputasi Modbus, lalu menyimpannya ke dalam berkas ids_dataset.csv. Dataset latih ini (*training dataset*) akan bertindak sebagai landasan bagi algoritma Deep Isolation Forest untuk memahami batas ambang kewajaran (*threshold of normalcy*) dari perilaku mesin PLC, Gambar 9 berikut adalah *Head Sample* dari *Training dataset* yang mencakup 6 fitur yang disubutkan di perancangan sebelumnya.

Dataset Train berhasil dimuat!
Total Baris: 6400, Total Kolom: 8

	timestamp_wib	exec_time_ns	delta_exec_time_ns	interval_ms	jitter_std_ms	delta_level	delta_pump	label
0	21:20:44.682	1096	0	105.22	0.0000	2	0	0
1	21:20:44.685	2690	1594	4.56	50.3345	0	0	0
2	21:20:44.726	2690	0	41.46	41.5828	2	0	0
3	21:20:44.776	4443	1753	49.91	36.0124	2	0	0
4	21:20:44.826	3254	1189	50.00	32.2107	2	0	0

Gambar 10. *Head Dataset Training*

2) *Fase Perekaman Data Uji*: Fase kedua dilakukan untuk menghasilkan data uji (testing dataset) yang digunakan dalam pengukuran metrik performa (F1-Score). Pada fase ini, sistem PLC beroperasi normal pada menit-menit awal. Namun, pada titik waktu tertentu, serangan *Stealth Memory Injection* (menggunakan *hot-patching* `GDB b0 01 90`) diluncurkan secara tiba-tiba. Fluktuasi siklus pemindaian yang melambat akibat interupsi, serta status fisik tangki yang mulai meluap, terekam bersama dengan data normal sebelum dan sesudah serangan terjadi. Dataset campuran ini disimpan sebagai ids_dataset_testing.csv dan merepresentasikan kondisi serangan nyata yang senyap, pada gambar 10 dapat dilihat *Head Sample* dari *Testing dataset.*

Dataset Test berhasil dimuat!
Total Baris: 2300, Total Kolom: 8

	timestamp_wib	exec_time_ns	delta_exec_time_ns	interval_ms	jitter_std_ms	delta_level	delta_pump	label
0	22:51:44.440	1297	0	224.52	0.0000	0	0	0
1	22:51:44.442	1297	0	3.66	110.4344	0	0	0
2	22:51:44.443	1297	0	1.71	104.5810	0	0	0
3	22:51:44.444	1297	0	1.20	96.2790	0	0	0
4	22:51:44.464	1297	0	19.78	87.4453	3	0	0

Gambar 11. *Head Dataset Testing*

D. Training, Evaluasi, dan Testing Model DIF

Berbeda dengan algoritma klasifikasi tradisional yang membutuhkan pelabelan kelas (labeling) pada setiap baris data, model pertahanan ini mengadopsi mekanisme unsupervised learning. Proses pelatihan dan pengujian difasilitasi menggunakan bahasa Python (beserta library Scikit-Learn dan PyTorch untuk jaringan saraf tiruannya).

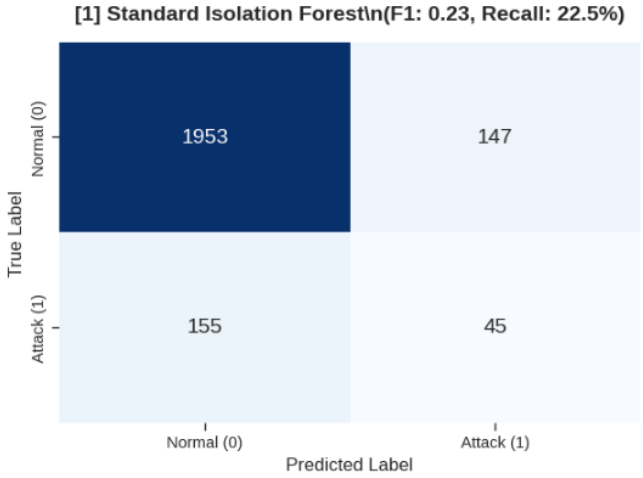
1) *Pelatihan Model:* Model Deep Isolation Forest (DIF) dilatih secara eksklusif menggunakan Dataset (ids_dataset.csv). Model memproses 6 fitur numerik utama dan memetakannya ke dalam 64 ruang dimensi menggunakan jaringan saraf acak (random neural network). Proses ini memungkinkan model untuk membangun pohon-pohon isolasi (*isolation trees*) yang mendefinisikan batasan ketat dari kondisi normal sistem. Karena model hanya diajarkan mengenai karakteristik "normal", setiap penyimpangan data di kemudian hari akan langsung diidentifikasi sebagai anomali tanpa perlu mengenali varian serangannya (*signature-less*).

2) *Pengujian dan Evaluasi* : Setelah model berhasil dibangun, model tersebut diuji ketajamannya dengan memprediksi dataset ids_dataset_testing.csv (*Testing Phase*). Skor keputusan (*anomaly score*) yang dihasilkan oleh DIF untuk setiap baris data dievaluasi. Data operasional normal umumnya menghasilkan skor positif (mendekati 1), sedangkan anomali akibat interupsi GDB akan ditekan (isolated) ke arah pangkal pohon keputusan, menghasilkan skor negatif ekstrem (mendekati -1).

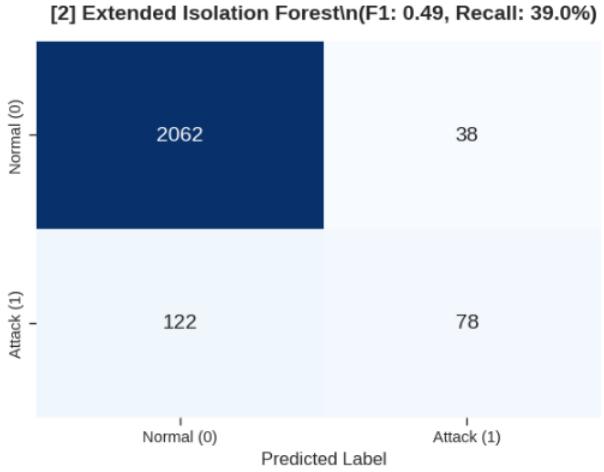
Hasil prediksi dari model tersebut kemudian dievaluasi menggunakan instrumen statistik Confusion Matrix. Evaluasi ini membandingkan kemampuan model DIF dengan varian algoritma standar, yaitu Standard Isolation Forest (IF) dan Extended Isolation Forest (EIF). Metrik utama yang dikalkulasi meliputi Akurasi (Accuracy), Presisi (Precision), Sensitivitas (Recall), dan F1-Score. F1-Score digunakan

e-ISSN: 2548-6861

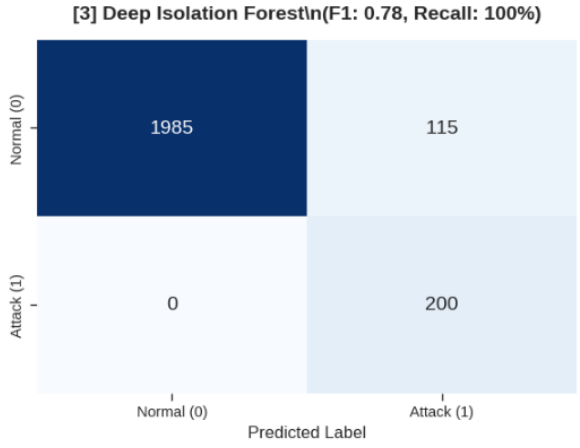
sebagai parameter validasi akhir untuk memastikan model tidak terjebak dalam bias akibat mayoritas data normal, sekaligus menjamin tidak adanya serangan yang lolos dari deteksi (False Negative).



Gambar 12. Hasil *Confusion Matrix* Isolation Forest



Gambar 13.Hasil *Confusion Matrix EIF*



Gambar 14.Hasil *Confusion Matrix Standard DIF*

Hasil *Confusion Matrix* dari *Standard Isolation Forest* yang terlihat pada gambar 10, *Extended Isolation Forest* pada Gambar 11, dan *Deep Isolation Forest* pada Gambar 12, kemudian dilakukan analisa perhitungan untuk mendapatkan nilai akurasi dengan rumus seperti pada perancangan.

1) *Perhitungan Akurasi Standard Isolation Forest :*
Dengan Nilai TN = 1953, FP = 147, FN = 155, TP = 45 didapatkan nilai akurasi sebagai berikut.

$$Accuracy = \frac{45 + 1953}{45 + 1953 + 147 + 155} = 0.86869$$

2) *Perhitungan Akurasi Extended Isolation Forest :*
Dengan Nilai TN = 2062, FP = 38, FN = 122 dan TP = 78 didapatkan nilai akurasi sebagai berikut.

$$Accuracy = \frac{78 + 2062}{78 + 2062 + 38 + 122} = 0.93043$$

3) *Perhitungan Akurasi Deep Isolation Forest :*
Dengan Nilai TN = 1985, FP = 115, FN = 0 dan TP = 200 didapatkan nilai akurasi sebagai berikut.

$$Accuracy = \frac{200 + 1985}{200 + 1985 + 115 + 0} = 0.9500$$

Dengan ini didapatkan akurasi untuk masing masing model adalah *Standard Isolation Forest* dengan akurasi 0.86869 = 86.87%, *Extended Isolation Forest* dengan akurasi 0.93043 = 93.04% dan Deep isolation forest dengan skor tertinggi dengan akurasi 0.9500 = 95.00%, Sesuai dengan perancangan pengujian tidak ditujukan pada metrik akurasi saja melainkan 3 metrik lainnya sebagaimana tertuang pada tabel 1, 2 dan 3 berikut :

Tabel 1
Hasil *Testing Matrix Standard Isolation Forest*

	precision	**Recall**	**F1-score**	**support**
0	0.9265	0.9300	0.9282	2100
1	0.2344	0.2250	0.2296	200

Tabel 2
Hasil *Testing Matrix Extended Isolation Forest*

	precision	**Recall**	**F1-score**	**support**
0	0.9441	0.9819	0.9627	2100
1	0.6724	0.3900	0.4937	200

Tabel 3
Hasil *Testing Matrix Deep Isolation Forest*

	precision	**Recall**	**F1-score**	**support**
0	1.0000	0.9452	0.9718	2100
1	0.6349	1.0000	0.7767	200

Dari tabel 1, 2 dan 3 kemudian dibuat tabel perbandingan *Macro Average* dengan rumus :

$$\text{Macro Average} = \frac{\text{Normal Metric} + \text{Anomaly Metric}}{2}$$

untuk ketiga model untuk data yang lebih terukur adil dan jelas sebagaimana tabel 4 berikut :

Tabel 4.
Nilai *Macro average* setiap model

Model	Acc	Precision	Recall	F1-score
IF	86.87%	0.5804	0.5775	0.5789
EIF	93.04%	0.8083	0.6860	0.7282
DIF	95.00%	0.8175	0.9726	0.8743

Meskipun arsitektur *Deep Isolation Forest* (DIF) yang diusulkan berhasil mencapai tingkat *Recall* 100% (berhasil mendeteksi seluruh serangan tanpa *false negative*), metrik *Precision* pada pengujian kelas anomali tercatat sebesar 63,49%. Hal ini mengindikasikan bahwa model menghasilkan sejumlah *False Positive* (alarm palsu), di mana fluktuasi operasional normal terklasifikasi sebagai serangan. Penggunaan komponen *Neural Network* untuk mentransformasikan parameter *scan cycle* mentah OpenPLC ke dimensi non-linier yang lebih tinggi terbukti meningkatkan sensitivitas model, sehingga fluktuasi mikroskopis seperti variasi latensi jaringan (*network delay*) atau lonjakan beban CPU sesaat ikut terisolasi sebagai anomali laten [5].

Dalam domain keamanan Sistem Kontrol Industri (ICS) dan *Operational Technology* (OT), konsekuensi dari kegagalan mendeteksi satu serangan manipulasi memori (False Negative) dapat berakibat fatal pada kerusakan fatal aset fisik di dunia nyata, sehingga minimalisasi risiko kebobolan serangan lewat pencapaian *Recall* maksimal menjadi prioritas pertahanan lapis terakhir yang paling rasional [6]. Meskipun demikian, tingkat *Precision* sebesar 63,49% ini membawa implikasi operasional yang serius berupa risiko *alert fatigue* (kelelahan alarm), suatu kondisi berbahaya di mana operator *Human-Machine Interface* (HMI) berpotensi mengabaikan atau mematikan fungsi peringatan karena terlalu sering menerima indikasi bahaya yang tidak valid.

Guna mengatasi kelemahan metrik presisi tersebut, arsitektur pertahanan hibrida dalam penelitian ini dirancang agar alarm yang dipicu oleh model *Machine Learning* tidak langsung dilemparkan sebagai peringatan akhir kepada operator, melainkan divalidasi terlebih dahulu melalui otomatisasi pemeriksaan otomatis terhadap batasan fisik (*security constraints*) operasional pada *volatile memory dump* [16]. Melalui mekanisme validasi silang ini, integritas pendeteksian ancaman stealth injection tingkat tinggi tetap terjaga secara rill tanpa mengorbankan kenyamanan operasional operator di lapangan.

E. Integrasi HMI Windows dan Service Automasi

Sistem yang telah dievaluasi tidak akan memberikan manfaat perlindungan jika tidak diimplementasikan pada lingkungan produksi nyata (*Live Deployment*). Oleh karena itu, model Deep Isolation Forest (DIF) beserta konfigurasi bobot jaringannya diekspor ke dalam bentuk berkas pickle (.pkl) dan .pth, lalu ditanamkan (*embedded*) secara langsung ke dalam mesin virtual Ubuntu Server.
Implementasi proaktif dilakukan dengan merancang skrip *Python* yang beroperasi secara laten sebagai *Background Service* bernama live_ids_service.py. Layanan otomatis ini bertindak sebagai penjaga gerbang siber-fisik dengan alur operasional sebagai berikut:

- Layanan melakukan penarikan data (*scraping*) setiap 500 milidetik (*sliding window*) dari port Modbus 502 (HMI) dan port Modbus 2605 (Simulator Fisik) secara bersamaan.
- Data mentah tersebut diekstraksi ke dalam 6 fitur matriks dan langsung diumpankan ke model DIF untuk diklasifikasikan.
- Selama status diprediksi normal (skor positif), sistem hanya akan memperbarui log informasi. Namun, jika GDB melakukan interupsi yang menyebabkan *jitter* tinggi, model secara instan mengeluarkan status [ANOMALY_DETECTED].

Gambar 15. HMI terintegrasi IDS kondisi Normal.

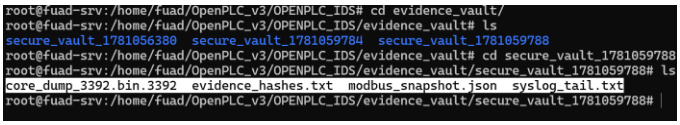
Gambar 16. HMI terintegrasi IDS kondisi injeksi

e-ISSN: 2548-6861

Pada lingkungan *Host* Windows, Gambar 13 dan Gambar 14 antarmuka operator (*Engineering Station*) tetap menampilkan visualisasi HMI OpenPLC secara normal melalui peramban web. Mekanisme ini memvalidasi kerentanan "kebutaan sensor" (*sensor blindspot*) pada arsitektur industri konvensional, di mana serangan manipulasi pada level memori bawah tidak akan memicu peringatan di layar pengguna, menegaskan urgensi dari keberadaan layanan *host-based IDS* yang independen.

F. Akuisisi Data Forensik

Tantangan terbesar dalam menangani insiden manipulasi memori aktif (*Stealth Injection*) adalah hilangnya barang bukti saat mesin dimatikan atau saat penyerang melakukan pembersihan jejak (*cleanup process*). Untuk menanggulangi hal tersebut, penelitian ini mengotomatisasi Programmable Logic Controller Security Incident Forensics Framework (PLC-SEIFF) melalui implementasi skrip seiff_acquire.py. Sesaat setelah layanan IDS mendeteksi anomali pada siklus pemindaian, sistem respons tidak melakukan pemutusan jaringan (*kill-switch*), melainkan langsung memicu (trigger) eksekusi modul forensik dalam hitungan milidetik. Proses akuisisi (*Data Acquisition*) dilakukan dalam tiga tahapan berantai untuk menjamin integritas pembuktian:



Gambar 17. *Data Evidence*

- Penarikan Memori Volatile (*Memory Dumping*): Modul mengeksekusi perintah `gcore -o core_dump [PID]` terhadap proses OpenPLC yang terinfeksi. Proses ini menyalin secara presisi seluruh ruang alamat memori RAM yang sedang dieksekusi (termasuk *payload hex* `b0 01 90` yang disuntikkan penyerang) ke dalam berkas core_dump.bin.
- Pengambilan Snapshot Cyber-Physical: Modul memotret ketidakselarasan status antara tampilan siber (HMI) dan respons fisik (Simulator) yang terjadi pada detik eksploitasi, lalu menyimpannya dalam dokumen berformat JSON.
- Penguncian Rantai Pasokan Bukti (*Chain of Custody*): Segera setelah berkas memori dan JSON terbentuk, modul menjalankan kalkulasi fungsi kriptografis menggunakan pustaka hashlib. Nilai *hash* SHA-256 dari setiap berkas artefak dikalkulasi dan disimpan secara permanen ke dalam berkas evidence_hashes.txt. Langkah ini merupakan prosedur wajib dalam investigasi digital untuk membuktikan bahwa berkas *core dump* tidak diubah atau dirusak secara sengaja sejak pertama kali diamankan dari mesin PLC.

V. KESIMPULAN DAN SARAN

Berdasarkan hasil eksperimen dan pengujian yang telah dilakukan pada arsitektur "Dua Realitas" lingkungan OpenPLC, penelitian ini berhasil membuktikan kerentanan fatal dari serangan Stealth Memory Injection (hot-patching). Serangan ini terbukti mampu membajak logika komputasi PLC secara langsung di memori volatile dan merusak proses cyber-physical (menyebabkan tangki air meluap), sekaligus menciptakan efek "kebutaan sensor" (blindspot) pada dashboard operator HMI.

Untuk memitigasi celah eksploitasi tingkat rendah (low-level) tersebut, integrasi host-based IDS proaktif terbukti sangat krusial. Pendekatan konvensional atau Machine Learning standar terbukti tidak memadai karena memicu tingkat False Positive (alarm palsu) yang berujung pada fenomena Alert Fatigue. Sebagai solusinya, pemanfaatan algoritma Deep Isolation Forest (DIF) yang mampu memproyeksikan data ke dalam jaringan saraf tiruan berdimensi tinggi terbukti sangat efektif. Model DIF ini sukses mendeteksi deviasi waktu siklus pemindaian (*jitter*) berukuran mikrosekon secara instan, Pada Evaluasi menghasilkan nilai Akurasi sebesar 99,05%, nilai Recall mutlak 1,0000 (100%), dan metrik tangguh F1-Score sebesar 0,9077, kemudian pada Testing mendapatkan nilai akurasi sebesar 95,00%., nilai Recall 100% precision 63,49% dan F1-Score Sebesar 77,67%, jauh melewati nilai yang didapatkan oleh *Standard Isolation Forest* dan *Extended Isolation Forest.*

Meskipun capaian metrik *precision* pada data uji kelas anomali berada di angka 63,49% yang menunjukkan adanya *False Positive* akibat sifat model DIF yang sangat sensitif terhadap fluktuasi operasional minor, hal ini merupakan *trade-off* rasional demi menjamin *Recall* mutlak 100% agar tidak ada serangan *fileless* yang lolos. Implikasinya, klaim efisiensi pengurangan alarm palsu dan mitigasi risiko *alert fatigue* pada operator HMI tetap terpenuhi secara sistemik, karena peringatan dini dari model ML ini tidak berdiri sendiri melainkan langsung divalidasi oleh pemeriksaan batasan fisik otomatis dari sistem forensik. Lebih jauh, penelitian ini sukses mengotomatisasi respons insiden siber. Sesaat setelah anomali sistem diklasifikasikan oleh model DIF, kerangka kerja forensik otomatis PLC-SEIFF berhasil dieksekusi secara instan (*zero time-gap*). Kerangka kerja ini mengunci bukti mutlak yang berupa berkas *core dump* memori yang berisi hex *string payload* `b0 01 90`, mencatat log inkonsistensi fisik pada modbus_snapshot.json, serta memastikan seluruh integritas barang bukti diamankan secara kriptografis dengan hash SHA-256 (*Chain of Custody*).

Sebagai arah pengembangan dan rekomendasi di masa mendatang, penelitian ini mengidentifikasi beberapa keterbatasan operasional, termasuk batasan skenario serangan tunggal (*single attack scenario*) pada fungsi `write_dig_out`, skema evaluasi data tunggal (*single*

train/test split), serta implikasi dari ketidakseimbangan data (*dataset imbalance*). Oleh karena itu, penelitian lanjutan disarankan untuk memvalidasi keandalan arsitektur ini terhadap variasi muatan serangan (*multiple injection payloads*) yang lebih luas serta menerapkan metode validasi silang (*cross-validation*) untuk memperkuat keandalan model secara statistik sekaligus meminimalkan bias distribusi data. Eksperimen pemisahan dataset dalam penelitian ini masih menggunakan metode *single train/test split*, ketiadaan variabel target mutlak (*Y*) pada model *unsupervised learning* membuat penerapan mekanisme *train/test split* maupun *cross-validation* konvensional menjadi sangat menantang dan subjektif [17]. Oleh karena itu, penelitian selanjutnya disarankan untuk merumuskan kerangka *cross-validation* hibrida yang secara khusus disesuaikan untuk kasus deteksi anomali, guna mengeliminasi potensi bias yang timbul dari fenomena *dataset imbalance*.

UCAPAN TERIMA KASIH

Puji syukur penulis panjatkan ke hadirat Allah SWT; Terimakasih yang paling dalam kepada kedua orang tua atas segala fasilitas dan dukunganya; Terimakasih Bapak Hamdani Arif atas waktu bimbinganya; Terimakasih kepada seluruh staf pengajar Program Studi Rekayasa Keamanan Siber Politeknik Negeri Batam.

DAFTAR PUSTAKA

[1] G. M. Makrakis, C. Kolias, G. Kambourakis, C. Rieger, and J. Benjamin, "Industrial and Critical Infrastructure Security: Technical Analysis of Real-Life Security Incidents," *IEEE Access*, vol. 9, pp. 165295–165325, 2021, doi: 10.1109/ACCESS.2021.3133348.

[2] J. Song and Y. Moon, "Security enhancement against insiders in cyber-manufacturing systems," *Procedia Manuf.*, vol. 48, no. 2019, pp. 864–872, 2020, doi: 10.1016/j.promfg.2020.05.124.

[3] Z. Sembiring, "Stuxnet Threat Analysis in SCADA (Supervisory Control And Data Acquisition) and PLC (Programmable Logic Controller) Systems," *J. Comput. Sci. Inf. Technol. Telecommun. Eng.*, vol. 1, no. 2, pp. 96–103, 2020, doi: 10.30596/jcositte.v1i2.5116.

[4] W. Alsabbagh, C. Kim, and P. Langendorfer, "Investigating the Security of OpenPLC: Vulnerabilities, Attacks, and Mitigation Solutions," *IEEE Access*, vol. 12, no. January, pp. 11561–11583, 2024, doi: 10.1109/ACCESS.2024.3356051.

[5] E. A. Boateng and J. W. Bruce, "Unsupervised Machine Learning Techniques for Detecting PLC Process Control Anomalies," pp. 220–244, 2022.

[6] J. H. Lee, I. H. Ji, and S. H. Jeon, "Anomaly Detection Method Considering PLC Control Logic Structure for ICS Cyber Threat Detection," 2025.

[7] S. Gupta, "Securing ICS Environments with AI-Enabled Classification : A Proactive Threat Detection Approach," vol. 27, no. 8, pp. 439–453, 2025.

[8] L. Xu, B. Wang, L. Wang, D. Zhao, X. Han, and S. Yang, "PLC-SEIFF: A programmable logic controller security incident forensics framework based on automatic construction of security constraints," *Comput. Secur.*, vol. 92, 2020, doi: 10.1016/j.cose.2020.101749.

[9] I. Haroon, "Inshal Haroon A METASPLOIT-BASED EXPLOITING APPROACH TO PLC VULNERABILITIES Faculty of Information Technology and Communication Sciences," no. June, 2025.

[10] L. Kirkup, *Experimental Methods for Science and Engineering Students: An Introduction to the Analysis and Presentation of Data*. 2019. doi: 10.1017/9781108290104.

[11] F. Amigoni and V. Schiaffonati, *Methods and in Computer Techniques Experimental Engineering*. Springer Science & Business Media, 2013.

[12] D. T. Campbell and J. C. Stanley, *Campbell and Stanley for Undergraduates*, vol. 29, no. 4. 1984. doi: 10.1037/022808.

[13] W. Alsabbagh and P. Langendorfer, "A Stealth Program Injection Attack against S7-300 PLCs," *Proc. IEEE Int. Conf. Ind. Technol.*, vol. 2021-March, pp. 986–993, 2021, doi: 10.1109/ICIT46573.2021.9453483.

[14] T. Alves and T. Morris, "OpenPLC: An IEC 61,131–3 compliant open source industrial controller for cyber security research," *Comput. Secur.*, vol. 78, pp. 364–379, 2018, doi: 10.1016/j.cose.2018.07.007.

[15] K. Wang, J. Wang, C. M. Poskitt, X. Chen, J. Sun, and P. Cheng, "K-ST: A Formal Executable Semantics of the Structured Text Language for PLCs," *IEEE Trans. Softw. Eng.*, vol. 49, no. 10, pp. 4796–4813, 2023, doi: 10.1109/TSE.2023.3315292.

[16] F. Fu, P. Liu, Z. Shao, J. Xu, and M. Fang, "MEvo-GAN: A Multi-Scale Evolutionary Generative Adversarial Network for Underwater Image Enhancement," *J. Mar. Sci. Eng.*, vol. 12, no. 7, 2024, doi: 10.3390/jmse12071210.

[17] J. T. Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani, *An Introduction to Statistical Learning: with Applications in Python*, vol. 7, no. 10. 2023. doi: 10.2174/0929867003374372.