# Aplikasi Pencocokan Pola dengan *Regular Expression* pada *Email Address Harvester*

Muchammad Dimas Sakti Widyatmaja – 13521160[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
[1]13521160@std.stei.itb.ac.id

***Abstract*—Email address harvesting atau email scraping merupakan proses pengumpulan alamat email dari berbagai sumber public, yang dapat digunakan untuk spamming, phishing, atau kegiatan jahat lainnya. Metode ini dapat dilakukan melalui pembelian atau pertukaran daftar email, menggunakan bot pengambil data web, atau serangan direktori. *Regular expression* (disingkat sebagai regex atau regexp) adalah sebuah urutan karakter yang menentukan pola pencocokan pada teks. Biasanya pola tersebut digunakan oleh algoritma pencarian string untuk operasi "cari" atau "cari dan ganti" pada string, atau untuk validasi masukan. Ekspresi reguler banyak digunakan dalam pemrograman untuk memanipulasi, mencari, dan mengganti teks dengan pola yang spesifik. Makalah ini akan membahas penerapan *pattern matching* dengan *regular expression* pada *email address harvesting*.**

***Keywords— Email address harvesting, Regular Expression***

## I. Introduction

*Email address harvesting* adalah proses pengumpulan sejumlah besar alamat email dengan mengikuti beberapa metode yang ilegal. Pengumpulan alamat email ini bertujuan untuk digunakan dalam pengiriman email massal atau *spamming*.

*Email address harvesting* dapat dilakukan melalui beberapa metode, namun salah satu cara yang paling umum adalah dengan membeli atau menukar daftar alamat email yang telah disusun sebelumnya yang tersedia melalui teknik web scraping.

Seseorang dapat menggunakan alat atau perangkat lunak pengumpul alamat email khusus yang dikenal sebagai "*harvesting bots*" atau hanya "*harvesters*" untuk mengumpulkan alamat email yang tersedia publik dari website, forum, dan sumber online lainnya.

Program yang saya buat dimaksudkan untuk membantu analis keamanan dalam tahap awal uji penetrasi untuk memahami jejak pengguna di Internet. Program ini juga berguna bagi pihak perusahaan yang ingin mengetahui apa yang bisa dilihat oleh penyerang tentang web perusahaan mereka.

*Regular expression* (regex) merupakan notasi standar yang mendeskripsikan suatu pola (*pattern*) urutan karakter atau string. Regex digunakan untuk pencocokan string (*string matching*) dengan efisien. Regex sudah terdapat di beberapa bahasa pemrograman dan sudah menjadi standar sehingga penting untuk dipelajari. Di python terdapat modul bawaan re untuk melakukan operasi regex. Dalam penggunaannya, kita dapat menggunakan berbagai karakter dan kelas karakter untuk mencocokkan pola tertentu dalam sebuah string.

Regex dapat digunakan untuk mencocokkan pola dalam sebuah teks dengan cepat dan efisien. Dalam kasus email harvesting, regex dapat digunakan untuk mencocokkan alamat email yang ditemukan dengan daftar alamat email yang sudah disimpan sebelumnya. Jika alamat email yang ditemukan sudah ada dalam daftar, maka regex akan melewatkan alamat email tersebut. Jika tidak ada pada daftar, maka alamat email tersebut akan ditambahkan ke dalam daftar.

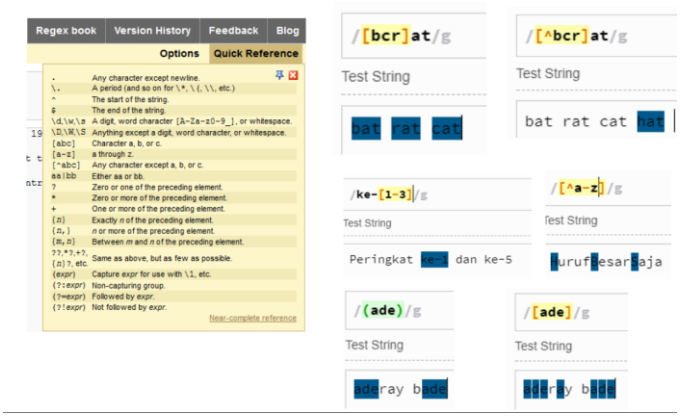
## II. Landasan Teori

### A. Pattern Matching

Pattern matching adalah proses mencari lokasi pertama dari sebuah pola (pattern) dalam sebuah teks (text) yang sangat panjang. Pola tersebut umumnya merupakan sebuah string dengan panjang yang jauh lebih pendek dibandingkan teks yang akan dicari. Tujuan dari pattern matching adalah untuk menemukan kecocokan atau kesesuaian antara pola dan teks, sehingga dapat ditemukan posisi pertama dari pola dalam teks. Sebagai contoh, jika diberikan sebuah teks "the rain in spain stays mainly on the plain" dan pola "main", maka proses pattern matching akan mencari lokasi pertama di dalam teks yang cocok dengan pola "main".

### B. Regular Expression

*Regular expression* (regex) merupakan notasi standar yang mendeskripsikan suatu pola (*pattern*) urutan karakter atau string. Regex digunakan untuk pencocokan string (*string matching*) dengan efisien. Regex sudah terdapat di beberapa bahasa pemrograman dan sudah menjadi standar sehingga penting untuk dipelajari. Di python terdapat modul bawaan re untuk melakukan operasi regex. Dalam penggunaannya, kita dapat menggunakan berbagai karakter dan kelas karakter untuk mencocokkan pola tertentu dalam sebuah string.




**Gambar 2.1.** Notasi Umum Regex
Sumber: https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/String-Matching-dengan-Regex-2019.pdf

Contoh regular expression antara lain, pola regular expression \d{4} merupakan sebuah pola yang digunakan untuk mencari empat karakter digit secara berurutan pada sebuah string. Artinya, metode ini akan mencocokkan setiap empat karakter digit yang ditemukan pada string input dan mengembalikan list yang berisi semua kemunculan tersebut.

### C. *Regular Expression di Python*

Dalam Python, library re digunakan untuk mengimplementasikan regular expression. Terdapat banyak fungsi tersedia pada library tersebut yang berfungsi untuk menjalankan *pattern matching* dengan *regular expression*.

re.compile() adalah sebuah method yang digunakan untuk meng-compile atau mengubah pola regular expression ke dalam sebuah objek regular expression yang bisa digunakan untuk mencocokkan string menggunakan method-match(), search() dan method lainnya yang dijelaskan di bawahnya.

Pattern.search() adalah sebuah method yang digunakan untuk mencari lokasi pertama pada string yang sesuai dengan pola regular expression dan mengembalikan objek yang sesuai. Jika tidak ada lokasi pada string yang sesuai dengan pola regular expression, maka method ini akan mengembalikan None.

Pattern.match() adalah sebuah method yang digunakan untuk mencocokkan pola regular expression dengan karakter-karakter pada awal string, dan mengembalikan objek yang sesuai. Jika tidak ada karakter yang sesuai dengan pola regular expression pada awal string, maka method ini akan mengembalikan.

re.findall() adalah sebuah method yang digunakan untuk mencari semua kemunculan string yang sesuai dengan pola regular expression pada string input dan mengembalikan sebuah list yang berisi semua string yang sesuai.

### D. *Email Address Harvesting*

*Email address harvesting* atau *email scraping* adalah proses pengumpulan alamat email dengan menggunakan bot otomatis dari sumber online, biasanya untuk membangun daftar email untuk serangan siber seperti phishing dan spam. Proses *email scraping* melibatkan pencarian pada halaman web, platform media sosial, dan lokasi online lainnya untuk menemukan alamat email yang kemudian dijadikan satu dalam sebuah database. *Email scraping* sering digunakan oleh pengirim spam atau penjahat siber untuk mengumpulkan alamat email yang valid dan kemudian digunakan untuk mengirimkan spam atau pesan phishing. Proses ini dapat dilakukan secara otomatis dengan menggunakan perangkat lunak khusus yang disebut "*email harvesters*" yang dapat digunakan untuk mengambil alamat email dari berbagai sumber secara efisien. Email scraping adalah kegiatan yang ilegal dan dapat melanggar privasi individu jika dilakukan tanpa izin atau persetujuan dari pemilik alamat email.

## III. IMPLEMENTASI

Berikut merupakan implementasi *pseudocode* dari *email address harvester* sederhana

```
function match_email_pattern(text):
    { Pola email }
    pattern = '\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b'
    matches = cariPola(pattern, text)
    return matches

function crawl_and_scrape(url):
    { Mengambil konten HTML dari website }
    page = requests.get(url)
    html_content = page.content.decode('utf-8')

    { Mencocokkan pola email dalam HTML menggunakan algoritma Boyer-Moore }
    matches = match_email_pattern(html_content)
    jika matches != []:
        cetak("List email yang ditemukan di", url + ":")
        untuk setiap email dalam matches:
            cetak(email)
    lainnya:
        cetak("Tidak ada email yang ditemukan di", url)
    cetak()

    semua_email.update(matches)
```

```
      soup = BeautifulSoup(html_content, 'html.parser')


      links = set()

      untuk setiap link dalam soup.find_all('a'):
          href = link.get('href')
          jika href:
              jika href.dimulai_dengan(start_url):
                  links.tambahkan(href)
              elif href.dimulai_dengan(path):
                  links.tambahkan(domain + href)


      untuk setiap link dalam links:
          crawl_and_scrape(link)


  start_url = input("Masukkan tautan halaman yang ingin dicari alamat emailnya: ")
  parsed_url = urlparse(start_url)
  path = parsed_url.path
  domain = urlunparse((parsed_url.scheme, parsed_url.netloc, '', '', '', ''))


  crawled_links = set()
  semua_email = set()


  crawl_and_scrape(start_url)


  jika semua_email != []:
      cetak("Email yang ditemukan secara keseluruhan:")
      untuk setiap email dalam semua_email:
          cetak(email)
  lainnya:
    cetak("Tidak ada email yang secara keseluruhan ditemukan pada", start_url)
```

Berikut merupakan implementasi dari pseudocode tersebut dalam kode python.

```python
import requests
from bs4 import BeautifulSoup
import re
from urllib.parse import urlparse, urlunparse
```

***Gambar 3.1.*** *Import Library untuk Program*

*Sumber: Dokumen pribadi.*

*Import* semua *library* yang diperlukan. *Library-library* yang diperlukan yaitu requests, untuk melakukan http request ke halaman yang ingin di-*scrape*. re, digunakan untuk mendapatkan fungsi-fungsi bawaan python yang sudah ada untuk melakukan operasi-operasi *regular expression*. Beautiful Soup dari bs4 berguna untuk melakukan parsing html untuk mencari href pada hasil teks hasil http request. urllib.parse digunakan untuk memparsing url menjadi domain dan pathnya.

```python
def match_email_pattern(text):
    # pola email
    pattern = r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b'
    matches = set(re.findall(pattern, text))
    matches = re.findall(pattern, text)
    if (matches != []):
        matches = set(matches)

    return matches
```

***Gambar 3.2.*** *Fungsi match_email_pattern*

*Sumber: Dokumen pribadi.*

Fungsi match_email_pattern menerima input berupa string teks yang nantinya berisi teks html dari web yang ingin di-*scrape*. Pada fungsi ini, didefinisikan *regular expression* yang akan digunakan untuk mengidentifikasi Apakah sebuah string adalah sebuah email atau bukan, Pola tersebut yaitu:

```
\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b
```

Pada kode tersebut, regular expression yang digunakan adalah r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b'.

Pola ini digunakan untuk mencocokkan alamat email dalam sebuah teks dengan format standar yang umum digunakan. Berikut adalah penjelasan dari pola regular expression tersebut:

“\b”, Karakter boundary atau batas kata.

“[A-Za-z0-9._%+-]+”, Kumpulan karakter alphanumeric, tanda titik, underscore, persen, plus, dan dash. Tanda plus menandakan bahwa karakter tersebut harus muncul satu atau lebih kali.

“@”, Karakter at atau tanda @.

“[A-Za-z0-9.-]+”, Kumpulan karakter alphanumeric, tanda titik, dan dash. Tanda plus menandakan bahwa karakter tersebut harus muncul satu atau lebih kali.

“\.”, Escaped dot atau tanda titik yang diberi backslash. Tanda titik sendiri memiliki arti khusus dalam regular expression, sehingga harus di-escape dengan backslash untuk mencocokkan tanda titik yang sebenarnya.

“[A-Z|a-z]{2,}”, Kumpulan karakter huruf besar atau kecil. Tanda kurung kurawalmengindikasikan bahwa karakter tersebut harus muncul minimal 2 kali. Tanda | menandakan bahwa karakter tersebut dapat berupa huruf besar atau kecil.

“\b”, Karakter *boundary* atau batas kata.

Jadi, pola regular expression tersebut akan mencocokkan setiap kemunculan alamat email dalam sebuah teks dengan format standar yang terdiri dari karakter *alphanumeric*, tanda titik, *underscore*, persen, plus, *dash*, dan tanda @, diikuti oleh domain yang valid dengan minimal dua karakter huruf besar atau kecil. Karakter *boundary* di awal dan akhir pola menandakan bahwa pola tersebut harus ditemukan sebagai kata yang utuh.

Setelah mendefinisikan pola ekspresi regulernya, re.findall digunakan untuk mencari semua teks yang memenuhi pola tersebut dari parameter text.

```
# Fungsi untuk melakukan web crawling dan email scraping
def crawl_and_scrape(url, crawled_links, all_emails, domain, path):
    # Memeriksa apakah tautan sudah di-crawl sebelumnya
    if url in crawled_links:
        return

    # Menambahkan tautan ke dalam crawled_links
    crawled_links.add(url)

    # Mengambil konten HTML dari website
    page = requests.get(url)
    try:
        html_content = page.content.decode('utf-8')
    except UnicodeDecodeError:
        print("UnicodeDecodeError: Tidak dapat mendecode konten HTML dari", url)
        return

    # Mencocokkan pola email dalam HTML menggunakan algoritma Boyer-Moore
    matches = match_email_pattern(html_content)
    if (matches != set()):
        # Menampilkan email yang cocok di layar
        print(f"List email yang ditemukan di {url}: ")
        # print(matches)
        for email in matches:
            print(email)
    else:
        print(f"Tidak ada email yang ditemukan di {url}")
    print()
```

***Gambar 3.3.*** *Fungsi crawl_and_scrape Bagian Satu*

*Sumber: Dokumen pribadi.*

```
    # Menambahkan email yang ditemukan ke dalam variabel all_emails
    all_emails.update(matches)

    # Mem-parsing HTML dengan BeautifulSoup
    soup = BeautifulSoup(html_content, 'html.parser')

    # Menemukan semua tautan (link) pada halaman
    links = set()
    for link in soup.find_all('a'):
        href = link.get('href')
        # print(start_url)
        if href:
            if href.startswith(start_url):
                links.add(href)
            elif href.startswith(path) :
                links.add(domain + href)

    # Melakukan web crawling ke halaman-halaman yang ditemukan
    for link in links:
        crawl_and_scrape(link, crawled_links, all_emails, domain, path)
```

***Gambar 3.3.*** *Fungsi crawl_and_scrape Bagian Dua*

*Sumber: Dokumen pribadi.*

Fungsi crawl_and_scrape merupakan fungsi yang berperan *web crawling* dan *email scraping* pada halaman yang dimasukkan dalam sebagai argumen fungsi. Pertama-tama, fungsi ini akan memeriksa apakah tautan yang diberikan sudah pernah di-*crawl* sebelumnya atau belum. Jika sudah, maka fungsi akan langsung mengembalikan nilai tanpa melakukan apapun. Namun jika tautan tersebut belum pernah di-*crawl* sebelumnya, maka tautan tersebut akan ditambahkan ke dalam variabel crawled_links agar tidak di-*crawl* lagi di lain waktu.

Selanjutnya, fungsi akan mengambil konten HTML dari halaman web menggunakan *library* requests. Jika terdapat kesalahan dalam proses pengambilan konten, seperti konten tidak bisa di-decode, maka fungsi akan menampilkan pesan kesalahan dan mengembalikan nilai.

Setelah itu, fungsi akan mencocokkan pola email pada konten HTML menggunakan regular expression sesuai dengan pola yang telah ditentukan sebelumnya. Hasil pencocokkan akan disimpan ke dalam variabel matches.

Jika fungsi berhasil menemukan email dalam konten HTML, maka email tersebut akan ditampilkan di layar. Jika tidak ditemukan, maka fungsi akan menampilkan pesan bahwa tidak ada email yang ditemukan pada halaman tersebut. Selanjutnya, email yang ditemukan akan ditambahkan ke dalam variabel all_emails sehingga dapat ditampilkan secara keseluruhan setelah proses crawling selesai.

Fungsi kemudian akan mem-parsing HTML menggunakan library BeautifulSoup untuk menemukan semua tautan (link) pada halaman tersebut. Setelah menemukan tautan, fungsi akan melakukan web crawling ke halaman-halaman yang ditemukan dengan memanggil kembali fungsi crawl_and_scrape untuk setiap tautan yang ditemukan. Proses crawling dan scraping hanya akan dilakukan kepada tautan dengan awalan start_url atau yang memiliki path sama dengan path dari start_url.

Proses crawling dan scraping akan berulang pada setiap halaman baru yang ditemukan, dan akan terus berjalan sampai tidak ada lagi tautan baru yang ditemukan. Setelah proses crawling selesai, fungsi akan menampilkan semua email yang ditemukan secara keseluruhan pada halaman-halaman yang di-crawl. Jika tidak ditemukan email sama sekali, maka fungsi akan menampilkan pesan bahwa tidak ada email yang ditemukan pada halamantersebut. Dengan demikian, fungsi crawl_and_scrape sangat bermanfaat dalam membantu pengguna untuk menemukan alamat email yang tersembunyi pada halaman web tertentu dengan cara yang cepat dan efektif.

```python
if __name__ == "__main__":
    # URL website yang akan di-crawl
    start_url = input("Masukkan tautan halaman yang ingin dicari alamat emailnya: ")
    parsed_url = urlparse(start_url)
    path = parsed_url.path
    domain = urlunparse((parsed_url.scheme, parsed_url.netloc, '', '', '', ''))

    # Set untuk menyimpan tautan yang sudah di-crawl
    crawled_links = set()

    # Set untuk menyimpan semua email yang ditemukan
    all_emails = set()

    # Memulai web crawling dan email scraping
    crawl_and_scrape(start_url, crawled_links, all_emails, domain, path)

    # Menampilkan semua email yang ditemukan
    if (all_emails != set()):
        # Menampilkan email yang cocok di layar
        print("Email yang ditemukan secara keseluruhan: ")
        # print(all_emails)
        for email in all_emails:
            print(email)
    else:
        print(f"Tidak ada email yang secara keseluruhan ditemukan pada {start_url}")
```

***Gambar 3.3.*** *Fungsi utama*

*Sumber: Dokumen pribadi.*

Pada blok fungsi utama tersebut, pengguna akan diminta untuk menginput url yang ingin di-*scrape* alamat-alamat emailnya. Lalu variabel crawled_links dan all_emails juga akan diinisiasi dengan set(). Setelah crawl_and_scrape dipanggil, hasil dari *email address harvesting* akan dicetak ke layar.

## IV. Uji Coba

Berikut merupakan hasil uji coba program *email address harvesting*.

### *A. Percobaan pada https://www.brin.go.id*

```
Masukkan tautan halaman yang ingin dicari alamat emailnya: https://www.brin.go.id/
List email yang ditemukan di https://www.brin.go.id/:
ppid@brin.go.id

List email yang ditemukan di https://www.brin.go.id/agenda:
ppid@brin.go.id

List email yang ditemukan di https://www.brin.go.id/announcement:
ppid@brin.go.id

List email yang ditemukan di https://www.brin.go.id/news:
ppid@brin.go.id

Email yang ditemukan secara keseluruhan:
ppid@brin.go.id
```

***Gambar 3.3.*** *Percobaan scrapper pada https://www.brin.go.id/*

*Sumber: Dokumen pribadi.*

Pada percobaan pertama di https://www.brin.go.id/, *crawler* menemukan empat tautan yang terhubung ke https://www.brin.go.id/, yaitu tautan dengan path /, /agenda, /announcement, dan /news. Pada keempat path tersebut ditemukan email yang sama jadi hasil email *address harvesting* pada https://www.brin.go.id/ adalah mendapatkan email ppid@brin.go.id.

### *B. Percobaan pada https://www.telkom.co.id/sites/about-telkom/id_ID/page/sites/about-telkom/id_ID/page/kontak-detail-193/kontak-detail-193*

```
PS D:\Kuliah\Sem_4\Stima\Makalah> python emailScrapper.py
List email yang ditemukan di https://www.telkom.co.id/sites/about-telkom/id_ID/page
/kontak-detail-193:
investor@telkom.co.id
occ@telkom.co.id
customercare@telkom.co.id
digitalsolution@telkom.co.id
corporate_comm@telkom.co.id
c4@telkom.co.id
cc-dbs@telkom.co.id

Email yang ditemukan secara keseluruhan:
investor@telkom.co.id
occ@telkom.co.id
customercare@telkom.co.id
digitalsolution@telkom.co.id
corporate_comm@telkom.co.id
c4@telkom.co.id
cc-dbs@telkom.co.id
```

***Gambar 3.3.*** *Percobaan scrapper pada https://www.telkom.co.id/sites/about-telkom/id_ID/page/sites/about-telkom/id_ID/page/kontak-detail-193/kontak-detail-193*

*Sumber: Dokumen pribadi.*

Untuk percobaan di https://www.telkom.co.id/sites/about-telkom/id_ID/page/sites/about-telkom/id_ID/page/kontak-detail-193/kontak-detail-193, *crawler* menemukan hanya satu tautan yaitu start_url sendiri. Pada url tersebut ditemukan beberapa email seperti investor@telkom.co.id, occ@telkom.co.id, customercare@telkom.co.id, digitalsolution@telkom.co.id, corporate_comm@telkom.co.id, c4@telkom.co.id, dan cc-dbs@telkom.co.id.

### *C. Percobaan pada https://bri.co.id/*

```
Masukkan tautan halaman yang ingin dicari alamat emailnya: https://bri.co.id/
Tidak ada email yang ditemukan di https://bri.co.id/

Tidak ada email yang secara keseluruhan ditemukan pada https://bri.co.id/
```

***Gambar 3.3.*** *Percobaan scrapper pada https://bri.co.id/*

*Sumber: Dokumen pribadi.*

Untuk percobaan di https://bri.co.id/, *crawler* menemukan hanya satu tautan yaitu start_url sendiri. Pada url tersebut tidak ditemukan satupun email.

## V. Kesimpulan

Regular expression dapat diterapkan untuk mengenali pola pada program *email address harvester*. Program *email address harvester* ini dapat memiliki banyak manfaat antara lain, Dalam bidang penelitian dan analisis data, *email address harvester* dapat digunakan untuk mengumpulkan data dan mengidentifikasi pola-pola tertentu dalam email. *email address harvester* dapat digunakan untuk mengumpulkan alamat email calon pelanggan atau prospek yang dapat digunakan dalam kampanye pemasaran dan penjualan. *email address harvester* juga dapat digunakan oleh perusahaan atau organisasi untuk melakukan audit keamanan, melacak aktivitas yang mencurigakan, atau mengidentifikasi potensi pelanggaran data. Akan tetapi, program ini juga dapat disalahgunakan untuk *spamming* dan *phising*. Oleh karena itu, Penulis berharap, program ini dapat digunakan sewajarnya sesuai dengan tujuan awal pembuatan.

## VI. Ucapan Terima Kasih

Penulis mengucapkan rasa syukur dan terima kasih kepada Tuhan Yang Maha Esa atas semua anugerah-Nya, termasuk kesehatan dan kesempatan untuk menyelesaikan makalah ini. Selain itu, penulis juga ingin menyampaikan penghargaan yang setinggi-tingginya kepada semua pihak yang telah membantu dalam proses penyelesaian makalah ini, antara lain:

1. Dr. Nur Ulfa Maulidevi, S.T, M.Sc. selaku dosen pengampu mata kuliah IF2211 Strategi Algoritma K02.
2. Dr. Ir. Rinaldi Munir, M.T. yang telah memberikan kontribusi melalui buku dan materi yang diacu dalam makalah ini.
3. Semua individu atau kelompok yang telah berkontribusi dalam pembuatan source code dan modul open-source yang diacu dalam makalah ini.

## VII. References

[1] https://informatika.stei.itb.ac.id/~rinaldi.munir/Stmik/2018-2019/String-Matching-dengan-Regex-2019.pdf

[2] https://docs.google.com/document/d/1ls6h1A6m-Zhzw6e5eriwMNUAG0D1iwL-eVmVMS2XQoc/edit

[3] https://www.guru99.com/python-regular-expressions-complete-tutorial.html

## VIII. PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 22 Mei 2023

Muchammad Dimas Sakti Widyatmaja
13521160