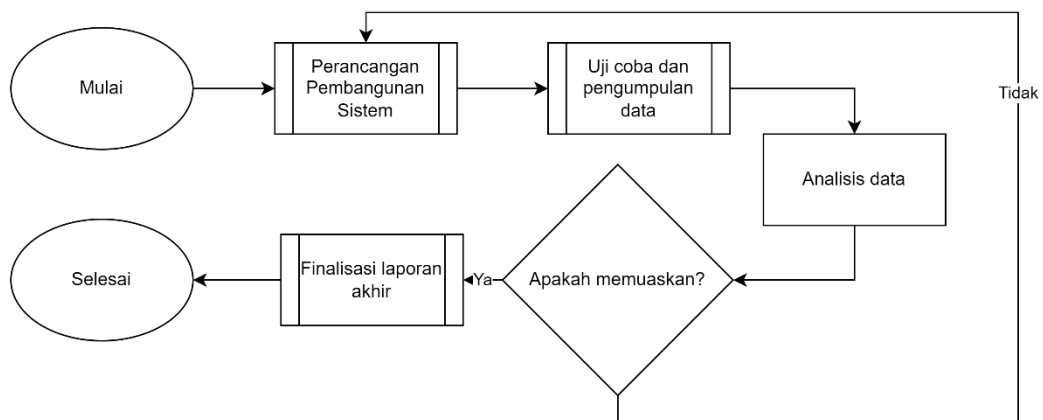


BAB III METODE PENELITIAN

3.1. Alur Penelitian

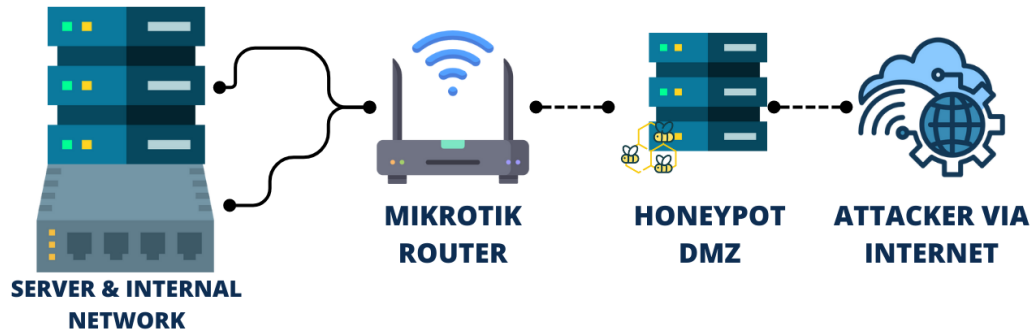
Pertama-tama, sebelum memulai penelitian, melakukan studi literatur agar mendapatkan informasi-informasi terkait yang akan diteliti. Selanjutnya adalah merancang dan membangun prototipe sistem, menggunakan mesin *virtual* dan beberapa perangkat keras asli, proses ini merupakan tahapan pertama pada Gambar 3. Setelah itu, pengujian dilakukan pada prototipe tersebut, lalu direkam menggunakan penangkap layar. Untuk pengumpulan data, terbagi atas dua metode pengumpulan data, yaitu pengumpulan data *UDP* dan pengumpulan data *TCP*. Pengumpulan data *UDP* dikumpulkan dengan cara merekam sumber daya penggunaan Mikrotik. Data pertama diambil pada detik ke-10 lalu data selanjutnya diambil setiap 5 detik sekali, hingga ke detik ke-30. Untuk pengumpulan data *TCP*, dilakukan melalui pemantauan grafik menggunakan *wireshark*, dari rentang 0 hingga 200, proses ini merupakan tahap kedua pada Gambar 3. Data yang telah dikumpulkan ditampung, setelah itu dianalisis, dan diolah menggunakan algoritma *TOPSIS entropy*. Jika data yang dikumpulkan tidak sesuai, atau tidak memenuhi harapan, maka tinjau ulang tahap perancangan, proses ini merupakan tahap ketiga pada Gambar 3. Setelah data yang dikumpulkan memenuhi harapan, maka selanjutnya adalah proses pembuatan grafik, hingga ke penyelesaian laporan akhir, bagian ini merupakan tahap terakhir pada Gambar 3.



Gambar 3. Diagram Alir Penelitian

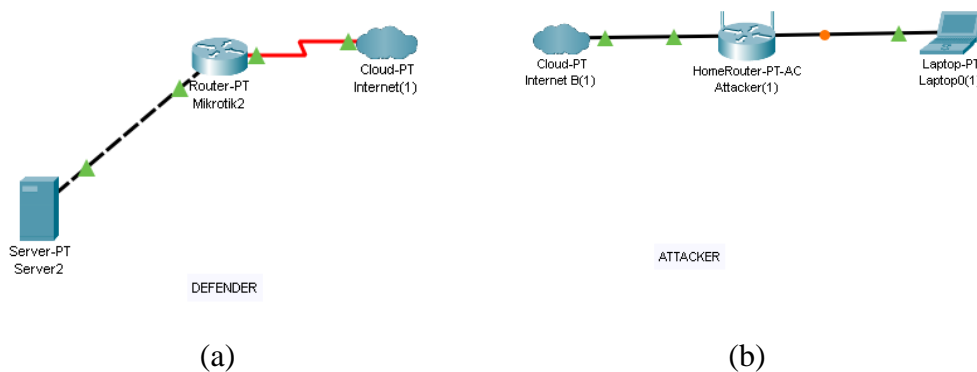
3.2. Desain Infrastruktur Sistem

Topologi jaringan yang akan dibuat akan terdiri dari perangkat *server* untuk *honeypot*, sebuah *router* yang *firewall* nya dapat diatur dengan leluasa, dalam percobaan ini menggunakan Mikrotik, dan terhubung ke internet, seperti pada Gambar 4. Umumnya penyerangan akan melalui internet lain, dan dapat terdiri dari *Botnet* yang isinya bisa jadi terdiri dari *PC* atau *smartphone*, bahkan *IoT*.



Gambar 4. Keseluruhan Sistem Pengamanan Jaringan yang Diharapkan

Pertama-tama akan diuji coba terlebih dahulu, penyerangan tanpa proteksi apapun seperti pada Gambar 5, lalu selanjutnya akan diproteksi melalui metode-metode yang akan digunakan, yaitu metode *Drop*, *Reject*, *Tarpit*, dan *Redirect*. Metode *Drop*, *Reject*, dan *Tarpit*, topologinya akan sama seperti pada Gambar 4, akan tetapi yang membedakan adalah cara proteksi oleh *firewall* seperti pada gambar diagram sistem Gambar 7.

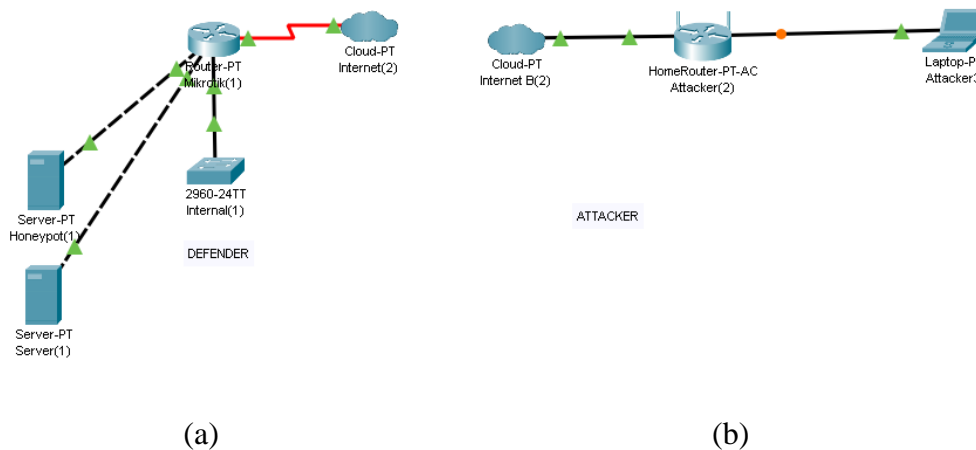


Gambar 5. Sistem Pengamanan Jaringan Tanpa Proteksi

a) Gambaran rancangan jaringan internal dengan pengamanan firewall Mikrotik, dan b) Gambaran simulasi penyerangan.

Selanjutnya akan diuji coba menggunakan cara *redirect* ke *honeypot* seperti pada Gambar 6. Dengan cara ini besar harapannya bahwa penyerang tidak

menyadari bahwa dirinya akan menyadari serangannya tidak berhasil, maka akan terus menyerang *honeypot* yang memang di desain untuk diserang. Lalu terakhir, percobaannya akan menggunakan *DMZ* seperti pada Gambar 4. Dimana *server* tersebut akan diekspos ke *public*, yang berguna agar jaringan *internal* terjaga agar tidak terekspos keluar. Pada setiap tahapan, akan didokumentasikan, penggunaan *CPU*, *memory*, dan *traffic* pada *Tx* maupun *rxnya*.

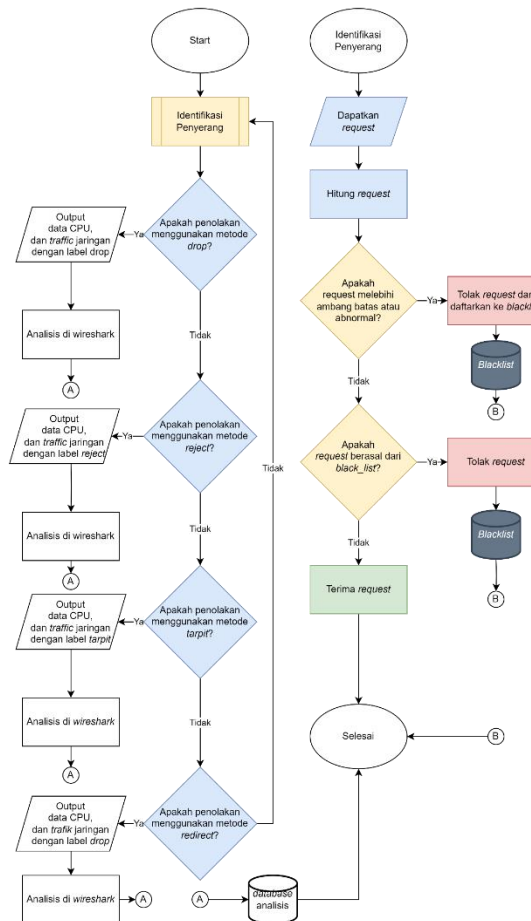


Gambar 6. Pengamanan Jaringan Melalui Metode *Redirect* ke *Honeypot*.

a) Gambaran rancangan jaringan internal dengan pengamanan firewall Mikrotik dan Honeypot, dan b) Gambaran simulasi penyerangan.

3.3. Diagram Alir *Firewall*

Langkah pertama adalah memberikan perintah terhadap *firewall* Mikrotik untuk mendeteksi dan mengklasifikasi serangan-serangan yang datang ke *server*. Sebelumnya akan dipastikan terlebih dahulu bahwa serangan yang datang telah terdaftar di *database blacklist*, jika terdaftar maka akan langsung ditolak. Sementara itu, jika belum terdaftar di *blacklist*, maka akan ditinjau terlebih dahulu, apakah serangan ini layak untuk didaftarkan sebagai bagian dari *blacklist*. Setelah serangan di klasifikasi, langkah selanjutnya adalah menolak serangan tersebut. Dengan metode yang telah disebutkan, yaitu mencoba menggunakan *Drop*, *Reject*, *Tarpit*, dan *redirect*. Setelah dilakukan percobaan tersebut, maka selanjutnya simpan data yang berkaitan dengan metode tersebut. Data tersebut selanjutnya akan ditinjau dan dianalisis, lalu dibandingkan dengan setiap metode yang telah dikumpulkan, setelah itu hasilnya akan diimplementasikan pada setiap kasus umum yang menyerang *server*. Seluruh proses ini terlihat pada Gambar 7.

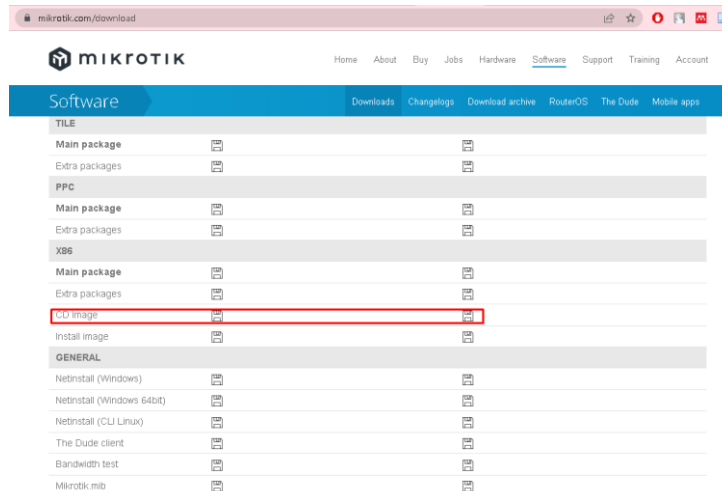
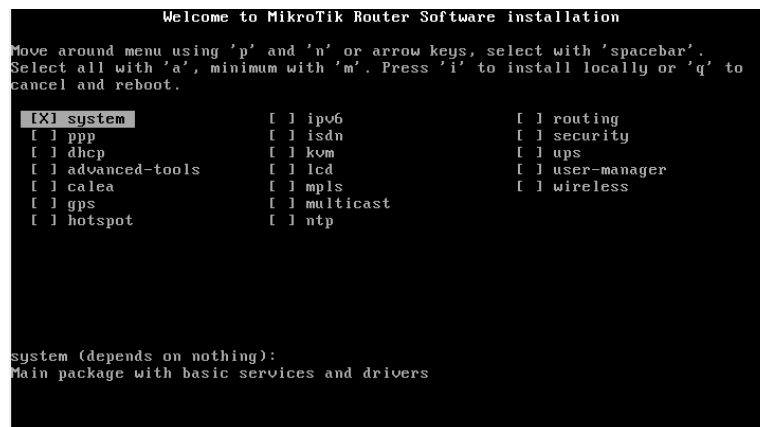


Gambar 7. Diagram Alir Pengamanan Menggunakan *Firewall* Mikrotik

3.4. Instalasi Mikrotik Router OS

Pertama-tama untuk melakukan percobaan ini memerlukan sistem operasi Mikrotik. Untuk mendapatkan sistem operasi ini dapat membeli seperangkat Mikrotik, atau menggunakan Mikrotik *Router OS* yang dipasang ke komputer. Pada percobaan kali ini akan menggunakan kedua sistem tersebut, yang dipasang ke komputer dan perangkat Mikrotik.

Untuk melakukan prosedur instalasi Mikrotik, dapat menggunakan *cd* atau *flash drive* yang telah diintegrasikan dengan *file router os*. Untuk mendapatkan *file* tersebut dapat dikunjungi “Mikrotik.com/download” seperti pada Gambar 8. Setelah itu *booting* ke Mikrotik, lalu pilihlah pilihan yang sesuai dengan kebutuhan. Pada percobaan ini akan mencoba memilih semua dengan tekan a pada *keyboard* lalu tekan i untuk melakukan proses instalasi seperti pada Gambar 9.

Gambar 8. Tampilan *Download* Mikrotik ISO

(a)



(b)

Gambar 9. Instalasi Mikrotik

a) Gambaran sebelum memilih paket instalasi. b) Gambaran setelah memilih paket instalasi.

Setelah melakukan instalasi, kunjungi akun Mikrotik pada “https://Mikrotik.com/Client/login” dengan akun yang telah dimiliki. Lalu dapatkan lisensi sesuai yang dibutuhkan, pada percobaan kali ini menggunakan lisensi *demo*. Untuk mendapatkan lisensi *demo*, pada halaman akun, tekan *make a demo key* lalu masukan *software-id* seperti pada Gambar 10.

Gambar 10. Membuat *Demo Key* Mikrotik

Melalui prosedur tersebut, lalu akan diarahkan ke halaman lisensi seperti pada Gambar 11. Untuk memudahkan ada baiknya, menghubungkan nya terlebih dahulu dengan jaringan, agar dapat menggunakan *winbox*. Untuk mencapai langkah ini, hubungkan *ether1* ke *port* yang sesuai. Misalnya pada percobaan kali ini menggunakan *ether1* 1, lalu ketik pada Mikrotik “*ip DHCP-Client add interface=ether1 disabled=no*”. Setelah mendaftarkan *ip* Mikrotik menggunakan *DHCP-Client*, cek informasi *ip*-nya menggunakan “*ip DHCP-Client Print*” seperti terlihat pada Gambar 12. Lalu daftarkan lisensi pada Gambar 11, melalui *winbox* seperti pada Gambar 13.

Edit software key 8PKT-ULRP!

Software ID	8PKT-ULRP	Fix Software ID
License	<pre> -----BEGIN MIKROTIK SOFTWARE KEY----- J50ILbQVguv1t0g8PSgpGNg9gjkv1ae4yBHiFb8hVb9 B4p31u0Zs0a28n2j25F5JSPRFctNBzMCAsC+HSZ1PA== -----END MIKROTIK SOFTWARE KEY----- </pre>	
Board type	x86 system	
License Level	Level 1 (Demo)	

Note: If this key is for x86 system, you are allowed to change 2 characters in Software ID for free if you made a typing mistake and the key is not upgraded yet. Please check and fix software ID before enabling CF or upgrading, otherwise you will not be able to change it later. You will be charged accordingly when upgrading.

[Back](#)

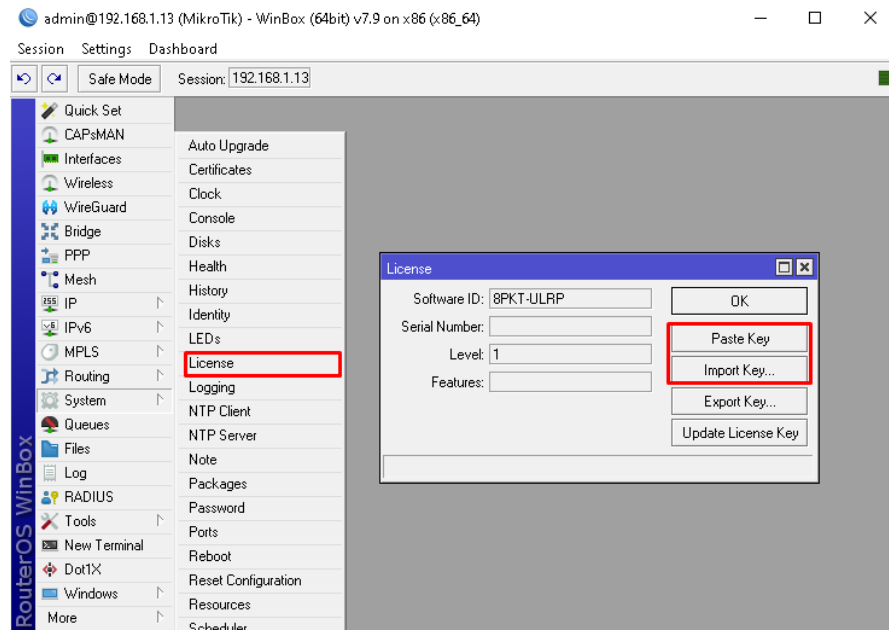
Gambar 11. Contoh Lisensi Mikrotik

Heru Purnama, 2023

**ANALISIS METODE PENGAMANAN PADA PENGGUNAAN
FIREWALL MIKROTIK UNTUK KEAMANAN JARINGAN KOMPUTER**

Universitas Pendidikan Indonesia | repository.upi.edu | perpustakaan.upi.edu

```
[admin@MikroTik] > ip dhcp-client/ add interface=ether1 disabled=no
[admin@MikroTik] > ip dhcp-client/ print
Columns: INTERFACE, USE-PEER-DNS, ADD-DEFAULT-ROUTE, STATUS, ADDRESS
# INTERFACE USE-PEER-DNS ADD-DEFAULT-ROUTE STATUS ADDRESS
0 ether1 yes yes bound 192.168.1.13/24
line 3 of 3> _
```

Gambar 12. Pembuatan dan *Print Status DHCP-Client*

Gambar 13. Menambah Lisensi pada Mikrotik

3.5. Instalasi *Honeypot* menggunakan *Ubuntu Server*

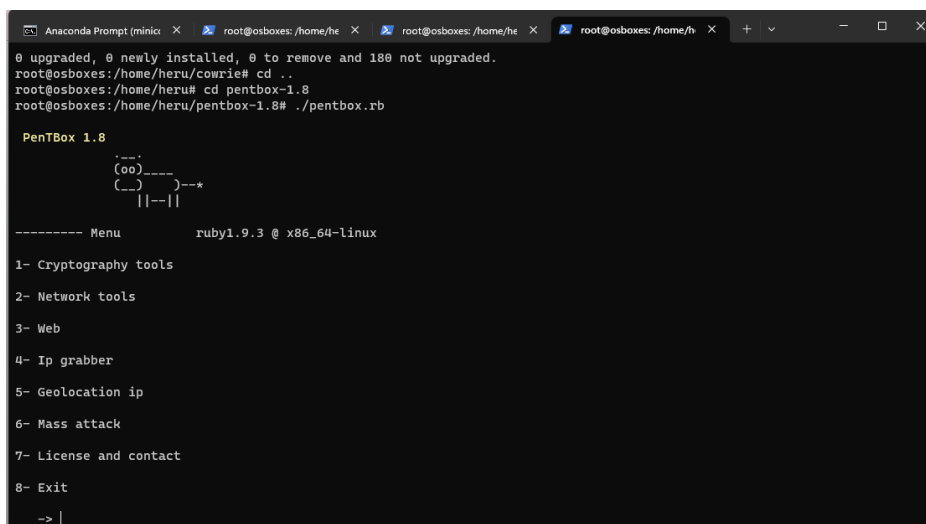
Pertama-tama siapkan dan *install* terlebih dahulu *server ubuntu*, pada perangkat. Setelah memiliki *server*, hal selanjutnya yang perlu dilakukan adalah memastikan bahwa *server* sudah terkoneksi dengan *internet*. Ada banyak jenis *honeypot*, dalam percobaan kali ini, akan menggunakan *pentbox honeypot*. Syarat dalam menggunakan *pentbox* yaitu sudah memiliki paket *ruby*. Cara untuk mendapatkannya dengan melalui “*apt install ruby*” dan akan mendapatkan respon seperti pada Gambar 14.

```
root@osboxes:/home/heru/cowrie# apt install ruby
Reading package lists... Done
Building dependency tree
Reading state information... Done
ruby is already the newest version.
0 upgraded, 0 newly installed, 0 to remove and 180 not upgraded.
root@osboxes:/home/heru/cowrie#
```

Gambar 14. Tanda Bahwa *Ruby* Sudah Terpasang.

Selanjutnya unduh *pentbox*, dengan cara “*wget* <http://downloads.sourceforge.net/project/pentbox18realised/pentbox-1.8.tar.gz>”.

Setelah itu, ekstrak menggunakan “*tar -zxvf pentbox-1.8.tar.gz*”. Lalu, masuk ke *directory* yang telah diekstrak, menggunakan “*cd*”. Langkah selanjutnya adalah jalankan *pentbox* tersebut menggunakan “*./pentbox.rb*” dengan respon yang diterima seperti pada Gambar 15.



```

0 upgraded, 0 newly installed, 0 to remove and 180 not upgraded.
root@osboxes:/home/heru/cowrie# cd ..
root@osboxes:/home/heru# cd pentbox-1.8
root@osboxes:/home/heru/pentbox-1.8# ./pentbox.rb

PenTBox 1.8
  ____
 (oo)____
 (oo)____)---*
  ||--||

----- Menu      ruby1.9.3 @ x86_64-linux

1- Cryptography tools
2- Network tools
3- Web
4- Ip grabber
5- Geolocation ip
6- Mass attack
7- License and contact
8- Exit
-> |

```

Gambar 15. Tampilan Awal Menjalankan *Pentbox*

Dengan demikian, masukan angka 2 untuk menggunakan *network tools*, lalu pilih *honeypot*. Setelah *honeypot* dipilih, maka sebaiknya memilih angka 2, agar konfigurasi lebih bebas. Lalu masukan *port* yang akan dijadikan sebagai servis pancingan, misal 80 terlihat pada Gambar 16, lalu berikan respons yang diinginkan untuk menipu penyerang. Setelah itu, jawab pertanyaan selanjutnya sesuai dengan kebutuhan seperti terlihat pada Gambar 17 dan Gambar 18 merupakan tanda bahwa *honeypot* sudah aktif.


```

Anaconda Prompt (minicr) X root@osboxes: /home/ht X root@osboxes: /home/ht X root@osboxes: /home/ht X + - - - - -
0- Exit
  -> 2
1- Net DoS Tester
2- TCP port scanner
3- Honeypot
4- Fuzzer
5- DNS and host gathering
6- MAC address geolocation (samy.pl)
0- Back
  -> 3
// Honeypot //
You must run PentBox with root privileges.
Select option.
1- Fast Auto Configuration
2- Manual Configuration [Advanced Users, more options]
  -> 2
Insert port to Open.
  -> 80

```

Gambar 16. Langkah untuk Mengaktifkan *Honeypot* dan Memilih *Port*

```

Anaconda Prompt (minicr) X root@osboxes: /home/ht X root@osboxes: /home/ht X root@osboxes: /home/ht X + - - - - -
Insert port to Open.
  -> 80
Insert false message to show.
  -> Welcome to Ubuntu 14.04.5 LTS (GNU/Linux 4.4.0-31-generic x86_64)
Save a log with intrusions?
(y/n)  -> y
Log file name? (incremental)
Default: */pentbox/other/log_honeypot.txt
  -> test80
Activate beep() sound when intrusion?
(y/n)  -> y
HONEYPOT ACTIVATED ON PORT 80 (2023-06-18 03:05:35 -0400)
-----
INTRUSION ATTEMPT DETECTED! from 192.168.56.1:46852 (2023-06-18 03:06:34 -0400)
-----
GET / HTTP/1.1
Host: 192.168.56.103

```

Gambar 17. Contoh *Pentbox* sudah Berjalan pada Port 80

```

Anaconda Prompt (minicr) X root@osboxes: /home/ht X root@osboxes: /home/ht X root@osboxes: /home/ht X + - - - - -
  -> ftp
Activate beep() sound when intrusion?
(y/n)  ->
HONEYPOT ACTIVATED ON PORT 21 (2023-06-18 03:15:18 -0400)
-----
INTRUSION ATTEMPT DETECTED! from 192.168.56.1:48599 (2023-06-18 03:16:47 -0400)
-----
INTRUSION ATTEMPT DETECTED! from 192.168.56.1:48647 (2023-06-18 03:17:07 -0400)
-----
INTRUSION ATTEMPT DETECTED! from 192.168.56.1:48772 (2023-06-18 03:17:54 -0400)
-----
INTRUSION ATTEMPT DETECTED! from 192.168.56.1:48823 (2023-06-18 03:18:18 -0400)
-----
INTRUSION ATTEMPT DETECTED! from 192.168.56.1:48875 (2023-06-18 03:18:42 -0400)
-----

```

Gambar 18. Contoh *Pentbox* sudah Berjalan pada Port 21

Heru Purnama, 2023

**ANALISIS METODE PENGAMANAN PADA PENGGUNAAN
FIREWALL MIKROTIK UNTUK KEAMANAN JARINGAN KOMPUTER**

Universitas Pendidikan Indonesia | repository.upi.edu | perpustakaan.upi.edu

3.6. Metode Penetrasi

3.6.1. Metode Penyerangan *Flood Attack*

Untuk melakukan pengujian terhadap penyerangan dengan algoritma membanjiri (*Flood attack*) salah satu caranya menggunakan modul *socket* pada *python* atau menggunakan *Hping3*. Dengan menggunakan modul *socket* ini atau *Hping3*, memungkinkan untuk mengirim paket yang banyak ke tujuan.

Pertama-tama, hubungkan *library socket* menggunakan *import socket* seperti pada baris ke-1. Langkah selanjutnya adalah menyiapkan *variable* kosong untuk menampung *ip* tujuan menggunakan *list* “*nama_var = []*” pada baris ke-4. Dikarenakan pada percobaan kali ini dilakukan tes terhadap beberapa *port*, maka ada baiknya menggunakan *library random* seperti pada baris ke-5. Untuk menghubungkan *library random* gunakan *import random*. Setelah itu agar tertata lebih rapi dan dapat dipanggil, maka program dimuat pada suatu fungsi seperti pada baris ke-9. Selanjutnya, agar proses *Flood* menjadi lebih kuat, dapat memanfaatkan fungsi *threading*, menggunakan *library thread* dengan *import threading* pada baris ke-3. *Thread* tersebut akan menjalankan fungsi *serang()*, proses *threading* ini berada pada baris ke-20.

Karena percobaan kali ini menggunakan *list*, maka dibutuhkan proses perulangan untuk mengakses data yang terdapat pada *list* tersebut. Dalam percobaan ini digunakan perulangan *for* seperti pada baris ke-15. Perulangan tersebut akan mengulang seberapa banyak data yang ada, misal pada percobaan ini memiliki 2 data, maka akan diulang sebanyak 2 data untuk menjalankan perintah pada baris ke-16 untuk memulai proses pengiriman data menggunakan *socket.sendto* 192.168.1.18 dengan *port* 5000-5221, lalu 192.168.1.1 dengan rentang *port* yang sama.

```

1. import socket
2. import random
3. import threading
4. ip = ['192.168.1.18', '192.168.1.1']
5. port = random.randint(5000, 5221)
6. jp = 65500
7. thread = 5
8. #jp = jumlah paket
9. def serang(ip, port, jp, thread):
10.     data = bytes(1024)
11.     while True:
12.         try:
13.             s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
14.             for x in range(jp):

```

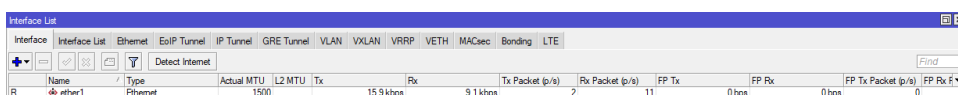
```

15.         for ulang in ip:
16.             s.sendto(data,(str(ulang),int(port)))
17.     except:
18.         Print('gagal');
19. for i in range(thread):
20.     th = threading.Thread(target=serang(ip,port,jp,i))
21.     th.start()

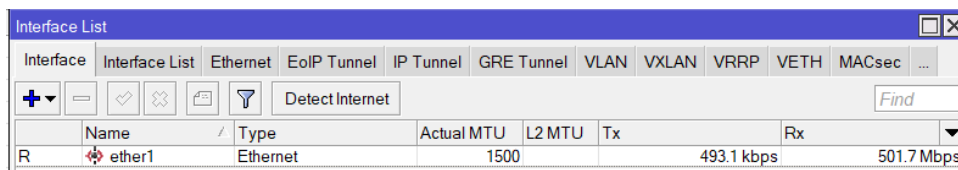
```

Gambar 19. Perintah untuk *Flood Attack*

Dampak dari percobaan penyerangan ini, menyebabkan reaksi spontan terhadap perangkat jaringan komputer. Suatu *interface* yang diserang mengalami lonjakan data yang signifikan secara tiba-tiba. Jika dalam kondisi diam yang normal, *interface* pada Mikrotik *Rx*-nya hanya berkisar pada *kbps* seperti terlihat pada Gambar 20. Akan tetapi, jika dalam kondisi diserang, *Rx*-nya akan berkisar pada besaran *500 Mbps* terlihat pada Gambar 21.



Name	Type	Actual MTU	L2 MTU	Tx	Rx	Tx Packet (p/s)	Rx Packet (p/s)	FP Tx	FP Rx	FP Tx Packet (p/s)	FP Rx Packet (p/s)
R ether1	Ethernet	1500			15.9 kbps	9.1 kbps	2	11	0 bps	0	0

Gambar 20. Kondisi Normal *Interface RX* Mikrotik


Name	Type	Actual MTU	L2 MTU	Tx	Rx
R ether1	Ethernet	1500			501.7 Mbps

Gambar 21. Kondisi Ketika *Interface RX* Mikrotik Diserang

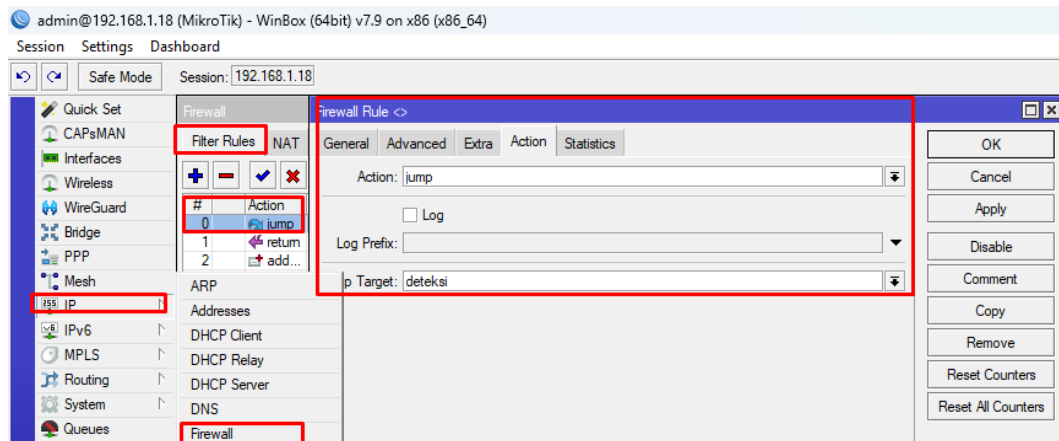
3.7. Metode Pengamanan

3.7.1. Deteksi Serangan Datang

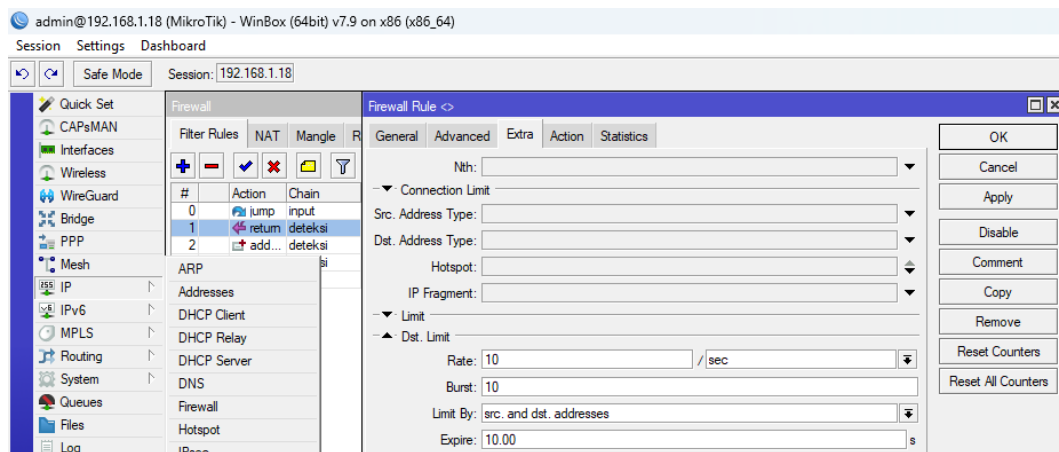
Sebelum memutuskan untuk menolak permintaan yang masuk, ada hal yang perlu dilakukan terlebih dahulu, yaitu mendeteksi serangan yang masuk. Pendeteksian ini dilakukan melalui sedemikian cara dengan memanfaatkan *ip* → *firewall* → *Filter Rule*, dengan menyaring *Destination Limit*, dengan *rate* dan *burst* yang telah ditentukan. Saat *traffic* melebihi *rate* dan *burst* yang telah ditentukan, lalu data *ip* tersebut disimpan ke dalam *list*.

Pertama-tama buat *jump action* menuju algoritma deteksi pada Gambar 22, agar setiap *Filter rules* ini dicek maka akan langsung meloncat menuju algoritma yang dibuat. Setelah *jump* dibuat pada Gambar 22, maka dapatkan data informasi *ip* sumber dan tujuan menggunakan *Dst Limit*, dengan *action* return pada Gambar 23, untuk mengembalikan umpan balik. Lalu daftarkan informasi penyerang menggunakan *add to src address list* pada Gambar 24. Akan tetapi jika ingin

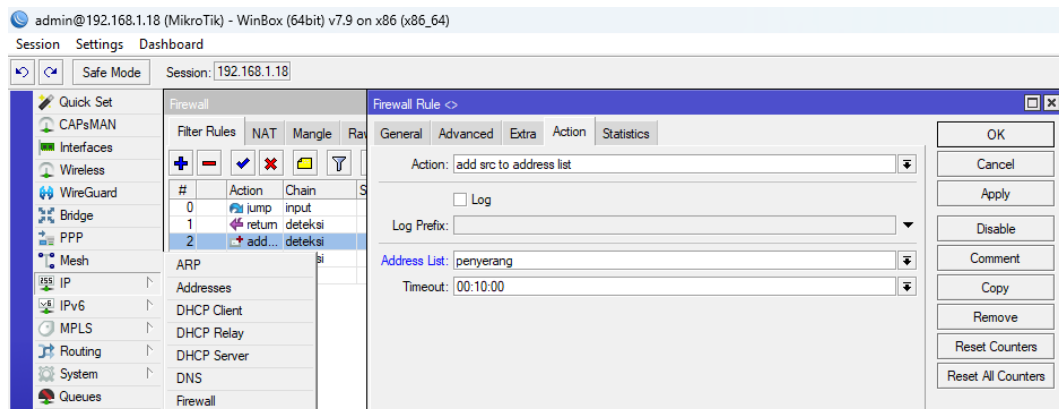
menolak jaringan secara khusus dari sumber ke tujuan (tidak diblokir secara global) dilakukan dengan cara mendaftarkan informasi *ip* tujuan menggunakan *add to Dst address list* pada Gambar 25. Lalu *list* tersebut dapat dilihat pada *tab Address Lists* pada Gambar 26.

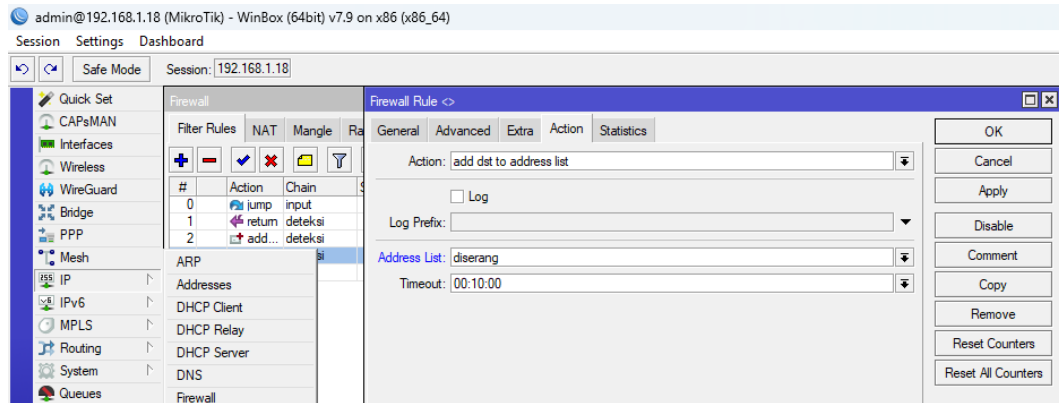
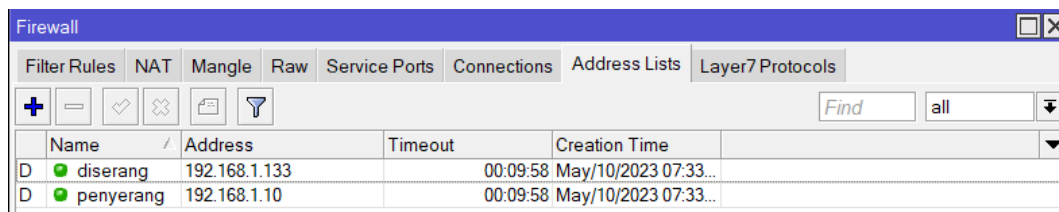


Gambar 22. Pendeteksian menggunakan *Jump Action*



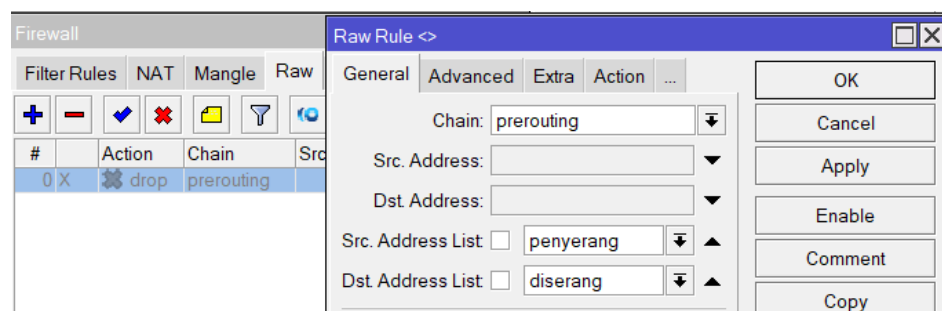
Gambar 23. Pengoleksian Data menggunakan *Firewall Return*



Gambar 24. Penyimpanan Data Penyerang ke *Src Address List*Gambar 25. Penyimpanan Data Tujuan yang Diserang ke *Dst Address List*Gambar 26. Data yang Telah Disimpan ke *Address Lists*

3.7.2. Pengujian Metode Pengamanan

Setelah mendapatkan *list* tujuan dan penyerang, langkah selanjutnya adalah menolak jaringan tersebut, yaitu menggunakan metode *Drop*, *Reject*, *Tarpit*, dan *redirect*. Lalu pada percobaan menghentikan serangan menggunakan *Drop*, dapat memanfaatkan *Filter Rule* atau *raw*. Pada percobaan kali ini akan menggunakan *raw*. Untuk menambahkan *raw*, tekan *ip* → *firewall* → *raw*, lalu tambahkan *src address list* sebagai penyerang, dan *Dst address list* sebagai yang diserang. Apabila ingin melakukan *Drop* secara global, maka tidak perlu menambahkan *Dst address list*, lalu setelahnya beri *action Drop*.

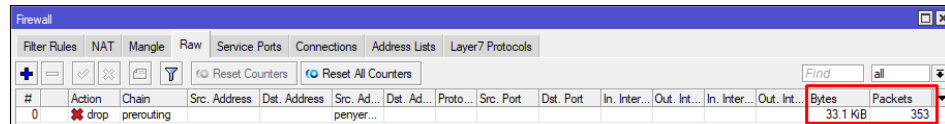
Gambar 27. Pengaplikasian Metode Menggunakan *Raw Drop*

Heru Purnama, 2023

**ANALISIS METODE PENGAMANAN PADA PENGGUNAAN
FIREWALL MIKROTIK UNTUK KEAMANAN JARINGAN KOMPUTER**

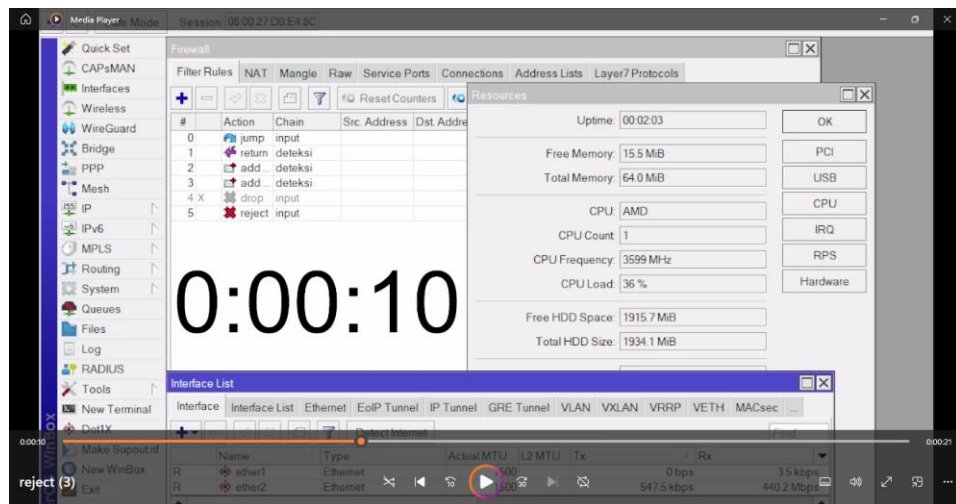
Universitas Pendidikan Indonesia | repository.upi.edu | perpustakaan.upi.edu

Terlihat pada Gambar 28 bahwa fungsi ini sudah berjalan yang ditandai dengan peningkatan nilai *bytes* dan *packets*. Tes ini dilakukan selama 30 detik, perekaman data diambil pada detik ke 10, koleksi data diambil setiap rentang 5 detik, sehingga diperoleh 5 data. Tes diulang sebanyak 3 kali, kemudian didapatkan 15 data untuk setiap metode. Pengujian ini dapat dilihat pada diagram alir Gambar 32.

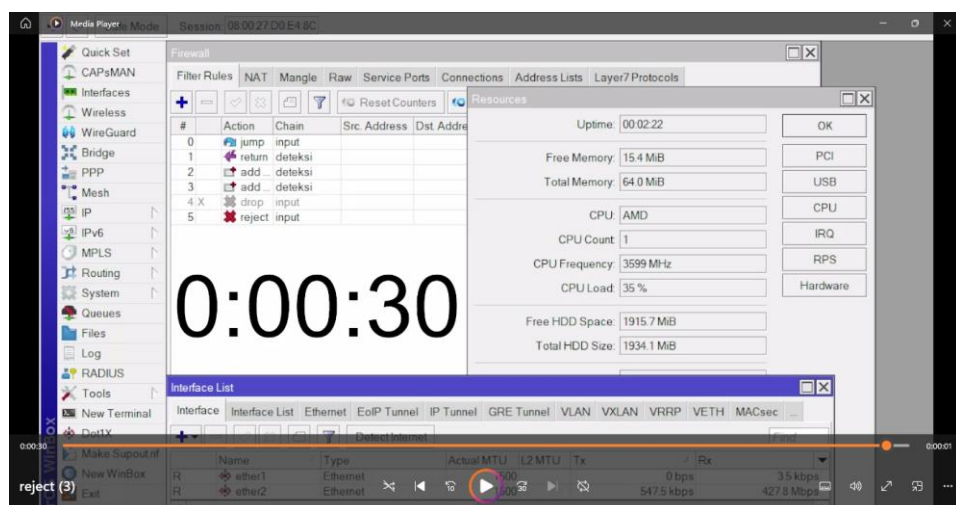


#	Action	Chain	Src. Address	Dst. Address	Src. Ad...	Dst. Ad...	Proto...	Src. Port	Dst. Port	In. Inter...	Out. Int...	In. Inter...	Out. Int...	Bytes	Packets
0	drop	prerouting			penyer...									33.1 KB	353

Gambar 28. Indikator yang Menandakan Fungsi Sudah Berjalan

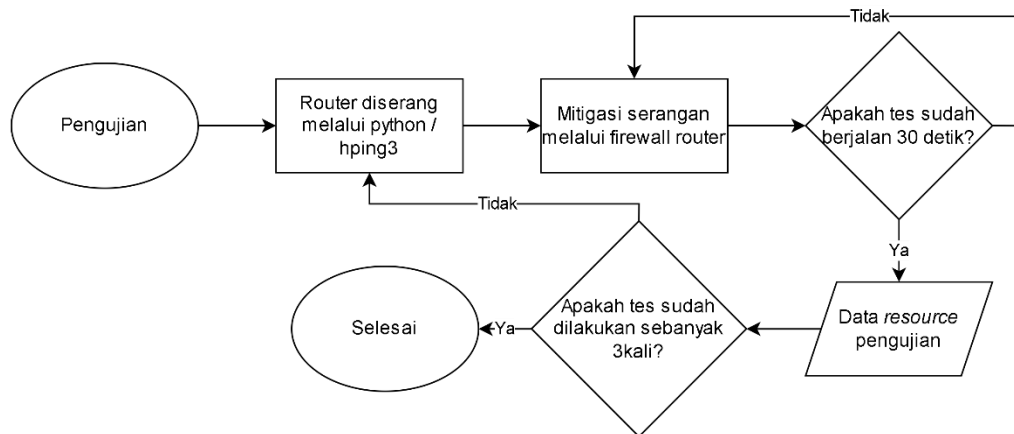


#	Action	Chain	Src. Address	Dst. Address	Src. Ad...	Dst. Ad...	Proto...	Src. Port	Dst. Port	In. Inter...	Out. Int...	In. Inter...	Out. Int...	Bytes	Packets
0	drop	prerouting			penyer...									33.1 KB	353



#	Action	Chain	Src. Address	Dst. Address	Src. Ad...	Dst. Ad...	Proto...	Src. Port	Dst. Port	In. Inter...	Out. Int...	In. Inter...	Out. Int...	Bytes	Packets
0	drop	prerouting			penyer...									33.1 KB	353

Gambar 29. Proses Perekaman Data



Gambar 30. Diagram Alir Proses Perekaman Data

3.8. Metode Pembobotan *Entropy*

Pembobotan *Entropy* ini dibutuhkan untuk memberikan bobot ketika menggunakan *TOPSIS*. Langkah pertama dari pembobotan ini adalah menambahkan semua data yang terlibat seperti yang disajikan pada Tabel 1. Secara matematis langkah ini telah ditulis pada persamaan (11).

Tabel 1. Langkah Awal Pembobotan *Entropy*

Aksi	RAM (Random Access Memory)	CPU Load (Central Processing Unit Load)	RX (Receiver)
Drop	14.6	0.420666667	559.1333333
Reject	15.50666667	0.396	533.3
RD	12.40666667	0.143333333	467.6333333
SUM	42.51333333	0.96	1560.066667

Setelah ditambahkan, simpan data tersebut ke variabel kosong, lalu gunakan untuk membagi variabel *RAM*, *CPU Load*, dan *RX* dengan *Sum*, hasil pembagian tersebut disajikan pada Tabel 2. Secara matematis langkah ini telah ditulis pada persamaan (11).

Tabel 2. Langkah Kedua Pembobotan *Entropy*

Aksi	RAM	CPU Load	RX
Drop	0.343421672	0.438194444	0.358403487
Reject	0.364748314	0.4125	0.341844366
RD	0.291830014	0.149305556	0.299752147

Selanjutnya adalah mengalikan variabel tersebut dengan LN, jadi $i[i,j] \cdot \ln(i[i,j])$. Secara matematis langkah mengalikan *Sum* dengan LN telah ditulis

pada persamaan (12). Setelah itu jumlahkan hasil persamaan (12), nilai ini disimpan pada variabel $LN\ SUM$. Kemudian, kalikan $entropy$ yang telah didapatkan pada persamaan (13) yaitu ($e = -0.9102$) dengan $LN\ SUM$. Hasil dari perkalian ini digunakan untuk mengurangi angka 1, dan disimpan pada variabel $1-entropy$. Lalu jumlahkan dan simpan pada variabel kosong, misalnya $Sum\ entropy$. Terakhir, $Sum\ entropy$ digunakan untuk membagi $1-entropy$, dan simpan ke variabel bobot seperti yang disajikan pada Tabel 3. Variabel bobot ini, jika ditulis secara matematis terlihat pada persamaan (14).

Tabel 3. Langkah Ketiga Pembobotan $Entropy$

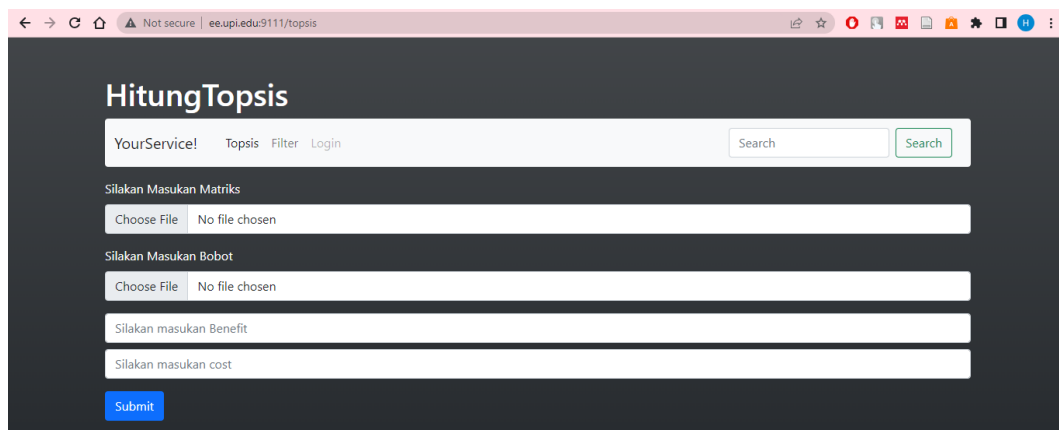
Aksi	RAM	CPU Load	RX	KETERANGAN
<i>Drop</i>	-0.367047786	-0.361550963	-0.367756337	$Drop[index]*LN(Drop[index])$ (13)
<i>Reject</i>	-0.367866078	-0.365276618	-0.366935645	$Reject[index]*LN(Reject[index])$ (13)
<i>RD</i>	-0.359413115	-0.283943388	-0.361141184	$RD[index]*LN(RD[index])$ (13)
$LN\ SUM$	-1.094326979	-1.010770968	-1.095833166	$SUM()$ (13)
$Entropy$	0.996099343	0.920043384	0.997470334	$LN\ SUM * e$ (13)
$1-entropy$	0.003901	0.079957	0.00253	$SUM\ ENTROPY = 0.086386939$ (14)
$Weight$	0.045153322	0.925563707	0.029282971	$(1-ENTROPY) / (SUM\ ENTROPY)$ (14)

3.9. Otomasi Pembobotan $Entropy$ dan $TOPSIS$ Menggunakan $Python$

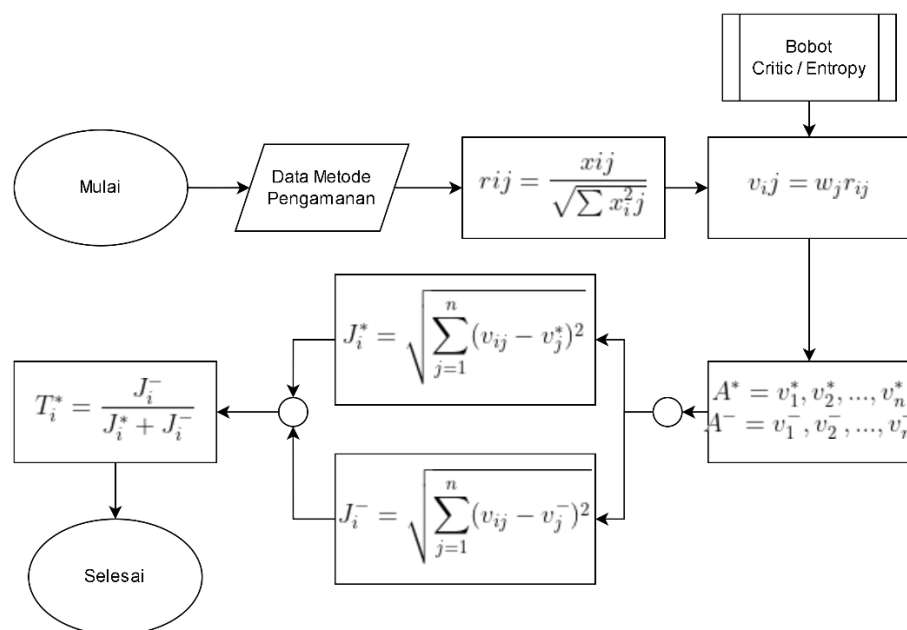
Pada penelitian ini, dibuat suatu antarmuka yang membantu input data berupa *file* dan *index* agar penggunaan $TOPSIS$ lebih mudah. Hal ini juga diharapkan dapat mempermudah orang lain, contohnya adalah salah satu rekan, sedang meneliti *multi criteria* menggunakan $HOMER$, yang membutuhkan $TOPSIS$, maka dapat menggunakan *website* yang telah dibuat seperti pada Gambar 31.

Sistem ini menggunakan $Flask$ sebagai antarmuka dan memanfaatkan $Numpy$, serta perulangan pada $python$ untuk mengakses *index* data. Algoritma utamanya terdapat pada fungsi $TOPSIS()$, lalu mengunggah *file* menggunakan *request.files* melalui $POST$ method. Setelah *file* diminta, selanjutnya adalah melakukan ekstraksi terhadap isi konten menggunakan $Numpy.loadtxt$. Tahap selanjutnya adalah meminta *index benefit* dan *cost*, menggunakan *variable* $n1$ dan $n2$.

Setelah mendapatkan *benefit* dan *cost*, selanjutnya digunakan fungsi $\text{cumsqrt}()$ untuk menjumlahkan akar dari *Sum* kumulatif x^2 (x adalah data yang di input). Data tersebut digunakan sebagai pembagi untuk x , jadi secara matematis dapat ditulis $\frac{x}{\sqrt{x}}$. Data pembagian tersebut dikalikan dengan bobot yang terdapat pada fungsi $\text{kalibobot}()$. Setelah itu, ditentukan solusi ideal positif dan negatif menggunakan fungsi $\text{fsip}(\text{benefit}, \text{cost})$, dengan n_1 dan n_2 yang disubstitusikan ke dalam fungsi tersebut. Terakhir, adalah menentukan d_{plus} dan d_{min} untuk mendapatkan nilai koefisien *index* terdekat menuju 1. Secara keseluruhan metode TOPSIS dapat dilihat pada Gambar 32.



Gambar 31. Tampilan Website TOPSIS yang dibuat menggunakan Flask Python



Gambar 32. Diagram Alir Metode Topsis