KERJA PRAKTIK - IF234603

Perancangan dan Implementasi *Analyzer* Artefak Forensik *Windows* untuk *DFTPL*

Laboratorium Teknologi Jaringan dan Keamanan Siber Cerdas (*NETICS*)
Departemen Teknik Informatika ITS, Keputih, Kec. Sukolilo, Surabaya, Surabaya 60111
Periode: 1 Juli 2024 - 31 Desember 2024

Oleh:

Christopher Clement Wijaya 5025211155

Pembimbing Jurusan
Dr. Wahyu Suadi, S.Kom, M.Kom.

Pembimbing Lapangan
Dr. Baskoro Adi P., S.Kom.,M.Kom.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2024

Perancangan dan Implementasi *Analyzer* Artefak Forensik *Windows* untuk *DFTPL*

Laboratorium Teknologi Jaringan dan Keamanan Siber Cerdas (NETICS)
Departemen Teknik Informatika ITS, Keputih, Kec. Sukolilo, Surabaya, Surabaya 60111
Periode: 1 Juli 2024 - 31 Desember 2024

Oleh:

Christopher Clement Wijaya 5025211155

Pembimbing Jurusan
Dr. Wahyu Suadi, S.Kom, M.Kom.

Pembimbing Lapangan
Dr. Baskoro Adi P., S.Kom.,M.Kom.

DEPARTEMEN TEKNIK INFORMATIKA
Fakultas Teknologi Elektro dan Informatika Cerdas
Institut Teknologi Sepuluh Nopember
Surabaya 2024

[Halaman ini sengaja dikosongkan]

LEMBAR PENGESAHAN

KERJA PRAKTIK

Perancangan dan Implementasi *Analyzer* Artefak Forensik *Windows* untuk *DFTPL*

Oleh:

Christopher Clement Wijaya 5025211155

Disetujui oleh Pembimbing Kerja Praktik:

1. Dr. Wahyu Suadi, S.Kom, M.Kom.
   NIP. 197110302002121001

(Pembimbing Departemen)

2. Dr. Baskoro Adi P., S.Kom.,M.Kom.
   NIP. 198702182014041001

(Pembimbing Lapangan)

[Halaman ini sengaja dikosongkan]

Perancangan dan Implementasi *Analyzer* Artefak Forensik *Windows* untuk *DFTPL*

Nama Mahasiswa	**: Christopher Clement Wijaya**
NRP	**: 5025211155**
Departemen	**: Informatika FTEIC-ITS**
Pembimbing Jurusan	**: Dr. Wahyu Suadi, S.Kom, M.Kom.**
Pembimbing Lapangan	**: Dr. Baskoro Adi P., S.Kom.,M.Kom.**

ABSTRAK

Dalam forensik digital, salah satu cara untuk peneliti bisa memahami insiden yang terjadi adalah dengan membuat *timeline* dari berbagai artefak dalam suatu sistem. Salah satu alat yang bisa digunakan untuk membuat *timeline* forensik secara otomatis adalah *Log2Timeline/Plaso*. *Plaso* secara otomatis mencari informasi dari beragam artefak dan menyatukannya dalam satu *super timeline*. Akan tetapi, jumlah data yang sangat banyak menyulitkan peneliti dalam menganalisis *timeline*.

Salah satu pendekatan untuk mengatasi masalah jumlah data tersebut adalah dengan membuat *high-level timeline*. *Low-level timeline* yang terdiri dari informasi dari artefak dianalisis untuk menghasilkan *high-level timeline* dimana setiap *event* berasal dari 1 atau lebih *low-level event*. *High-level timeline* juga dirancang agar mudah dibaca manusia. Salah satu alat yang mengimplementasikan pendekatan ini adalah *PyDFT*.

Saat ini, *PyDFT* sudah lama tidak dikembangkan. Maka, dikembangkan *DFTPL*, alat berbahasa *Python* yang menggunakan pendekatan serupa untuk menghasilkan *high-level timeline* dari *timeline* luaran *Plaso*. Akan tetapi, masih banyak jumlah artefak yang belum didukung oleh *DFTPL*. Dalam kerja praktik ini, penulis mengembangkan 22 *analyzer* baru untuk artefak dari *Windows*. Dibuat juga *unit test* menggunakan luaran *Plaso* untuk memastikan *analyzer* menghasilkan luaran yang diharapkan.

Kata Kunci: Timeline, Digital Forensic, Windows Artifact, Event Reconstruction

KATA PENGANTAR

Puji syukur penulis ucapkan kepada Tuhan yang Maha Esa karena berkat rahmat-Nya penulis dapat melaksanakan salah satu kewajiban penulis sebagai mahasiswa Departemen Informatika, yakni Kerja Praktik (KP).

Penulis menyadari masih ada kekurangan baik dalam pelaksanaan kerja praktik maupun penyusunan buku laporan ini. Namun, diharapkan buku laporan ini dapat menambah wawasan pembaca dan dapat menjadi sumber referensi. Penulis mengharapkan kritik dan saran yang membangun untuk kesempurnaan buku laporan kerja praktik ini.

Melalui buku ini, penulis juga ingin menyampaikan rasa terima kasih kepada orang-orang yang telah membantu, baik langsung maupun tidak langsung, dalam pelaksanaan kerja praktik hingga penyusunan laporan. Orang-orang tersebut antara lain adalah:

1. Kedua orang tua penulis.
2. Bapak Dr. Wahyu Suadi, S.Kom, M.Kom. selaku dosen pembimbing.
3. Bapak Dr. Baskoro Adi P., S.Kom.,M.Kom. selaku pembimbing lapangan Kerja Praktik.
4. Bapak Hudan Studiawan, S.Kom., M.Kom., Ph.D selaku *developer DFTPL*.

Surabaya, 27 Desember 2024
Christopher Clement Wijaya

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I PENDAHULUAN

1.1. Latar Belakang

Dalam forensik digital, rekonstruksi kejadian adalah salah satu cara peneliti bisa memahami insiden yang terjadi. Salah satu caranya adalah dengan membuat *timeline* (Carrier & Spafford, 2004). Sebelumnya, *timeline* hanya berisi waktu terjadinya proses *MAC (Modified, Accessed, and Creation)* dari *file system*. Namun, dengan menambahkan lebih banyak sumber informasi lain, peneliti bisa mengetahui konteks ketika suatu kejadian terjadi. Peneliti juga bisa lebih percaya pada informasi yang ditemukan jika didapat dari beberapa sumber (Carvey & Altheide, 2011).

Salah satu alat yang dirancang untuk membangun *timeline* dari beberapa sumber (artefak) adalah *log2timeline* yang dikembangkan oleh Guðjónsson, (2010) dan sekarang menjadi *Plaso* (Metz, 2024a). *Framework* tersebut bertujuan untuk membuat *super timeline* berisi informasi dari berbagai sumber. Akan tetapi, banyaknya informasi dalam *timeline* menyulitkan peneliti dalam proses menganalisis. Maka, Hargreaves & Patterson, (2012) merancang pendekatan yang membangun *high-level event* yang lebih mudah dipahami manusia dari *low-level event*. Pendekatan ini diimplementasikan peneliti tersebut dalam *framework Python PyDFT*.

Saat ini, *PyDFT* sudah tidak dikembangkan dan hanya tersedia kode *analyzers* (Hargreaves, 2023) sehingga Studiawan, (2024) mengembangkan *DFTPL* dengan pendekatan serupa dengan sumber *low-level timeline* berupa *timeline* dari *log2timeline*. Akan tetapi, kode *analyzer* dari Hargreaves, (2023) dibangun sebelum *Windows 10* (2015) dan *Windows 11* (2021). Terdapat juga artefak *Windows* yang masih belum tersedia *analyzer*nya. Maka dalam KP ini, penulis diberi kesempatan untuk berkontribusi dalam pengembangan *DFTPL* dengan mengembangkan *analyzer* untuk artefak *Windows* yang belum dibuat.

1.2. Tujuan

Tujuan kerja praktik ini adalah menyelesaikan kewajiban nilai kerja praktik sebesar 2 sks dan membantu *developer* DFTPL dalam pengembangan *analyzer* dengan kontribusi minimal 20 *analyzer* untuk artefak dari sistem operasi *Windows 11*.

1.3. Manfaat

Manfaat dari luaran kerja praktik ini adalah untuk mendukung pengembangan *DFTPL* agar alat bisa digunakan oleh peneliti dalam mengolah *timeline Plaso.*

1.4. Rumusan Masalah

Rumusan masalah dari kerja praktik ini adalah sebagai berikut:

- Apa saja artefak dari *Windows 11* dalam *timeline Plaso* yang belum didukung *DFTPL?*
- Bagaimana cara mengimplementasi *analyzer* dalam *DFTPL* agar mendukung *high-level event* artefak tersebut?

1.5. Lokasi dan Waktu Kerja Praktik

Lokasi kerja praktik ini dikerjakan dari rumah *(work from home)* dengan pertemuan tatap muka atau *online* untuk koordinasi dengan *developer DFTPL*. Waktu kerja praktik dilakukan mulai dari 1 Juli 2024 hingga 31 Desember 2024.

1.6. Metodologi Kerja Praktik

1.6.1. Perumusan Masalah

Rumusan masalah dibentuk dalam pertemuan dengan Pak Hudan Studiawan, S.Kom., M.Kom., Ph.D. selaku *developer DFTPL*. Dalam pertemuan, beliau memaparkan masalah yang dihadapi *DFTPL* dan target penelitian yang harus dicapai. Beliau juga mengenalkan penulis dengan *log2timeline/Plaso* dan dasar teori dari *DFTPL*.

1.6.2. Studi Literatur

Studi literatur dilakukan terhadap alat yang akan digunakan yaitu bahasa pemrograman *Python*, *Plaso*, dan *DFTPL*. Dilakukan juga riset terhadap artefak-artefak forensik *Windows* yang belum dikembangkan untuk *DFTPL*. Studi literatur dilakukan dengan menggunakan berbagai sumber seperti artikel ilmiah dari jurnal dan dokumen atau artikel dari sumber relevan seperti organisasi di bidang keamanan digital.

1.6.3. Analisis dan Perancangan

Artefak-artefak yang ditemukan dalam tahap studi literatur kemudian dikonfirmasi keberadaannya dalam *timeline Plaso* dengan menjalankan skenario. Penulis menggunakan *virtual machine Windows 11* dan melakukan kejadian-kejadian yang menghasilkan artefak yang diuji. *Image vmdk* dari *virtual machine* tersebut kemudian dianalisis menggunakan *Plaso.* Apabila ditemukan *low-level event* dari *timeline Plaso*, maka akan dirancang dan diimplementasikan *analyzer* untuk artefak tersebut. Selain skenario dari penulis, Pak Hudan membantu menyediakan *timeline* dari skenario yang beliau buat. Penulis juga menganalisis artefak dari *test data Plaso* dan dari sistem operasi *Windows 11* dari laptop pribadi untuk beberapa artefak.

1.6.4. Implementasi Sistem

Implementasi *analyzer* dilakukan untuk artefak-artefak yang ditemukan *low-level event*nya oleh *Plaso*. *Analyzer* dikembangkan mengikuti struktur dasar *analyzer* yang sudah dikembangkan oleh

Pak Hudan. Proses implementasi *analyzer* meliputi analisis *regular expression* yang mendefinisikan *low-level event* dan membangun *high-level event* dengan mengekstrak dan menyimpan nilai-nilai penting dari teks *low-level event*.

1.6.5. Pengujian dan Evaluasi

Pengujian *analyzer* dilakukan dengan merancang *unit test*. *Unit test* digunakan untuk memastikan hasil kode *analyzer* sesuai dengan yang diharapkan. Apabila tidak sesuai, kode *analyzer* direvisi hingga memenuhi *unit test*.

1.6.6. Kesimpulan dan Saran

Penulis memberikan kesimpulan dan saran yang bisa diberikan.

1.7. Sistematika Laporan

1.7.1. Bab I Pendahuluan

Pada bab ini, dipaparkan latar belakang permasalahan, tujuan, tempat dan waktu pelaksanaan, sistematika pengerjaan kerja praktik, dan sistematika penulisan laporan kerja praktik.

1.7.2. Bab II Profil Lab

Pada bab ini, dipaparkan secara singkat profil lab *NETICS*.

1.7.3. Bab III Tinjauan Pustaka

Pada bab ini, dijelaskan konsep dan informasi yang relevan dengan kerja praktik. Topik yang dijelaskan adalah mengenai bahasa pemrograman *Python*, *Regular Expression*, artefak forensik, *timeline* forensik, *log2timeline/Plaso*, serta *PyDFT* dan *DFTPL*.

1.7.4. Bab IV Implementasi Sistem

Pada bab ini, dijelaskan proses implementasi *analyzer* pada umumnya seperti perancangan *regex* dan pembuatan *high-level event*. Dipaparkan juga referensi dari artefak dan *regex* yang mendefinisikan *low-level event* yang ditangkap *analyzer* untuk semua artefak yang diimplementasi dalam kerja praktik ini. Hal ini karena banyak kesamaan antara struktur setiap *analyzer* dan jumlah *analyzer* yang cukup banyak.

1.7.5. Bab V Pengujian dan Evaluasi

Pada bab ini, dijelaskan proses implementasi dan pengujian menggunakan *unit test analyzer*. Dipaparkan juga sumber *low-level event* dan *file* artefak yang dianalisis *Plaso* untuk membuat *unit test analyzer*.

1.7.6. Bab VI Kesimpulan dan Saran

Pada bab ini, dipaparkan kesimpulan dan saran yang bisa diberikan.

[Halaman ini sengaja dikosongkan]

BAB II PROFIL LAB

2.1. Profil Lab *NETICS*

NETICS (Networking Technology and Intelligent Cybersecurity Laboratory) atau Laboratorium Teknologi Jaringan dan Keamanan Siber Cerdas adalah laboratorium dengan fokus pada arsitektur dan keamanan jaringan. *NETICS* memiliki beberapa mata kuliah keahlian bidang keamanan seperti perancangan keamanan sistem dan jaringan dan topik khusus teknologi jaringan dan keamanan siber cerdas. *NETICS* juga aktif dalam pengabdian masyarakat seperti dengan kegiatan "Raising cybersecurity awareness: Developing a Cyber Range for education and simulation" pada tahun 2024 (Departemen Teknik Informatika, 2024).

[Halaman ini sengaja dikosongkan]

BAB III TINJAUAN PUSTAKA

3.1 *Python*

Python adalah bahasa pemrograman tingkat tinggi yang dikembangkan oleh Guido van Rossum. *Python* memiliki beberapa karakteristik. Untuk tipe variabel, *Python* bersifat dinamis sehingga tidak harus mendefinisikan jenis variabel karena menggunakan *interpreter* untuk menerjemahkan tipe variabel yang dimaksud. Desain *Python* membuat kode lebih mudah dibaca (Lubanovic, 2015). *Python* juga gratis untuk digunakan dengan versi terakhir adalah 3.13.1 (Python Software Foundation, 2024).

3.2 *Regular Expression*

Regular Expression (Regex) adalah bahasa yang digunakan untuk mengolah dan memanipulasi teks. *Regex* tersusun dari serangkaian karakter yang terdiri dari karakter normal (dipasangkan dengan karakter itu sendiri, "v" sama dengan "v") dan *metacharacter* atau *metasequence*. *Metacharacter* atau *metasequence* adalah karakter yang melambangkan fungsi seperti jumlah, lokasi, atau jenis karakter khusus. Beberapa contoh dari *metacharacter* dan *metasequence* bisa dilihat pada Tabel 3.1 (Stubblebine, 2007).

Tabel 3.1 Contoh Metacharacter dan Metasequence

Jenis	Fungsi	Contoh
Representasi Karakter	Melambangkan karakter yang sulit ditulis.	• "\n": Baris baru. • Representasi karakter dalam *octal* atau *hex* seperti "\xoD" untuk *carriege return*.
Kelas Karakter	Mendefinisikan suatu set karakter.	• [...] atau [^...]: Mencocokkan atau tidak mencocokkan salah satu dari

		karakter dalam kurung. • ".": Artinya mencocokkan hampir semua karakter. • Kelas karakter seperti "\w" untuk karakter huruf dan "\W" untuk selain karakter huruf.
Anchors	Mendefinisikan posisi dalam teks.	• "^" atau "\A": Mencocokkan awal dari kalimat. • "$" atau "\Z": Mencocokkan akhir dari kalimat.
Grouping and Capturing	Mengelompokkan dan menangkap bagian tertentu dari teks.	• (...): Pola *regex* yang mengelompokkan teks yang cocok untuk ditangkap dan digunakan nantinya.
Control	Mengendalikan berapa kali suatu pola *regex* dicocokkan.	• *, +, ?, {a, b}: *Greedy quantifier*, berusaha mencocokkan dengan sebanyak mungkin karakter dalam teks. Contohnya "*" berarti 0 atau lebih sedangkan

		"+" berarti 1 atau lebih. • *?, +?, ??, {a, b}?: *Lazy quantifier*, berusaha mencocokkan dengan sesedikit mungkin karakter dalam teks.

Dalam *Python*, karakter khusus di atas seperti "$" bisa didahului dengan "\" agar dicocokkan dengan karakter "$", bukan fungsi khususnya (Python Software Foundation, 2024).

Regex bisa digunakan untuk *pattern matching* yaitu mencari teks yang dideskripsikan oleh suatu *regex* menggunakan program *regular expression engine*. Menurut Stubblebine, (2007) terdapat 2 aturan dasar yang membantu dalam memahami teks yang dideskripsikan *regex*:

- *Regex* dicocokkan dengan teks dari kiri ke kanan mulai dari karakter pertama *regex* hingga ke akhir. Jika semua karakter *regex* berhasil dicocokkan, *regex engine* mengembalikan teks yang dicocokkan.
- Jenis *quantifier* dasar adalah *greedy*. Hal ini berarti *regex engine* akan mencoba mencocokkan *regex* pada sebanyak mungkin karakter. Jika tidak menemukan kecocokan, maka *engine* akan mengurangi karakter yang dicoba satu per satu (*backtracking*).

3.3 Artifak Forensik *Windows*

Artefak forensik adalah jejak-jejak dari kejadian atau aktivitas yang terjadi dalam suatu sistem. Dalam forensik digital, peneliti menggunakan artefak forensik untuk membuktikan validitas suatu hipotesis kejadian. Artefak akan dianalisis untuk mendapatkan informasi dan maksud dari informasi tersebut. Artefak forensik digital bisa didapat dari berbagai sumber, seperti *disk, file system*, sistem operasi seperti *Linux, Windows*, dan *Mac OS*, serta dari aktivitas internet seperti *web browser* (Carvey & Altheide, 2011).

Dalam *Windows*, terdapat beragam sumber artefak forensik. Contoh artefak bisa ditemukan dari *registry* dan *event logs*. *Registry* adalah *database* berisi informasi dan konfigurasi untuk sistem operasi dan aplikasi. *Registry* juga berisi informasi yang unik untuk setiap *user* (Carvey & Altheide, 2011). *Event logs* adalah rekaman *error* dan kejadian baik dari sistem operasi maupun aplikasi yang dikumpulkan dalam satu tempat (Karl-Bridge-Microsoft et al., 2021b). *Event logs* berisi informasi yang bisa digunakan untuk memahami kondisi dan konteks terjadinya suatu masalah (Karl-Bridge-Microsoft et al., 2021a).

3.4 *Timeline* Forensik

Timeline forensik adalah salah satu bentuk dari rekonstruksi kejadian dalam forensik. Dalam rekonstruksi kejadian, kejadian-kejadian diurutkan berdasarkan bukti dan informasi yang ada. Hal ini bisa membantu peneliti untuk memahami penyebab terjadinya suatu insiden. Dalam forensik digital, kejadian bisa diurutkan dengan mencari *timestamp* atau informasi waktu dalam suatu barang bukti (Carrier & Spafford, 2004).

3.5 *Log2Timeline/Plaso*

Log2Timeline adalah *framework* yang dikembangkan untuk secara otomatis melakukan *parsing* atau mengolah informasi dari berbagai artefak dan membangun *timeline* yang berisi seluruh informasi yang diolah (Guðjónsson, 2010). Saat ini, *Log2Timeline* telah dikembangkan menjadi *Plaso*, program *Python* yang *open source* sehingga pengguna bisa berkontribusi seperti mengusulkan *parser* dan *plugin* untuk artefak baru (Metz, 2024a). Selain

log2timeline, *Plaso* sendiri menyediakan beberapa program lain seperti *pinfo* untuk mengolah pesan dan *error* dari *log2timeline*, *psort* untuk menghasilkan luaran sesuai format serta melakukan *filtering* dan *sorting* luaran *log2timeline*, dan *psteal* jika ingin menjalankan *log2timeline* dan *psort* sekaligus (Metz, 2024b).

3.6 *PyDFT* dan *DFTPL*

PyDFT adalah *framework Python* yang dikembangkan oleh Hargreaves & Patterson, (2012). *Framework* ini menggunakan *parsers* untuk menghasilkan *high-level events* dari *low-level events*. *Low-level event* adalah informasi kejadian yang diambil langsung dari artefak seperti waktu modifikasi *file* atau perubahan *registry key*. *High-level event* adalah informasi kejadian yang mudah dipahami manusia. *High-level event* sendiri dibentuk dari satu atau lebih *low-level event* tertentu. Tujuan dari pendekatan ini adalah untuk mengatasi banyaknya jumlah data dari *low-level event* yang dihasilkan sehingga mempersulit proses analisis dengan mencari informasi yang relevan dalam *timeline*.

PyDFT sendiri sudah tidak dikembangkan oleh peneliti (Hargreaves, 2023). Maka, *DFTPL* dikembangkan oleh Studiawan, (2024) menggunakan pendekatan yang serupa dengan *PyDFT* namun tidak menghasilkan sendiri *low-level events*. *DFTPL* menerima *timeline* luaran *log2timeline* dalam format *.csv* sebagai sumber *low-level events* yang akan diproses menggunakan *analyzers* untuk menghasilkan *high-level events*.

[Halaman ini sengaja dikosongkan]

BAB IV IMPLEMENTASI SISTEM

4.1 Struktur Analyzer

Berikut adalah struktur *analyzer* pada umumnya. Untuk kode, agar ditampilkan *analyzer* yang sebelumnya dirancang Pak Hudan yaitu *ProcessCreation.py*.

4.1.1 Author dan Import

Kode bisa dilihat pada Gambar 4.1.

```
__author__ = 'Hudan Studiawan'

import re
from dftpl.events.LowLevelEvent import LowLevelEvent
from dftpl.events.HighLevelEvent import HighLevelEvent, ReasoningArtefact
from dftpl.timelines.HighLevelTimeline import HighLevelTimeline
```

Gambar 4.1 Contoh Kode *Author* dan *Import*

Selain *import library* yang dibutuhkan seperti *re* untuk *regex*, dilakukan juga *import* untuk kelas-kelas *event* dan *HighLevelTimeline*. Kelas-kelas tersebut sudah diimplementasikan oleh Pak Hudan.

4.1.2 Informasi *Parser*

Kode bisa dilihat pada Gambar 4.2.

```
description = "Process Creation"
analyser_category = "Windows"
```

Gambar 4.2 Contoh Kode Informasi *Parser*

Deskripsi dan kategori *analyzer* disesuaikan dengan jenis artefak. Umumnya deskripsi menjelaskan kejadian yang terjadi jika artefak ditemukan dan kategori memberi keterangan sistem operasi seperti "*windows*"atau jenis kejadian seperti "*user activity*".

4.1.3 Metode *run()*

Kode bisa dilihat pada Gambar 4.3.

```
def Run(low_timeline, start_id=0, end_id=None):
    """Runs the Process Creation analyser"""
    if end_id == None:
        end_id = len(low_timeline.events)

    return FindProcessCreation(low_timeline, start_id,
```

Gambar 4.3 Contoh Kode Metode *run()*

Metode ini memanggil metode *analyzer* utama. Metode ini belum digunakan, namun menurut penulis bisa bermanfaat ketika semua *analyzer* dimasukkan dalam kelas lalu dibutuhkan kode yang memanggil *analyzer* dalam daftar yang tidak tentu dimana kelas *analyzer* menggunakan *abstract class*.

4.1.4 Metode Utama

4.1.4.1 *Test Event*

Kode bisa dilihat pada Gambar 4.4.

```
def FindProcessCreation(low_timeline, start_id, end_id):
    """Finds process creation events based on event structure"""

    # Create a test event to match against
    test_event = LowLevelEvent()
    test_event.type = "Creation Time-EVT"
    test_event.evidence = r'\[\s*(9707|0×25eb)\s*/\s*(9707|0×25eb)\s*\].*\'([^\']+\.exe)'
```

Gambar 4.4 Contoh Kode *Test Event*

Setelah membuat *instance* objek *LowLevelEvent*, didefinisikan nilai *type* dan *evidence*. Nilai 2 variabel tersebut akan digunakan dalam pencarian *regex* untuk mencari *low-level event* yang memenuhi definisi *analyzer* tersebut.

4.1.4.2 *Matching Event*

Kode bisa dilihat pada Gambar 4.5.

```
    # Create a high level timeline to store the results
    high_timeline = HighLevelTimeline()

    # Find matching events
    trigger_matches = low_timeline.find_matching_events_in_id_range(start_id, end_id,
test_event)
```

Gambar 4.5 Contoh Kode *Matching* Event

Setelah membuat *HighLevelTimeline*, nilai dari *test_event* akan digunakan dalam pencarian *regex* dalam *low-level timeline* untuk mendapatkan daftar objek *LowLevelEvent* yang memenuhi definisi kejadian *analyzer* tersebut.

4.1.4.3 Membuat *High-Level Event*

Kode bisa dilihat pada Gambar 4.6.

```
    # Extract details from matching events
    for each_low_event in trigger_matches:
        if each_low_event.match(test_event):
            match = re.search(test_event.evidence, each_low_event.evidence)
            if match:
                # Get matched groups from the regex
                windows_event_id = match.group(1)
                windows_event_id_hex = match.group(2)
                executable_name = match.group(3)

                # Create a high level event
                high_event = HighLevelEvent()
                high_event.id = each_low_event.id
                high_event.add_time(each_low_event.date_time_min)
                high_event.evidence_source = each_low_event.evidence
                high_event.type = "Process Creation"
                high_event.description = f"Process creation of '{executable_name}'"
                high_event.category = analyser_category
                high_event.plugin = each_low_event.plugin
                high_event.files = each_low_event.path
                high_event.set_keys("Windows Event ID", windows_event_id)
                high_event.set_keys("Windows Event ID (hex)", windows_event_id_hex)
                high_event.set_keys("Executable name", executable_name)
                high_event.supporting =
low_timeline.get_supporting_events(each_low_event.id)
```

Gambar 4.6 Contoh Kode Membuat *High-Level Event*

Untuk setiap objek *LowLevelEvent* yang ditemukan, akan dibuat 1 *HighLevelEvent*. Berikut adalah penjelasan setiap variabel *HighLevelEvent* yang diberi nilai:

- *id:* Nilai *id* dari *LowLevelEvent*. Berisi *indeks* baris dari *timeline Plaso*.
- *evidence_source:* Berisi informasi yang didapat *Plaso* dari suatu artefak.
- *type:* Tipe dari *analyzer*. Umumnya berisi kejadian yang ditandai oleh *event* tersebut dan sumbernya jika bisa berasal dari beberapa artefak.
- *description:* Deskripsi informasi yang ditemukan dari *low-level event*. Tertulis dengan cara yang mudah dipahami manusia sehingga terdapat informasi yang tidak disertakan.

- *category:* Kategori *analyzer*.
- *plugin:* Informasi *plugin Plaso* yang menganalisis artefak sumber informasi *event*.
- *files: Path* dari *file* artefak yang dianalisis *Plaso*.
- *keys:* Nilai-nilai penting yang didapat dari artefak. Umumnya menjadi bagian dari *evidence_source* sehingga *evidence_source* harus diolah agar *substring keys* bisa diambil.
- *supporting: LowLevelEvent* yang terjadi sesaat sebelum atau setelah *LowLevelEvent* yang dianalisis *analyzer* ini.

4.1.4.4 Membuat *Reasoning*

Kode bisa dilihat pada Gambar 4.7.

```
                # Create a reasoning artefact
                reasoning = ReasoningArtefact()
                reasoning.id = each_low_event.id
                reasoning.description = f"Process creation event of '{executable_name}'
found with Windows event ID {windows_event_id}"
                reasoning.test_event = test_event
                reasoning.provenance = each_low_event.provenance
                reasoning.references = 'https://github.com/Psmths/windows-forensic-
artifacts/blob/main/execution/evtx-9707-shell-core.md'
```

Gambar 4.7 Contoh Kode Membuat *Reasoning*

Reasoning adalah informasi mengenai *low-level event* dan artefak yang menjadi sumbernya. Berisi deskripsi sumber informasi, referensi artefak, dan baris *low-level event* luaran *Plaso*.

4.1.4.5 Menambahkan ke *High-Level Timeline*

Kode bisa dilihat pada Gambar 4.8.

```
            # Add the reasoning artefact to the high level event
            high_event.trigger = reasoning.to_dict()

            # Add the high level event to the high level timeline
            high_timeline.add_event(high_event)

    return high_timeline
```

Gambar 4.8 Contoh Kode Menambahkan ke *High-Level Timeline*

Terakhir, objek *HighLevelEvent* dimasukkan ke dalam *HighLevelTimeline*. Proses menggabungkan *HighLevelTimeline* dari beberapa *analyzer* dan mengurutkannya berdasarkan waktu sudah ada dalam kelas *HighLevelTimeline*.

Terdapat beberapa *analyzer* yang memiliki perbedaan struktur dengan yang di atas. Contohnya adalah metode tambahan yang dibutuhkan untuk mengolah informasi dalam *low-level timeline*. Salah satu contoh metode tambahan bisa ditemukan dalam *FileDownloads.py* yang bisa dilihat pada Gambar 4.9.

```
def GetURIStringFromMessage(message: str) → str:
    """Extracts the URI from a message"""
    uri = re.search(r"https?:\/\/[^\s]+\/", message)
    if uri:
        return uri.group(0)
    else:
        return None
```

Gambar 4.9 Kode Metode GetURIStringFromMessage()

Contoh ini adalah fungsi tambahan untuk mendapatkan *substring* berisi informasi *URI* dari *evidence_source*.

4.2 Daftar *Analyzer* yang Dibuat

4.2.1 Tabel Informasi *Analyzer*

Di bawah ini adalah Tabel 4.1 dan Tabel 4.2 yang berisi daftar semua *analyzer* yang dibuat penulis dalam kerja praktik ini beserta penjelasan tambahan, *regex* yang digunakan, dan referensi artefaknya. Untuk kode setiap *analyzer* bisa diakses pada *Github repository DFTPL* (Studiawan, 2024) pada *branch "python-analyzers"*.

Tabel 4.1 Daftar *Regex Analyzer*

Analyzer	***Regex***
Users.py (*user account* baru dibuat)	1. *Registry:* \\SAM\\Domains\\Account\\Users\\Names\\.+$ 2. Dibuatnya *user folder:* *Users*\\.*[^\\]+$*
TimezoneSettings.py (*Timezone Windows*)	1. *Registry:* \\Control\\TimeZoneInformation] 2. *File* artefak: Windows\\System32\\config\\SYSTEM
NetworkCards.py (Waktu *installation network interface* baru)	*Registry:* \\Microsoft\\Windows NT\\CurrentVersion\\NetworkCards\\[^\\]*?$
NetworkProfiles.py (Waktu sistem operasi terhubung dengan *network* baru)	*Registry:* \\Microsoft\\Windows NT\\CurrentVersion\\NetworkList\\Profiles\\{.*?}
DeviceInstallation.py (Perangkat terhubung dengan sistem operasi)	*setup.api*: Device Install \(.+\) - .+ -
WindowsFirewallDisable.py (*Windows Firewall* dimatikan)	1. *Event log:* ^\[2082.*?\].+Source Name: Microsoft-Windows-Windows Firewall With Advanced Security Strings: \['[124]' '1' '.+?' '.+?' 'No' 2. *Registry:* Services\\SharedAccess\\Parameters\\FirewallPolicy\\(?:Standard|Public|Domain)Profile\]

WindowsEventLogCleared.py (Ada jenis *log* dalam *Event Log* yang dihapus pengguna)	*Event Log:* 1. ^\[104 \/.+\] Provider identifier: {.+} Source Name: Microsoft-Windows-Eventlog 2. ^\[1102 \/.+\] Provider identifier: {.+} Source Name: Microsoft-Windows-Eventlog
DefaultBrowser.py (*Browser* yang otomatis dibuka sistem)	*Registry:* ^\[HKEY_CURRENT_USER\\Software\\Microsoft\\Windows\\Shell\\Associations\\UrlAssociations\\https\\UserChoice
FailedLogin.py (*User* gagal *login*)	*Event Log*: ^\[4625 \/.+\] Provider identifier: {.+} Source Name: Microsoft-Windows-Security-Auditing
USBConnectedRegDeviceClasses.py (Perangkat USB terhubung)	*Registry*: ^\[HKEY_LOCAL_MACHINE\\System\\ControlSet00\d\\Control\\DeviceClasses\\{a5dcbf10-6530-11d2-901f-00c04fb951ed}\\[^\\]+\\#
USBConnectedRegUSB.py (Perangkat USB terhubung)	*Registry*: ^\[HKEY_LOCAL_MACHINE\\System\\ControlSet00\d\\Enum\\USB] Product: PID_
USBConnectedRegUSBSTOR.py (Perangkat penyimpanan USB terhubung)	*Registry*: ^\[HKEY_LOCAL_MACHINE\\System\\ControlSet00\d\\Enum\\USBSTOR\\[^\\]+\\[^\\]+\]
USBConnectedWinevt.py (Perangkat penyimpanan USB terhubung)	*Event*: ^\[1006 \/ 0x03ee\] Provider identifier: {.+} Source Name: Microsoft-Windows-Partition
LastExecutedPrefetch.py (Terakhir eksekusi)	^Prefetch \[.+\] was executed
LastExecutedUserAssist.py (Terakhir eksekusi)	*Registry*: ^\[HKEY_CURRENT_USER\\Software\\Microsoft\\Windows\\CurrentVersion\\Explorer\\UserAssist\\\{\S*\}\\Count
LastExecutedBAM.py (Terakhir eksekusi)	1. \[\S+\]$ 2. Jenis *plugin*: REG-Background Activity Moderator Registry Key-winreg/bam
LastExecutedPCA.py (Terakhir eksekusi)	1. ^\[.+\] was executed – 2. Jenis *plugin*: Program Compatibility Assistant (PCA) Log

ServiceInstalled.py (*Windows Service* baru)	*Event Log:* ^\[7045 \/ 0x1b85\] .+? Source Name: (.*?) Strings: \[(?:\'(.*?)\'|None) (?:\'(.*?)\'|None) (?:\'(.*?)\'|None) (?:\'(.*?)\'|None) (?:\'(.*?)\'|None)\]
RunRunOnceRegistry.py (Eksekusi saat *user login*)	*Registry*: ^\[(?:HKEY_CURRENT_USER|HKEY_LOCAL_MACHINE)\\Software\\(?:WOW6432Node\\)? Microsoft\\Windows\\CurrentVersion\\ (?:Run|RunOnce|RunOnce\\Setup|RunServices|RunServicesOnce)\]
TrustRecordsRegistry.py (Dokumen *Office* dari internet yang dipercayai)	*Registry:* ^\[HKEY_CURRENT_USER\\Software\\ Microsoft\\Office\\\S*?\\ (?:PowerPoint|Excel|Word)\\Security\\Trusted Documents\\TrustRecords\]
FileMRURegistry.py (Dokumen yang dibuka pengguna)	*Registry:* ^\[HKEY_CURRENT_USER\\Software\\ Microsoft\\Office\\\S*?\\ (?:PowerPoint|Excel|Word)\\(?:File MRU|User MRU\\LiveId_.+?\\File MRU)\]
RecycleBin.py (Informasi *file* di *recycle bin*)	1. Tipe: Content Deletion Time-RECBIN 2. .* *Evidence* tidak dicari karena berisi *path* asli *file* yang dihapus.

Tabel 4.2 Daftar Referensi *Analyzer*

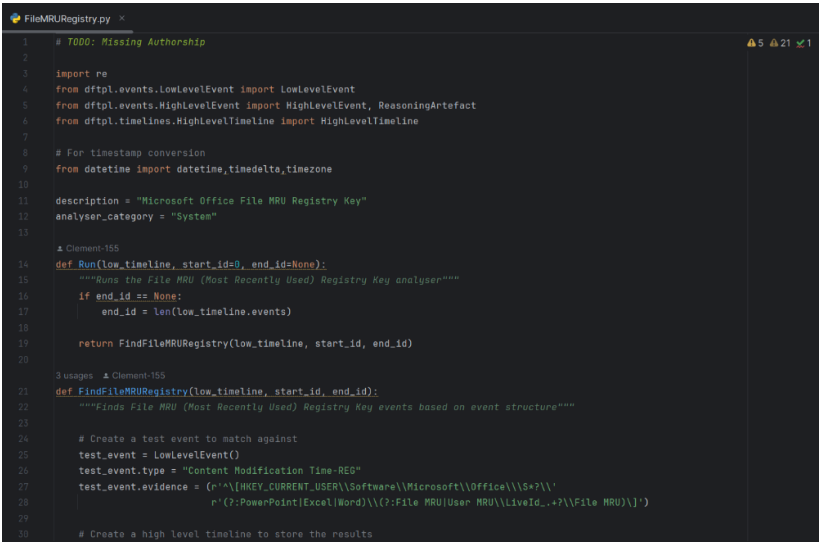
Analyzer	**Referensi**
Users.py (*user account* baru dibuat)	https://rwmj.wordpress.com/2010/06/09/windows-sam-and-hivex/
TimezoneSettings.py (*Timezone Windows*)	https://www.digital-detective.net/time-zone-identification/ (Accessed : 14:00, 26th of October 2024)

NetworkCards.py (Waktu *installation network interface*)	https://www.giac.org/paper/gawn/1623/wireless-networks-windows-registry-computer-been/121403
NetworkProfiles.py (Waktu sistem operasi terhubung dengan *network* baru)	https://www.giac.org/paper/gawn/1623/wireless-networks-windows-registry-computer-been/121403
DeviceInstallation.py (Perangkat terhubung dengan sistem operasi)	https://learn.microsoft.com/en-us/windows-hardware/drivers/install/format-of-a-text-log-section
WindowsFirewallDisable.py (*Windows Firewall* dimatikan)	1. *Event log:* https://detection.fyi/sigmahq/sigma/windows/builtin/firewall_as/win_firewall_as_setting_change/ 2. *Registry:* https://blogs.cisco.com/security/talos/opening-zxshell
WindowsEventLogCleared.py (Ada jenis *log* dalam *Event Log* yang dihapus pengguna)	*Event Log:* 1. https://docs.logrhythm.com/devices/docs/v-2-0-evid-104-eventlog-log-file-cleared 2. https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-10/security/threat-protection/auditing/event-1102
DefaultBrowser.py (*Browser* yang otomatis dibuka sistem)	https://forensafe.com/blogs/Windows-Default-Browser.html
FailedLogin.py (*User* gagal *login*)	https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-10/security/threat-protection/auditing/event-4625
USBConnectedRegDeviceClasses.py (Perangkat USB terhubung)	https://doi.org/10.1016/j.diin.2019.02.004
USBConnectedRegUSB.py (Perangkat USB terhubung)	https://doi.org/10.1016/j.diin.2019.02.004
USBConnectedRegUSBSTOR.py (Perangkat	https://doi.org/10.1016/j.diin.2019.02.004

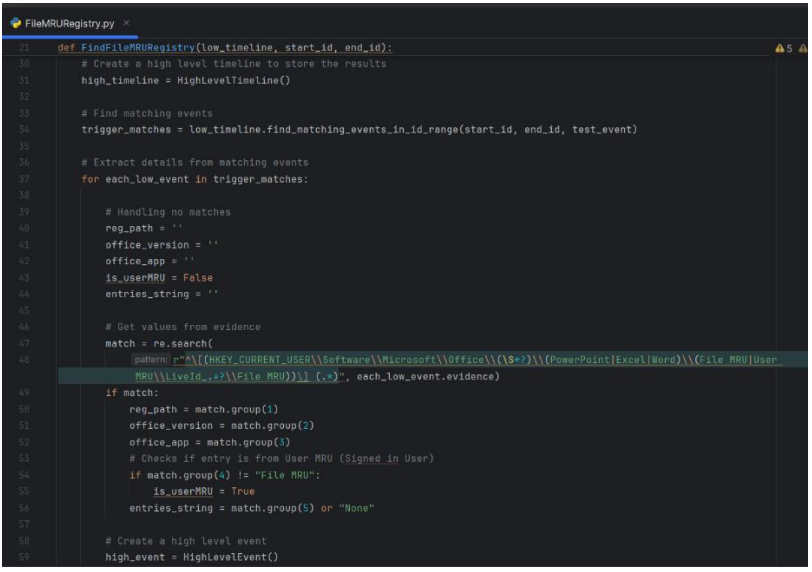
penyimpanan USB terhubung)	
USBConnectedWinevt.py (Perangkat penyimpanan USB terhubung)	https://dfir.pubpub.org/pub/h78di10n/release/2
LastExecutedPrefetch.py (Terakhir eksekusi)	https://www.magnetforensics.com/blog/forensic-analysis-of-prefetch-files-in-windows/
LastExecutedUserAssist.py (Terakhir eksekusi)	https://www.magnetforensics.com/blog/artifact-profile-userassist/
LastExecutedBAM.py (Terakhir eksekusi)	https://docs.velociraptor.app/docs/forensic/evidence_of_execution/
LastExecutedPCA.py (Terakhir eksekusi)	https://artefacts.help/windows_pca.html
ServiceInstalled.py (*Windows Service* baru)	https://research.splunk.com/endpoint/429141be-8311-11eb-adb6-acde48001122/
RunRunOnceRegistry.py (Eksekusi saat *user login*)	https://attack.mitre.org/techniques/T1547/001/
TrustRecordsRegistry.py (Dokumen *Office* dari internet yang dipercayai)	https://www.bleepingcomputer.com/news/security/windows-registry-helps-find-malicious-docs-behind-infections/
FileMRURegistry.py (Dokumen yang dibuka pengguna)	https://www.cybertriage.com/artifact/office-mru-registry/
RecycleBin.py (Informasi *file* di *recycle bin*)	https://www.magnetforensics.com/blog/artifact-profile-recycle-bin/

4.2.2 Contoh *Analyzer*: FileMRURegistry.py

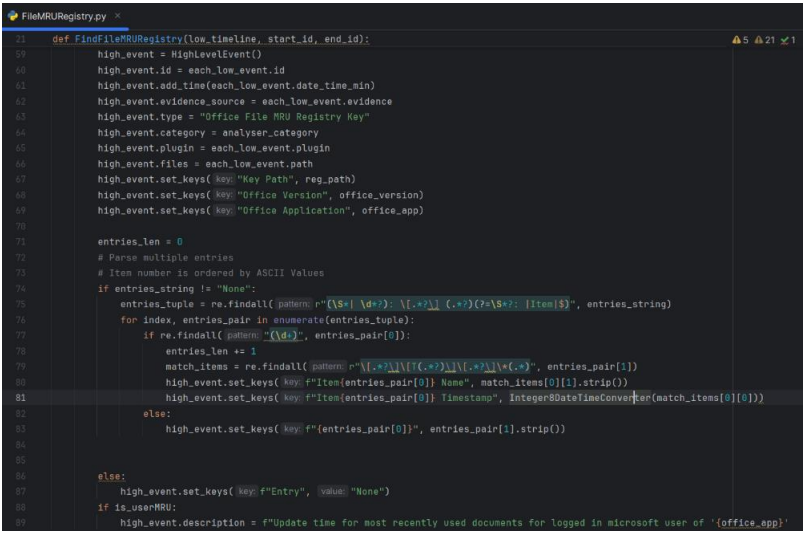
Kode dari salah satu *analyzer* yang dibuat penulis bisa dilihat pada Gambar 4.10, 4.11, 4.12, 4.13, 4.14.



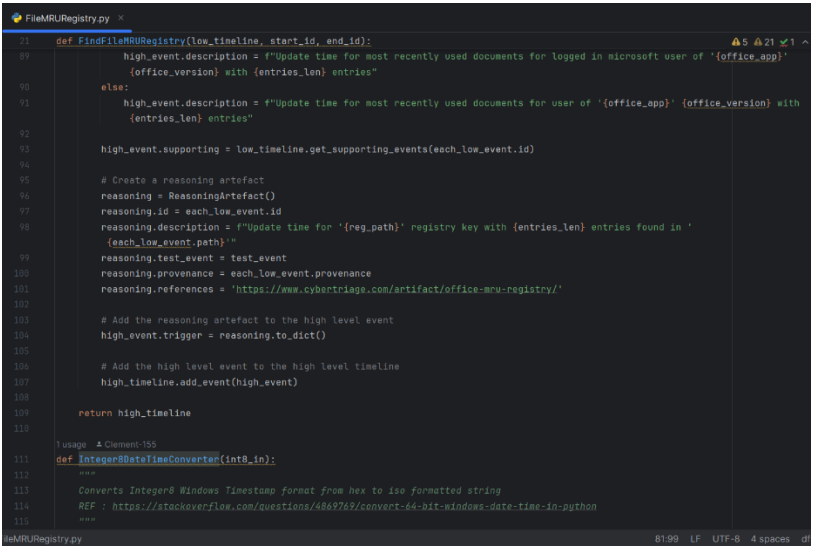
Gambar 4.10 Contoh Kode *Analyzer* 1/5



Gambar 4.11 Contoh Kode *Analyzer* 2/5



Gambar 4.12 Contoh Kode *Analyzer* 3/5



Gambar 4.13 Contoh Kode *Analyzer* 4/5

Gambar 4.14 Contoh Kode *Analyzer* 5/5

[Halaman ini sengaja dikosongkan]

BAB V PENGUJIAN DAN EVALUASI

Pengujian dan evaluasi dilakukan terhadap setiap *analyzer* menggunakan *unit test*. Pada bab ini akan dipaparkan *unit test* yang digunakan untuk setiap *analyzer*.

5.1. Daftar Luaran *Plaso* untuk *Unit Test*

Di bawah ini adalah Tabel 5.1 yang berisi contoh baris *low-level event* dari *Plaso* yang digunakan untuk *unit test* dengan keterangan sumber. Apabila luaran *analyzer* tidak sesuai dan *unit test* gagal, *analyzer* direvisi hingga memenuhi. *Low-level event* yang berasal dari laptop penulis diubah agar nama komputer penulis tidak terlihat. Terdapat pula *low-level event* hasil gabungan agar bisa mencerminkan beberapa kasus sekaligus.

Tabel 5.1 Daftar *Low-Level Event Unit Test*

Analyzer	**Sumber dan *Low-Level Event***
Users.py (*user account* baru dibuat)	Pak Hudan: 2023-12-26 23:30:24.568036+00:00,Content Modification Time,REG,Registry Key,[HKEY_LOCAL_MACHINE\SAM\SAM\Domains\Account\\Users\\Names\\root] (default): [UNKNOWN],(empty),winreg/winreg_default,NTFS:\Windows\System32\config\SAM,-
TimezoneSettings.py (*Timezone Windows*)	Penulis: 2024-07-12T05:52:00.813418+00:00,Content Modification Time**,**REG,[HKEY_LOCAL_MACHINE\System\ControlSet001\Control\TimeZoneInformation] ActiveTimeBias: -420 Bias: -420 DaylightBias: 0 DaylightName: @tzres.dll -561 DynamicDaylightTimeDisabled: 0 StandardBias: 0 StandardName: @tzres.dll -562 TimeZoneKeyName: SE Asia Standard Time,winreg/windows_timezone,NTFS:\Windows\System32\config\SYSTEM,-
NetworkCards.py (Waktu *installation network interface* baru)	Penulis: 2024-04-17T01:16:00.609890+00:00,Content Modification Time,REG,Registry Key,[HKEY_LOCAL_MACHINE\Software\

	Microsoft\Windows NT\CurrentVersion\NetworkCards\2] Description: [REG_SZ] Intel(R) PRO/1000 MT Desktop Adapter ServiceName: [REG_SZ] {9F272040-23C5-42CB-BBB3-EBCA31FB81C8},winreg/winreg_default,NTFS:\Windows\System32\config\SOFTWARE,-
NetworkProfiles.py (Waktu sistem operasi terhubung dengan *network* baru)	Penulis: 2024-07-12T05:50:14.347797+00:00,Content Modification Time,REG,Registry Key,[HKEY_LOCAL_MACHINE\Software\Microsoft\Windows NT\CurrentVersion\NetworkList\Profiles\{400F2E8B-9AB8-4DA6-8705-455176209E17}] Category: [REG_DWORD_LE] 0 DateCreated: [REG_BINARY] (16 bytes) DateLastConnected: [REG_BINARY] (16 bytes) Description: [REG_SZ] Network Managed: [REG_DWORD_LE] 0 NameType: [REG_DWORD_LE] 6 ProfileName: [REG_SZ] Network,winreg/winreg_default,NTFS:\Windows\System32\config\SOFTWARE,-
DeviceInstallation.py (Perangkat terhubung dengan sistem operasi)	Penulis: 2024-08-29T07:56:20.548000+00:00,Added Time,LOG,Setup API Log,Device Install (Hardware initiated) - SWD\WPDBUSENUM_??_USBSTOR#Disk&Ven__USB&Prod__SanDisk_3.2Gen1&Rev_1.00#0101192e17421d310eb90845e69173df47bc6b9ad0c88c0cb0cf7e8d8cc6a15#{53f56307-b6bf-11d0-94f2-00a0c91efb8b} - SUCCESS,text/setupapi,NTFS:\Windows\INF\setupapi.dev.log,- Penulis: 2024-08-29T07:56:22.111000+00:00,End Time,LOG,Setup API Log,Device Install (Hardware initiated) - SWD\WPDBUSENUM_??_USBSTOR#Disk&Ven__USB&Prod__SanDisk_3.2Gen1&Rev_1.00#0101192e17421d310eb90845e69173df47bc6b9ad0c88c0cb0cf7e8d8cc6a15#{53f56307-b6bf-11d0-94f2-00a0c91efb8b} -

	SUCCESS,text/setupapi,NTFS:\Windows\INF\setupapi.dev.log,-
WindowsFirewallDisable.py (*Windows Firewall* dimatikan)	Pak Hudan: 2023-12-26T23:26:57.492844+00:00,Content Modification Time,EVT,WinEVTX,[2082 / 0x0822] Provider identifier: {d1bc9aff-2abf-4d71-9146-ecb2a986eb85} Source Name: Microsoft-Windows-Windows Firewall With Advanced Security Strings: ['2' '1' '4' '00000000' 'No' '1' 'S-1-5-21-4087375726-1105420669-2909453743-1000' 'C:\\Windows\\System32\\dllhost.exe' '0'] Computer Name: WinDev2311Eval Record Number: 894 Event Level: 4,winevtx,NTFS:\Windows\System32\winevt\Logs\Microsoft-Windows-Windows Firewall With Advanced Security%4Firewall.evtx,-
WindowsEventLogCleared.py (Ada jenis *log* dalam *Event Log* yang dihapus pengguna)	Pak Hudan: 2023-12-27T00:40:31.258240+00:00,Creation Time,EVT,WinEVTX,[104 / 0x0068] Provider identifier: {fc65ddd8-d6ef-4962-83d5-6e5cfe9ce148} Source Name: Microsoft-Windows-Eventlog Strings: ['User' 'WINDEV2311EVAL' 'Application' '\\\\WINDEV2311EVAL\\C$\\Users\\User\\Documents\\app-event-log.evtx' '3500' '3096224743817719'] Computer Name: WinDev2311Eval Record Number: 1747 Event Level: 4,winevtx,NTFS:\Windows\System32\winevt\Logs\System.evtx,- Pak Hudan: 2023-11-15T19:15:11.309955+00:00,Creation Time,EVT,WinEVTX,[1102 / 0x044e] Provider identifier: {fc65ddd8-d6ef-4962-83d5-6e5cfe9ce148} Source Name: Microsoft-Windows-Eventlog Strings: ['S-1-5-21-2939114745-2192642559-1429779423-500' 'Administrator' 'WINDEVEVAL' '0x000000000005ce5e' '1148' '2533274790396089'] Computer Name: WinDevEval Record Number: 2577 Event Level:

	4,winevtx,NTFS:\Windows\System32\winevt\Logs\Security.evtx,-
DefaultBrowser.py (*Browser* yang otomatis dibuka sistem)	Penulis: 2024-08-29T07:45:39.210988+00:00,Content Modification Time,REG,Registry Key,[HKEY_CURRENT_USER\Software\Microsoft\Windows\Shell\Associations\UrlAssociations\https\UserChoice] Hash: [REG_SZ] 2F2Buyu+SaM= ProgId: [REG_SZ] MSEdgeHTM,winreg/winreg_default,NTFS:\Users\User\NTUSER.DAT,-
FailedLogin.py (*User* gagal *login*)	Pak Hudan: 2023-12-26T23:33:11.631160+00:00,Creation Time,EVT,WinEVTX,[4625 / 0x1211] Provider identifier: {54849625-5478-4994-a5ba-3e3b0328c30d} Source Name: Microsoft-Windows-Security-Auditing Strings: ['S-1-5-18' 'WINDEV2311EVAL$' 'WORKGROUP' '0x00000000000003e7' 'S-1-0-0' 'root' 'WINDEV2311EVAL' '0xc000006d' '%%2313' '0xc000006a' '2' 'User32 ' 'Negotiate' 'WINDEV2311EVAL' '-' '-' '0' '0x0000000000000de0' 'C:\\Windows\\System32\\svchost.exe' '127.0.0.1' '0'] Computer Name: WinDev2311Eval Record Number: 3332 Event Level: 0,winevtx,NTFS:\Windows\System32\winevt\Logs\Security.evtx,-
USBConnectedRegDeviceClasses.py (Perangkat USB terhubung)	Penulis: 2024-08-29T07:58:04.467136+00:00,Content Modification Time,REG,Registry Key,[HKEY_LOCAL_MACHINE\System\ControlSet001\Control\DeviceClasses\{a5dcbf10-6530-11d2-901f-00c04fb951ed}\##?#USB#VID_1532&PID_0098#5&12c8f4c0&0&2#{a5dcbf10-6530-11d2-901f-00c04fb951ed}\#] (empty),winreg/winreg_default,NTFS:\Windows\System32\config\SYSTEM,-
USBConnectedRegUSB.py (Perangkat USB terhubung)	Penulis: 2024-08-29T07:57:24.603365+00:00,Last Connection Time,REG,USB Registry Key,[HKEY_LOCAL_MACHINE\System\Co

	ntrolSet001\Enum\USB] Product: PID_1234 Serial: 5&12c8f4c0&0&2 Subkey name: VID_ABCD&PID_1234 Vendor: VID_ABCD,winreg/windows_usb_devices,NTFS:\Windows\System32\config\SYSTEM,-
USBConnectedRegUSBSTOR.py (Perangkat penyimpanan USB terhubung)	Penulis: 2024-08-29T07:54:10.113400+00:00,Content Modification Time,REG,Registry Key,[HKEY_LOCAL_MACHINE\System\ControlSet001\Enum\USBSTOR\Disk&Ven_Seagate&Prod_Expansion&Rev_0712\NAAXJ5NB&0] Address: [REG_DWORD_LE] 9 Capabilities: [REG_DWORD_LE] 16 ClassGUID: [REG_SZ] {4d36e967-e325-11ce-bfc1-08002be10318} CompatibleIDs: [REG_MULTI_SZ] [USBSTOR\Disk USBSTOR\RAW GenDisk] ConfigFlags: [REG_DWORD_LE] 0 ContainerID: [REG_SZ] {13b3a4df-ff65-5930-93d3-1ef7378756c2} DeviceDesc: [REG_SZ] @disk.inf %disk_devdesc%;Disk drive Driver: [REG_SZ] {4d36e967-e325-11ce-bfc1-08002be10318}\0001 FriendlyName: [REG_SZ] Seagate Expansion USB Device HardwareID: [REG_MULTI_SZ] [USBSTOR\DiskSeagate_Expansion_____0712 USBSTOR\DiskSeagate_Expansion_____ USBSTOR\DiskSeagate_ USBSTOR\Seagate_Expansion_____0 Seagate_Expansion_____0 USBSTOR\GenDisk GenDisk] Mfg: [REG_SZ] @disk.inf %genmanufacturer%;(Standard disk drives) Service: [REG_SZ] disk,winreg/winreg_default,NTFS:\Windows\System32\config\SYSTEM,-
USBConnectedWinevt.py (Perangkat penyimpanan USB terhubung)	Penulis: 2024-08-29T07:56:20.468109+00:00,Creation Time,EVT,WinEVTX,[1006 / 0x03ee] Provider identifier: {412bdff2-a8c4-470d-8f33-63fe0d8c20e2} Source Name: Microsoft-Windows-Partition Strings: ['1' '8208' '262401' 'false' '0' '0' '0' '512' '61530439680' '7' 'USB' 'SanDisk 3.2Gen1' '1.00'

	'0101192e17421d310eb9' 'Integrated : Bus 0 : Device 0 : Function 9 : Adapter 0 : Port 0' 'USB\\VID_0781&PID_5591\\0101192e17421d310eb90845e69173df47bc6b9ad0c88c0cb0cf7e8d8cc6a15a7dfa000000000000000000000046dcbec3ff1c560091558107ab2a1d42' '-2147483647' '-2147483647' '0' '0' '9' '0' '0' '0' '0' '59681' '0' '{60A28913-5BF8-3861-5443-C194F43168A0}' '{00000000-0000-0000-0000-000000000000}' '{46672950-65DB-11EF-A73C-08002732B5A2}' '{00000000-0000-0000-0000-000000000000}' 'false' '0' '0' '0' '0' '0' '0' None '0' '0' '0' '0' 'false' 'false' 'false' '512' '512' '0' 'false' 'false' 'false' '0' '0' '0' '0' '0' '0' '0' 'false' '0' '524288' '129' '0' 'NULL' '5' 'true' '0' '4' '624' '0000000004000000629753470E64C52E0000 0000000000000000000000000000000000000 0000000000000000000000000000000000000 0000001000000000000000000008000000010 00000000000000C0101000008000062975347 0000000000001000000000000000000000000 0000000000000000000000000000000000000 0000000000000000000000000000000000000 0000000000000000000000000000000000000 0000000000000000000000000000000000000 0000000000000000000000000000000000000 0000000000000000000000000000000000000 0000000000000000000006297534700000000 0000000000000000000000000000000000000 0000000000000000000000000000000000000 0000000000000000000000000000000000000 0000000000000000000000000000000000000 0000000000000000000000000000000000000 0000000000000000000000000000000000000 0000000000000000000000000000000000000 0000000000000629753470000000000000000 0000000000000000000000000000000000000 0000000000000000000000000000000000000 0000000000000000000000000000000000000 0000000000000000000000000000000000000 0000000000000000000000000000000000000 0000000000000000000000000000000000000 0000000000000000000000000000000000000

	000006297534700000000000000000000000 000000000000000000000000000000000000 000000000000000000000000000000000000 000000000000000000000000000000000000 000000000000000000000000000000000000 000000000000000000000000' '512' '33C08ED0BC007C8EC08ED8BE007CBF00 06B90002FCF3A450681C06CBFBB90400B DBE07807E00007C0B0F850E0183C510E2F 1CD1888560055C6461105C6461000B441BB AA55CD135D720F81FB55AA7509F7C1010 07403FE46106660807E10007426666800000 0066FF760868000068007C680100681000B4 428A56008BF4CD139F83C4109EEB14B801 02BB007C8A56008A76018A4E028A6E03C D136661731CFE4E11750C807E00800F848A 00B280EB845532E48A5600CD135DEB9E81 3EFE7D55AA756EFF7600E88D007517FAB 0D1E664E88300B0DFE660E87C00B0FFE66 4E87500FBB800BBCD1A6623C0753B6681F B54435041753281F90201722C666807BB000 06668000200006668080000006653665366556 66680000000006668007C0000666168000007C D1A5A32F6EA007C0000CD18A0B707EB08 A0B607EB03A0B50732E40500078BF0AC3 C007409BB0700B40ECD10EBF2F4EBFD2B C9E464EB002402E0F82402C3496E76616C6 9642070617274697 6F6E207461626C6500 4572726F72206C6F6164696E67206F706572 6174696E672073797374656D004D69737369 6E67206F7065726174696E672073797374656 D000000637B9A6297534700008020210 00CF EFFFF0008000000000040000000000000000 000000000000000000000000000000000000 000000000000000000000000000000000000 00000055AA' '512' 'EB58904D53444F53352E3000024004200200 000000F800003F00FF000008000000000004F E1F000000000000020000001000600000000 000000000000000000800129964C63AA4E4F 204E414D4520202020464154333220202033 C98ED1BCF47B8EC18ED9BD007C8856408 84E028A5640B441BBAA55CD13721081FB5 5AA750AF6C1017405FE4602EB2D8A5640 B408CD137305B9FFFF8AF1660FB6C64066

	0FB6D180E23FF7E286CDC0ED0641660FB7 C966F7E1668946F8837E16007539837E2A00 7733668B461C6683C00CBB0080B90100E82 C00E9A803A1F87D80C47C8BF0AC84C074 173CFF7409B40EBB0700CD10EBEEA1FA7 DEBE4A17D80EBDF98CD16CD196660807 E02000F842000666A00665006536668100001 00B4428A56408BF4CD13665866586658665 8EB33663B46F87203F9EB2A6633D2660FB 74E1866F7F1FEC28ACA668BD066C1EA10 F7761A86D68A56408AE8C0E4060ACCB80 102CD1366610F8274FF81C30002664049759 4C3424F4F544D4752202020200000000000 00000000000000000000000000000000000000 00000000000000000000000000000000000000 00000000000000000000D0A4469736B20657 2726F72FF0D0A507265737320616E79206B6 57920746F20726573746172740D0A0000000 00000000000000000000000000000000000000 00000000000000000000000AC01B90100 0055AA' '0' None '0' None '0' None] Computer Name: WinDev2404Eval Record Number: 15 Event Level: 4 Message string: For internal use only.,winevtx,NTFS:\Windows\System32\win evt\Logs\Microsoft-Windows- Partition%4Diagnostic.evtx,-
LastExecutedPrefetch.py (Terakhir eksekusi)	Penulis: 2024-08- 29T08:00:30.616847+00:00,Last Time Executed,LOG,WinPrefetch,Prefetch [CHROMESETUP.EXE] was executed - run count 4 path hints: \USERS\USER\DOWNLOADS\CHROMESE TUP.EXE hash: 0x5C9EC5EC volume: 1 [serial number: 0xB63032B5 device path: \VOLUME{01da906b2f897af6- b63032b5}],prefetch,NTFS:\Windows\Prefetc h\CHROMESETUP.EXE-5C9EC5EC.pf,-
LastExecutedUserAssist.py (Terakhir eksekusi)	Penulis: 2024-08- 29T07:59:34.847000+00:00,Last Time Executed,REG,UserAssist Registry Key,[HKEY_CURRENT_USER\Software\Mi crosoft\Windows\CurrentVersion\Explorer\Us erAssist\{CEBFF5CD-ACE2-4F4F-9178- 9926F41749EA}\Count] UserAssist entry: 15

	Value name: MSEdge Count: 1 Application focus count: 6 Application focus duration: 946607,winreg/userassist,NTFS:\Users\User\NTUSER.DAT,-
LastExecutedBAM.py (Terakhir eksekusi)	Penulis: 2024-08-29T09:06:01.232572+00:00,Last Time Executed,REG,Background Activity Moderator Registry Key,\Device\HarddiskVolume4\Windows\explorer.exe [S-1-5-21-3206686254-308435427-1198852649-1000],winreg/bam,NTFS:\Windows\System32\config\SYSTEM,-
LastExecutedPCA.py (Terakhir eksekusi)	Plaso Test Data: 2022-11-15T00:00:17.499000+00:00,Last Time Executed,LOG,Program Compatibility Assistant (PCA) Log,[\program files\git\mingw64\bin\git.exe] was executed - Description: git Version: 2.33.0.windows.2 Vendor: the git development community Exit code: Abnormal process exit with code 0x1,winpca_db0,OS:/data/file/PcaGeneralDb0.txt,- Penulis: 2024-08-29T08:00:19.327000+00:00,Last Time Executed,LOG,Program Compatibility Assistant (PCA) Log,[C:\Users\User\Downloads\ChromeSetup.exe] was executed - ,winpca_dic,NTFS:\Windows\appcompat\pca\PcaAppLaunchDic.txt,-
ServiceInstalled.py (*Windows Service* baru)	Penulis: 2024-08-29T07:54:11.402180+00:00,Content Modification Time,EVT,WinEVTX,[7045 / 0x1b85] Provider identifier: {555908d1-a6d7-4695-8e1e-26931d2012f4} Source Name: Service Control Manager Strings: ['WPD File System driver' '\\SystemRoot\\system32\\DRIVERS\\WUDFRd.sys' 'kernel mode driver' 'demand start' None] Computer Name: WinDev2404Eval Record Number: 1999 Event Level: 4 Message string: A service was installed in the

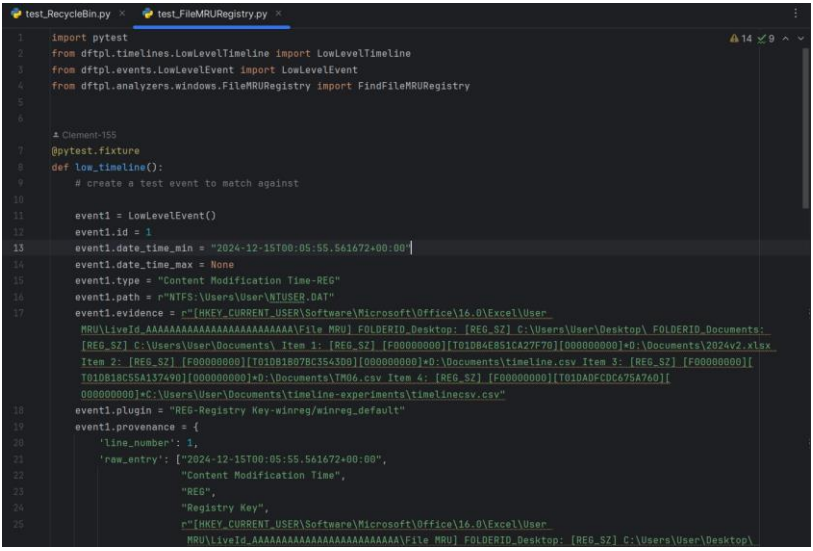
	system.\n\nService Name: WPD File System driver\nService File Name: \SystemRoot\system32\DRIVERS\WUDFRd.sys\nService Type: kernel mode driver\nService Start Type: demand start\nService Account: ,winevtx,NTFS:\Windows\System32\winevt\Logs\System.evtx,-
RunRunOnceRegistry.py (Eksekusi saat *user login*)	Penulis: 2024-04-16T20:18:43.574367+00:00,Content Modification Time,REG,Run/Run Once Registry Key,[HKEY_LOCAL_MACHINE\Software\WOW6432Node\Microsoft\Windows\CurrentVersion\RunOnce] Entries: [],winreg/windows_run,NTFS:\Windows\System32\config\SOFTWARE,- Laptop: 2024-08-13T02:13:18.949571+00:00,Content Modification Time,REG,Run/Run Once Registry Key,[HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\Run] Entries: ['Discord: "C:\\Users\\User\\AppData\\Local\\Discord\\Update.exe" --processStart Discord.exe' 'Docker Desktop: C:\\Program Files\\Docker\\Docker\\Docker Desktop.exe -Autostart' 'Free Download Manager: "C:\\Users\\User\\AppData\\Local\\Softdeluxe\\Free Download Manager\\fdm.exe" --hidden' 'MicrosoftEdgeAutoLaunch_C4A6E550E8B9B5F80F5E289FEBECE979: "C:\\Program Files (x86)\\Microsoft\\Edge\\Application\\msedge.exe" --no-startup-window --win-session-start' 'OneDrive: "C:\\Program Files\\Microsoft OneDrive\\OneDrive.exe" /background' 'Steam: "D:\\Program Files (x86)\\steam.exe" -silent' 'org.openvpn.client: C:\\Program Files\\OpenVPN Connect\\OpenVPNConnect.exe --opened-at-login --

	minimize'],winreg/windows_run,OS:/data/run5/HKCUSave,-
TrustRecordsRegistry.py (Dokumen *Office* dari internet yang dipercayai)	Laptop dan jumlah *key value* dikurangi: 2024-12-09T04:08:42.712388+00:00,Content Modification Time,REG,Registry Key,[HKEY_CURRENT_USER\Software\Microsoft\Office\16.0\PowerPoint\Security\Trusted Documents\TrustRecords] %USERPROFILE%/Documents/Template%20II.pptx: [REG_BINARY] (24 bytes) file:///D:/Documents/Theory.ppt: [REG_BINARY] (24 bytes) file:///D:/Documents/Digital%20Forensics.pptx: [REG_BINARY] (24 bytes) file:///D:/Documents/Drone.pptx: [REG_BINARY] (24 bytes) file:///D:/Documents/Anti.odp: [REG_BINARY] (24 bytes) file:///D:/Documents/Graph%20Database.pptx: [REG_BINARY] (24 bytes),winreg/winreg_default,NTFS:\Users\User\NTUSER.DAT,-
FileMRURegistry.py (Dokumen yang dibuka pengguna)	Laptop dan jumlah key value dikurangi: 2024-12-15T00:05:55.561672+00:00,Content Modification Time,REG,Registry Key,[HKEY_CURRENT_USER\Software\Microsoft\Office\16.0\Excel\User MRU\LiveId_AAAAAAAAAAAAAAAAAAAAAAA\File MRU] FOLDERID_Desktop: [REG_SZ] C:\Users\User\Desktop\ FOLDERID_Documents: [REG_SZ] C:\Users\User\Documents\ Item 1: [REG_SZ] [F00000000][T01DB4E851CA27F70][O00000000]*D:\Documents\2024v2.xlsx Item 2: [REG_SZ] [F00000000][T01DB1B07BC3543D0][O00000000]*D:\Documents\timeline.csv Item 3: [REG_SZ] [F00000000][T01DB18C55A137490][O00000000]*D:\Documents\TM06.csv Item 4: [REG_SZ] [F00000000][T01DADFCDC675A760][O00000000]*C:\Users\User\Documents\timeline-

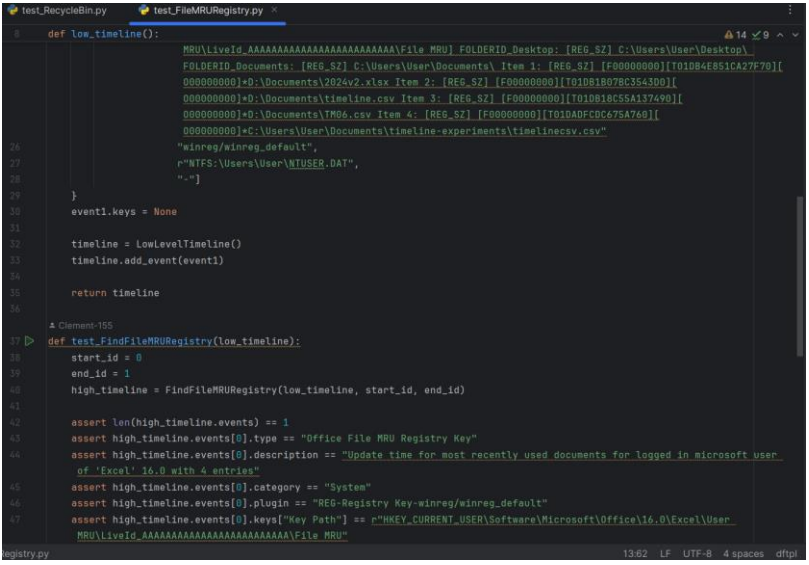
	experiments\timelinecsv.csv,winreg/winreg_default,NTFS:\Users\User\NTUSER.DAT,-
RecycleBin.py (Informasi *file* di *recycle bin*)	Laptop: 2024-12-19T08:28:58.150000+00:00,Content Deletion Time,RECBIN,Recycle Bin,C:\Users\Anonymous\Downloads\vod-2201104836-offset-12344-preview-260x147.jpg,recycle_bin,OS:/data/$IMTP1O2.jpg,-

5.2. Contoh *Unit Test*: *Test_FileMRURegistry.py*

Kode dari salah satu *unit test* yang dibuat penulis bisa dilihat pada Gambar 5.1, 5.2, dan 5.3.

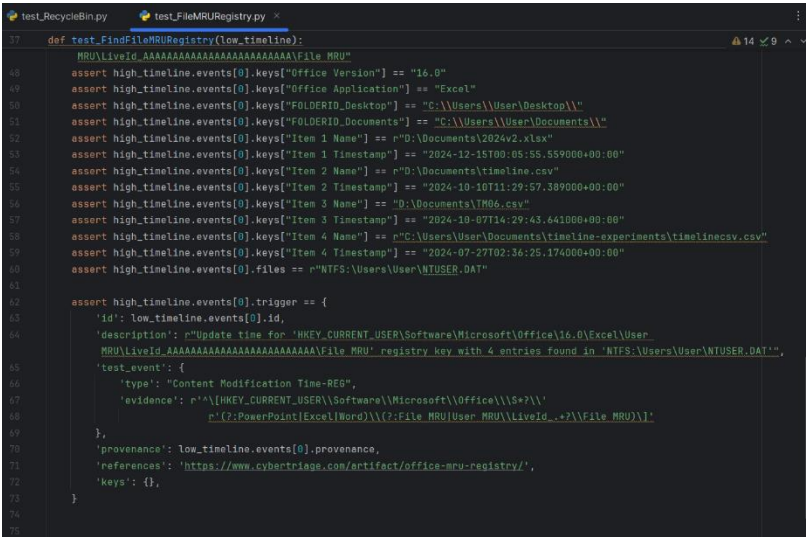


Gambar 5.1 Contoh Kode *Unit Test* 1/3



```
test_RecycleBin.py    test_FileMRURegistry.py
8   def low_timeline():                                                    14  9
                        MRU\LiveId_AAAAAAAAAAAAAAAAAAAAAAAA\File MRU] FOLDERID_Desktop: [REG_SZ] C:\Users\User\Desktop\
                        FOLDERID_Documents: [REG_SZ] C:\Users\User\Documents\ Item 1: [REG_SZ] [F00000000][T01D84E851CA27F70][
                        000000000]*D:\Documents\2024v2.xlsx Item 2: [REG_SZ] [F00000000][T01DB1B07BC3543D0][
                        000000000]*D:\Documents\timeline.csv Item 3: [REG_SZ] [F00000000][T01DB18C55A137490][
                        000000000]*D:\Documents\TM06.csv Item 4: [REG_SZ] [F00000000][T01DADFCDC675A760][
                        000000000]*C:\Users\User\Documents\timeline-experiments\timelinecsv.csv"
                    "winreg/winreg_default",
                    r"NTFS:\Users\User\NTUSER.DAT",
                    "-"]
        }
        event1.keys = None

        timeline = LowLevelTimeline()
        timeline.add_event(event1)

        return timeline

    Clement-155
def test_FindFileMRURegistry(low_timeline):
        start_id = 0
        end_id = 1
        high_timeline = FindFileMRURegistry(low_timeline, start_id, end_id)

        assert len(high_timeline.events) == 1
        assert high_timeline.events[0].type == "Office File MRU Registry Key"
        assert high_timeline.events[0].description == "Update time for most recently used documents for logged in microsoft user
         of 'Excel' 16.0 with 4 entries"
        assert high_timeline.events[0].category == "System"
        assert high_timeline.events[0].plugin == "REG-Registry Key-winreg/winreg_default"
        assert high_timeline.events[0].keys["Key Path"] == r"HKEY_CURRENT_USER\Software\Microsoft\Office\16.0\Excel\User
         MRU\LiveId_AAAAAAAAAAAAAAAAAAAAAAAA\File MRU"
egistry.py                                                    13:62  LF  UTF-8  4 spaces  dftpl
```

Gambar 5.2 Contoh Kode *Unit Test* 2/3



```
test_RecycleBin.py    test_FileMRURegistry.py
37  def test_FindFileMRURegistry(low_timeline):                            14  9
         MRU\LiveId_AAAAAAAAAAAAAAAAAAAAAAAA\File MRU"
        assert high_timeline.events[0].keys["Office Version"] == "16.0"
        assert high_timeline.events[0].keys["Office Application"] == "Excel"
        assert high_timeline.events[0].keys["FOLDERID_Desktop"] == "C:\\Users\\User\\Desktop\\"
        assert high_timeline.events[0].keys["FOLDERID_Documents"] == "C:\\Users\\User\\Documents\\"
        assert high_timeline.events[0].keys["Item 1 Name"] == r"D:\Documents\2024v2.xlsx"
        assert high_timeline.events[0].keys["Item 1 Timestamp"] == "2024-12-15T00:05:55.559000+00:00"
        assert high_timeline.events[0].keys["Item 2 Name"] == r"D:\Documents\timeline.csv"
        assert high_timeline.events[0].keys["Item 2 Timestamp"] == "2024-10-10T11:29:57.389000+00:00"
        assert high_timeline.events[0].keys["Item 3 Name"] == "D:\Documents\TM06.csv"
        assert high_timeline.events[0].keys["Item 3 Timestamp"] == "2024-10-07T14:29:43.641000+00:00"
        assert high_timeline.events[0].keys["Item 4 Name"] == r"C:\Users\User\Documents\timeline-experiments\timelinecsv.csv"
        assert high_timeline.events[0].keys["Item 4 Timestamp"] == "2024-07-27T02:36:25.174000+00:00"
        assert high_timeline.events[0].files == r"NTFS:\Users\User\NTUSER.DAT"

        assert high_timeline.events[0].trigger == {
            'id': low_timeline.events[0].id,
            'description': r"Update time for 'HKEY_CURRENT_USER\Software\Microsoft\Office\16.0\Excel\User
             MRU\LiveId_AAAAAAAAAAAAAAAAAAAAAAAA\File MRU' registry key with 4 entries found in 'NTFS:\Users\User\NTUSER.DAT'",
            'test_event': {
                'type': "Content Modification Time-REG",
                'evidence': r'^\[HKEY_CURRENT_USER\\Software\\Microsoft\\Office\\\S*?\\'
                            r'(?:PowerPoint|Excel|Word)\\(?:File MRU|User MRU\\LiveId_.+?\\File MRU)\]'
            },
            'provenance': low_timeline.events[0].provenance,
            'references': 'https://www.cybertriage.com/artifact/office-mru-registry/',
            'keys': {},
        }
```

Gambar 5.3 Contoh Kode *Unit Test* 3/3

[Halaman ini sengaja dikosongkan]

BAB VI KESIMPULAN DAN SARAN

6.1 Kesimpulan

Kesimpulan yang didapat setelah melaksanakan kegiatan kerja praktik pembuatan *analyzer Windows* untuk program DFTPL di lab NETICS adalah sebagai berikut:

- Terdapat banyak artefak yang bisa digunakan dalam rekonstruksi kejadian pada *Windows*. Beberapa bahkan tidak banyak referensi seperti *event log* 2082 untuk *Windows Firewall*.
- Alat membangun *high-level timeline* yang bersifat *open-source* masih tidak terlalu banyak. Penulis merasa *DFTPL* memiliki banyak potensi yang bisa dikembangkan agar peneliti bisa menganalisis luaran *Plaso* dengan lebih mudah.

6.2 Saran

Saran dalam pembuatan *analyzer Windows* untuk program DFTPL adalah sebagai berikut:

- Banyak *analyzer* yang masih belum dibuat.
- Sebaiknya dibuat *analyzer* yang mendukung format *.JSON* karena ada informasi tambahan yang bisa didapatkan seperti nilai *binary registry Windows*.
- *Supporting events* sebaiknya hanya *events* yang relevan. Untuk *event* terdekat bisa disimpan dalam variabel terpisah.

[Halaman ini sengaja dikosongkan]

DAFTAR PUSTAKA

Carrier, B., & Spafford, E. (2004). Defining event reconstruction of digital crime scenes. *Journal of Forensic Sciences*, *49*(6), JFS2004127-8. https://doi.org/10.1520/jfs2004127

Carvey, H., & Altheide, C. (2011). *Digital Forensics with Open Source Tools*. Syngress. https://doi.org/10.1016/C2009-0-62460-0

Departemen Teknik Informatika. (2024, December). *Laboratorium Teknologi Jaringan dan Keamanan Siber Cerdas—Departemen Teknik Informatika*. https://www.its.ac.id/informatika/id/fasilitas/laboratorium/laboratorium-arsitektur-dan-jaringan-komputer/

Guðjónsson, K. (2010, August 25). *Mastering the Super Timeline With log2timeline*. SANS Institute. https://www.sans.org/white-papers/33438/

Hargreaves, C. (2023, February). *Chrishargreaves / pydft-analysers*. https://bitbucket.org/chrishargreaves/pydft-analysers/src/master/

Hargreaves, C., & Patterson, J. (2012). An automated timeline reconstruction approach for digital forensic investigations. *Digital Investigation*, *9*, S69–S79. https://doi.org/10.1016/j.diin.2012.05.006

Karl-Bridge-Microsoft, v-kents, DCtheGeek, Robo210, mijacobs, & msatranjr. (2021, August). *About Event Logging—Win32 apps*. https://learn.microsoft.com/en-us/windows/win32/eventlog/about-event-logging

Karl-Bridge-Microsoft, v-kents, drewbatgit, DCtheGeek, Robo210, mijacobs, & msatranjr. (2021, August). *Event Logging (Event Logging)—Win32 apps*. https://learn.microsoft.com/en-us/windows/win32/eventlog/event-logging

Lubanovic, B. (2015). *Introducing Python* (1st ed.). O'Reilly Media.

Metz, J. (2024a, December). *GitHub—Log2timeline/plaso: Super timeline all the things*. https://github.com/log2timeline/plaso

Metz, J. (2024b, December). *User's Guide*. https://plaso.readthedocs.io/en/latest/sources/user/Users-Guide.html#the-tools

Python Software Foundation. (2024, December). *The Python tutorial*. https://docs.python.org/3/tutorial/index.html

Stubblebine, T. (2007). *Regular expression pocket reference* (2nd ed.). O'Reilly Media, Inc.

Studiawan, H. (2024, September). *GitHub—Studiawan/dftpl: PyDFT analyzer for plaso CSV file*. https://github.com/studiawan/dftpl/tree/main

[Halaman ini sengaja dikosongkan]

BIODATA PENULIS

Nama	: Christopher Clement Wijaya
Tempat, Tanggal Lahir	: Jakarta, 24 Juni 2024
Jenis Kelamin	: Laki-laki
Telepon	: +6281529311389
Email	: christopher.clement.true@gmail.com

AKADEMIS

Kuliah	: Departemen Teknik Informatika – FTEIC , ITS
Angkatan	: 2021
Semester	: 7 (Tujuh)
IPK	: 3.75