**Journal of Applied Multimedia and Networking (JAMN)**
Vol.x, No.x, Maret 201x, pp. xx~xx
e-ISSN: 2548-6853 

# Implementasi Open Policy Agent sebagai Solusi pengaturan Akses Dinamis untuk Meningkatkan Keamanan jaringan dengan Pendekatan Terpusat

**Khivan Bintang Khalillah *, Antoni Haikal ***
* Informatics Engineering, Batam State Polytechnic

**Article Info**

***Article history:***

Received Mar 14th, 2024
Revised Nov 23th, 2024
Accepted Nov 28th, 2024

***Keyword:***

Open Policy Agent
Jaringan
iptables

**ABSTRACT** (10 PT)

*The implementation of Open Policy Agent (OPA) as a dynamic access management solution aims to improve network security in decentralized multi-server systems. The system utilizes the Policy as Code (PaC) concept so that admins can manage access policies in a more structured and consistent manner from a central server. With centralized settings, the system monitors activities and analyzes logs from each branch server to automatically detect suspicious activities. Tests show that the system successfully identifies and prevents unauthorized access attempts in real-time, and accelerates responses to potential security threats through responsive automated policies. This implementation demonstrates the importance of adaptive security in maintaining stability and access integrity on distributed networks.*



***Corresponding Author: Khivan Bintang Khalillah***

## 1. PENDAHULUAN

Dalam era digital, keamanan jaringan menjadi salah satu aspek penting bagi organisasi untuk menjaga integritas dan kerahasiaan data. Berdasarkan laporan Badan Sandi Negara (BSSN) tahun 2019, Indonesia mencatat 290,3 juta serangan siber, termasuk 117,9 juta serangan trojan, 137,4 juta pembocoran data, 12,5 juta serangan pada port 80, dan ribuan aduan publik terkait kerentanan sistem. Angka ini mencerminkan tingginya ancaman terhadap infrastruktur jaringan.Pengelolaan jaringan dalam lingkungan multi-server semakin kompleks, terutama dalam memastikan pengaturan akses yang aman dan efisien [1]. IPTables, sebagai firewall bawaan Linux, banyak digunakan untuk mengendalikan akses jaringan. Namun, pengelolaan kebijakan secara manual seringkali rumit dan rawan kesalahan, terutama ketika aturan yang diterapkan cukup banyak. Kesalahan konfigurasi dapat menciptakan celah keamanan atau menghambat akses yang sah [2].Teknologi Policy as Code (PaC) menawarkan solusi melalui pengelolaan kebijakan dalam bentuk kode yang dapat diterapkan secara otomatis dan terpusat [3]. Salah satu alat untuk implementasi PaC adalah Open Policy Agent (OPA), yang memungkinkan pembuatan, pengelolaan, dan penerapan kebijakan akses jaringan secara dinamis. Dengan OPA, sistem dapat mendeteksi ancaman secara real-time dan menegakkan kebijakan secara otomatis, meningkatkan respons keamanan sekaligus meminimalkan risiko kesalahan manual [4].

Penelitian sebelumnya telah mengeksplorasi berbagai teknik untuk meningkatkan keamanan jaringan, khususnya dalam mengurangi risiko akses tidak sah. Contohnya, Mardiansyah (2021) dan Amien (2020) memanfaatkan kombinasi antara iptables dengan teknik port knocking dan honeypot untuk memperketat akses ke server. Teknik ini memungkinkan sistem untuk membuka port hanya untuk pengguna yang telah melewati autentikasi tertentu, sehingga membatasi akses dari pihak yang tidak diinginkan. Studi lainnya, seperti yang dilakukan oleh Hawari (2016) dan Sularno (2016), lebih berfokus pada penerapan iptables pada Linux sebagai firewall untuk memfilter akses jaringan. Teknik ini digunakan untuk mencegah akses ilegal ke sumber daya jaringan dan menjaga integritas data. Dalam ranah Policy as Code, Ahmed (2020) memperkenalkan OPA sebagai alat yang memudahkan pengelolaan kebijakan jaringan dalam bentuk kode yang dapat diterapkan dan diatur secara terpusat. Baier (2023) dan Connelly (2023) menyoroti keunggulan OPA dalam menangani aturan kebijakan kompleks dengan menggunakan bahasa Rego, yang dirancang khusus untuk membuat kebijakan yang dapat dievaluasi dan diproses secara efisien. Matharu (2022) juga menunjukkan bahwa PaC sangat cocok untuk jaringan dinamis yang membutuhkan penanganan keamanan berkelanjutan, di mana kebijakan dapat

diperbarui dengan cepat tanpa mengganggu operasional jaringan. Salah satu tutorial yang disediakan oleh komunitas Open Policy Agent (GitHub, 2023) juga menjadi salah satu referensi penting. Tutorial ini mendemonstrasikan cara integrasi OPA dengan iptables untuk mengelola aturan firewall secara dinamis. Dalam implementasi tersebut, OPA memproses kebijakan yang ditulis dalam bahasa Rego, menentukan apakah suatu permintaan jaringan harus diizinkan atau ditolak[5]. Namun, tutorial ini hanya mencakup pengaturan kebijakan statis tanpa melibatkan pemantauan log aktivitas atau deteksi anomali jaringan secara otomatis. Keterbatasan ini menjadi dasar untuk mengembangkan sistem yang lebih komprehensif, di mana OPA tidak hanya memproses kebijakan tetapi juga mendukung pemantauan log secara real-time dan otomatisasi kebijakan melalui integrasi dengan bash script dan cronjob. Penelitian-penelitian ini menjadi landasan penting dalam mengembangkan sistem keamanan jaringan yang lebih responsif, khususnya di lingkungan yang membutuhkan adaptasi cepat terhadap ancaman siber yang terus berkembang.

Penelitian ini bertujuan untuk mengembangkan sistem pengaturan akses dinamis yang menggunakan OPA sebagai evaluator kebijakan terpusat. Dalam sistem ini, OPA di server pusat menerima permintaan evaluasi akses dari server lain di jaringan dan menentukan apakah akses tersebut diperbolehkan atau harus diblokir berdasarkan kebijakan yang telah ditentukan. Setiap server di jaringan akan memantau aktivitasnya sendiri dan melaporkannya ke OPA server pusat untuk dievaluasi, sehingga OPA dapat mengambil keputusan apakah akses harus diizinkan atau ditolak. Berdasarkan kebijakan yang diterapkan, OPA akan memberikan respons yang sesuai terhadap potensi ancaman secara otomatis. Dengan demikian, aplikasi ini diharapkan dapat membantu organisasi dalam mengelola pengaturan akses jaringan secara efisien, aman, dan terpusat. Sistem ini tidak hanya dapat mengidentifikasi dan menanggapi ancaman jaringan secara cepat tetapi juga mengurangi risiko serangan terhadap data dan layanan yang dilindungi. Solusi ini dirancang untuk mengoptimalkan keamanan jaringan secara adaptif, memungkinkan pengaturan akses yang fleksibel dan responsif dalam menghadapi berbagai ancaman siber yang semakin kompleks.

## 2. TINJAUAN PUSTAKA

### 2.1 IPTABLES

Iptables merupakan alat firewall berbasis command-line pada Linux yang digunakan untuk mengelola dan menyaring lalu lintas jaringan dengan menerapkan aturan keamanan berdasarkan alamat IP, protokol, port, atau kondisi tertentu, yang pada proyek ini digunakan untuk memblokir akses mencurigakan berdasarkan evaluasi kebijakan dari Open Policy Agent (OPA). Iptables merupakan sistem firewall yang mendukung layer3 (Network layer), layer4 (Transport layer) dan layer7 OSI layer.

### 2.2 BASH SCRIPT

Bash script merupakan kumpulan perintah dalam file teks yang dijalankan menggunakan Bash (Bourne Again SHell), sebuah interpreter command-line di sistem operasi Linux atau Unix. Bash script digunakan untuk mengotomatisasi tugas-tugas rutin di sistem, seperti pengelolaan file, manajemen proses, dan jaringan. Shell adalah program yang menyediakan antarmuka baris perintah untuk berinteraksi dengan sistem operasi. Dengan Bash script, berbagai tugas yang biasanya dilakukan secara manual melalui antarmuka, seperti navigasi direktori, pembuatan file, atau eksekusi program, dapat diotomatisasi. Setiap Bash script dimulai dengan shebang (#!/bin/bash), yang menunjukkan lokasi interpreter Bash dan memberi tahu sistem untuk menggunakan Bash dalam menjalankan skrip tersebut.

### 2.3 CRONJOB

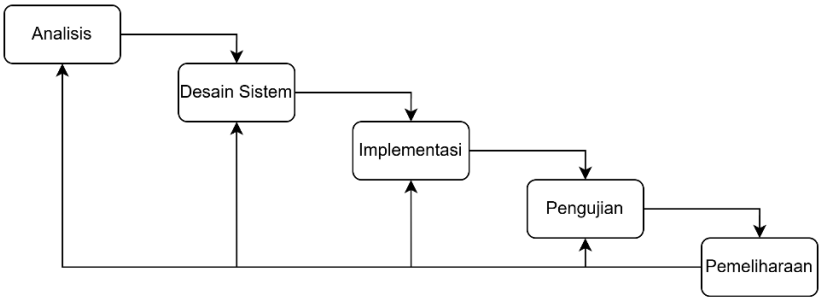
Cronjob adalah alat penjadwalan pada Linux yang menjalankan skrip atau tugas secara terjadwal dan otomatis. Terjadwal yang dimaksud adalah pekerjaan itu akan berlangsung sesuai dengan waktu yang kita tentukan. CronJob bekerja dengan menggunakan daemon yang disebut cron. Cron membaca file konfigurasi yang disebut “crontab” (singkatan dari “cron table”) yang berisi daftar tugas dan waktu pelaksanaan[6].

### 2.4 OPEN POLICY AGENT

Open Policy Agent (OPA) adalah alat open-source yang digunakan untuk mengelola kebijakan dan membuat keputusan secara terpusat dalam sistem terdistribusi. Diperkenalkan pertama kali pada tahun 2016 oleh Styra, Inc., OPA dirancang untuk membantu organisasi dalam mengelola kebijakan akses, validasi konfigurasi, dan pengambilan keputusan lainnya secara konsisten di berbagai layanan atau infrastruktur[7]. OPA bekerja menggunakan kebijakan yang ditulis dalam bahasa deklaratif bernama Rego dan OPA hanya menerima permintaan dan mengirim respons dalam format json. OPA melakukan pemisahan logika agar aturan atau kebijakan tidak tercampur dengan kode utama aplikasi.

## 3. METODOLOGI PENGEMBANGAN

Penelitian ini dilakukan dengan menggunakan metode waterfall, yaitu pendekatan sistematis dan berurutan dalam pengembangan sistem. Model ini dimulai dari tahap spesifikasi kebutuhan, diikuti oleh perencanaan, implementasi, pengujian, hingga pemeliharaan [8]. Metode waterfall pertama kali diperkenalkan oleh Herbert D. Benington pada tahun 1956 dan dipopulerkan lebih rinci oleh Winston Royce pada tahun 1970. Meskipun sering dianggap kuno, model ini tetap menjadi salah satu yang paling banyak digunakan karena pendekatannya yang terstruktur. Metode ini disebut waterfall karena setiap langkah pengembangan harus diselesaikan terlebih dahulu sebelum melanjutkan ke langkah berikutnya, seperti aliran air terjun. Dalam penelitian ini, metode waterfall digunakan untuk mengintegrasikan kebijakan keamanan dinamis pada iptables dengan evaluasi menggunakan Open Policy Agent (OPA) secara bertahap [9].




Gambar 1. Metode Waterfall

Pada tahap analisis kebutuhan, kita mengidentifikasi kebutuhan keamanan yang diperlukan untuk mengelola akses jaringan di lingkungan server terpusat. Analisis ini melibatkan pemahaman terhadap skenario yang dibuat, seperti deteksi percobaan akses yang berlebihan dan pemblokiran otomatis. Hasil dari tahap ini adalah panduan atau spesifikasi kebutuhan sistem yang jelas.

Berdasarkan hasil analisis, tahapan desain sistem membuat rancangan lengkap sistem yang akan dibangun. Ini mencakup pengaturan arsitektur server, pola komunikasi antara server cabang dan pusat, serta rancangan kebijakan keamanan yang dibuat dengan Rego. Diagram dan dokumentasi alur kerja juga disusun sebagai panduan pengembangan.

Pada tahap implementasi, sistem mulai dikembangkan sesuai desain. Ini mencakup penerapan Open Policy Agent (OPA) di server pusat dan lokal, penyiapan pemantauan log, dan otomatisasi pemblokiran di server cabang dengan iptables. Skrip bash dibuat agar proses berjalan otomatis, mulai dari pemantauan hingga pengiriman data ke server pusat.

Di tahap pengujian, seluruh fungsi sistem diuji untuk memastikan semuanya berjalan sesuai rencana. Pengujian meliputi evaluasi kebijakan Rego dan komunikasi antar-server, serta memastikan pemblokiran bekerja efektif ketika ada aktivitas mencurigakan, seperti serangan SSH dan HTTP. Jika ada masalah, perbaikan segera dilakukan.

Setelah sistem aktif, pemeliharaan dilakukan untuk menjaga performanya tetap optimal. Ini meliputi pemantauan kinerja, pembaruan kebijakan atau skrip bila diperlukan, dan respons terhadap potensi ancaman keamanan baru.

## 4. HASIL DAN PEMBAHASAN

### 4.1 Analisis Kebutuhan Jaringan

Analisis kebutuhan jaringan merupakan sebuah proses mengidentifikasi dan menentukan spesifikasi suatu teknis dan fungsional dari jaringan komputer yang dibutuhkan [10]. Dalam sistem ini, jaringan harus mendukung komunikasi yang intensif antara server pusat, server cabang, dan klien. Koneksi yang andal sangat penting untuk mengirimkan permintaan evaluasi kebijakan dari server cabang ke server pusat secara real-time, serta untuk memastikan klien dapat mengakses layanan di server cabang sesuai dengan kebijakan yang berlaku.

**a. Gambaran Umum**

Sistem ini menggunakan Open Policy Agent (OPA) sebagai pusat pengelolaan kebijakan keamanan jaringan yang terpusat dan otomatis. OPA berfungsi sebagai evaluator kebijakan yang menerima permintaan (request) dari server cabang, memprosesnya berdasarkan kebijakan yang telah ditentukan dalam file Rego, dan mengirimkan respons (response) kembali ke server cabang untuk menentukan tindakan selanjutnya. Dengan menggunakan OPA, admin dapat mengelola

semua aturan keamanan dari satu lokasi di server pusat, sehingga memastikan konsistensi kebijakan di seluruh server yang terhubung dengan OPA.

Dalam sistem ini, kebijakan keamanan ditulis dalam bentuk kode menggunakan bahasa deklaratif Rego, yang memungkinkan admin untuk dengan mudah mendefinisikan aturan yang kompleks. Admin bertanggung jawab memuat file Rego ke server OPA melalui endpoint API, yang membuat kebijakan dapat diperbarui tanpa menghentikan layanan OPA. Ketika server cabang mendeteksi aktivitas mencurigakan, seperti percobaan login SSH berulang atau permintaan HTTP yang berlebihan, server cabang akan mengirimkan permintaan ke OPA melalui HTTP API dengan data input, seperti jumlah percobaan login atau alamat IP sumber permintaan. OPA kemudian mengevaluasi data tersebut berdasarkan kebijakan dalam file Rego, memutuskan apakah aktivitas tersebut sesuai dengan aturan yang telah ditentukan, dan mengirimkan respons berupa true (untuk memblokir akses) atau false (untuk mengabaikan). Respons dari OPA ini diterima oleh server cabang, yang kemudian secara otomatis memperbarui aturan pada iptables untuk memblokir IP sumber jika respons adalah true.

Pendekatan ini memastikan bahwa kebijakan keamanan diterapkan secara konsisten dan otomatis di seluruh jaringan, mengurangi risiko kesalahan manual, dan memungkinkan sistem untuk merespons ancaman dengan cepat dan adaptif. Dengan mekanisme ini, pengelolaan keamanan jaringan menjadi lebih efisien, fleksibel, dan responsif terhadap ancaman siber yang terus berkembang.

**b. Spesifikasi Komponen Sistem**

Komponen-komponen sistem yang diperlukan untuk menjalankan OPA dan mendukung seluruh fungsionalitas sistem secara optimal**.**

**1. Spesifikasi Laptop**

- Prosesor : AMD Ryzen 5 3500U @ 2.1 Ghz.
- RAM : 8 GB DDR 4
- Penyimpanan : SSD 256 GB
- OS : Windows 11

**2. Spesifikasi Virtual Server**

Tabel 1. Spesifikasi Virtual Box

| Server | CPU | RAM | Size | Jaringan | OS |
|---|---|---|---|---|---|
| **Pusat** | **2 Core** | **3624 MB** | **25,31 GB** | **NAT** | **Fedora server 40** |
| **Cabang** | **2 Core** | **3000 MB** | **10 GB** | **NAT** | **Fedora server 40** |
| **Client** | **2 Core** | **2500 MB** | **10 GB** | **NAT** | **Fedora server 40** |

**c. Kebutuhan Fungsional dan Non-Fungsional**

Kebutuhan fungsional merupakan jenis kebutuhan yang mencakup proses-proses yang diperlukan agar sistem dapat berjalan sesuai fungsi utamanya [11]. Berikut adalah hasil analisis kebutuhan fungsional dari Implementasi Open Policy Agent sebagai Solusi pengaturan Akses Dinamis untuk Meningkatkan Keamanan jaringan dengan Pendekatan Terpusat.

Tabel 2. Kebutuhan Fungsional

| NO | KEBUTUHAN FUNGSIONAL |
|---|---|
| F01 | Admin dapat menjalankan dan menghentikan OPA di setiap server |
| F02 | Admin dapat membuat, menyunting, dan menghapus kebijakan Rego yang tersimpan di server pusat. |
| F03 | Admin dapat memuat kebijakan Rego ke dalam OPA melalui API. |
| F04 | Admin dapa membuat, menyunting, dan menghapus file bash script yang ada pada server cabang |

| | |
|---|---|
| F05 | Sistem memantau log aktivitas pada port SSH dan HTTP untuk mendeteksi percobaan akses tidak sah secara otomatis setiap menit. |
| F06 | Sistem mampu menerima permintaan evaluasi kebijakan dari server cabang dan merespons sesuai dengan aturan yang berlaku. |
| F07 | Sistem dapat memblokir akses yang dianggap tidak aman menggunakan iptables berdasarkan evaluasi kebijakan OPA di server pusat. |
| F08 | Client melakukan akses layanan di server cabang. |

Kebutuhan non-fungsional adalah kebutuhan yang berhubungan dengan sifat-sifat atau karakteristik yang harus dimiliki sistem agar dapat beroperasi secara optimal [12]. Berikut adalah hasil analisis kebutuhan non-fungsional dari Implementasi Open Policy Agent sebagai Solusi pengaturan Akses Dinamis untuk Meningkatkan Keamanan jaringan dengan Pendekatan Terpusat.

Tabel 3. Kebutuhan Non-Fungsional

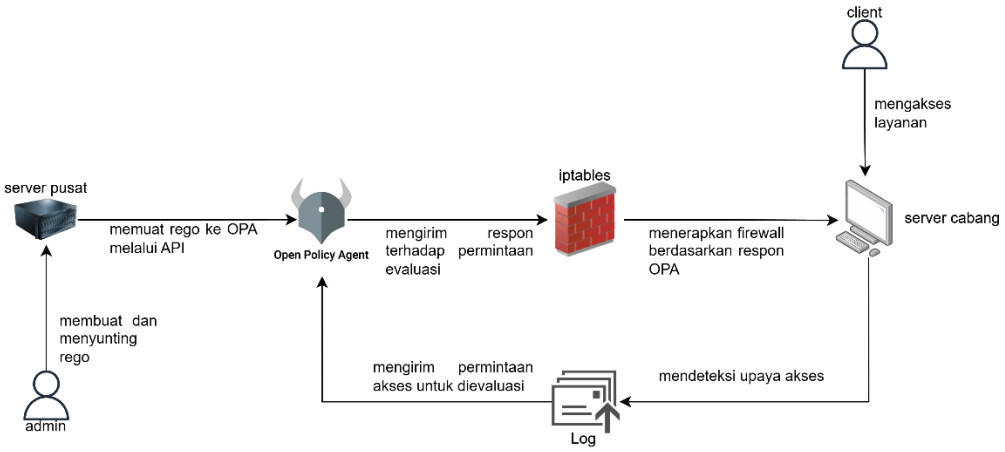
| KRITERIA | PARAMETER |
|---|---|
| Reliability | Sistem harus tersedia 24/7, sehingga setiap permintaan akses dan evaluasi kebijakan dapat diproses tanpa gangguan waktu.sistem |
| Compability | Sistem dapat menangani permintaan dari beberapa server cabang tanpa penurunan performa. |

### 4.2 Rancangan Sistem

Pata tahapan ini, analisis menghasilkan rincian spesifikasi kebutuhan terhadap sistem yang akan dikembangkan. Perancangan menjadikan rincian spesifikasi kebutuhan untuk menghasilkan spesfikasi rancangan sistem yang akan dibangun [13].

**a. Diagram Arsitektur Jaringan**

Diagram arsitektur jaringan merupakan representasi visual yang menunjukkan struktur fisik dan logis dari sistem jaringan komputer yang dirancang. Diagram ini menggambarkan bagaimana berbagai komponen saling terhubung dan berkomunikasi satu sama lain. Diagram ini bertujuan untuk memberikan gambaran menyeluruh mengenai infrastruktur jaringan disusun, termasuk jalur komunikasi dan interaksi antara perangkat atau sistem [14].




Gambar 2. Diagram Arsitektur Jaringan

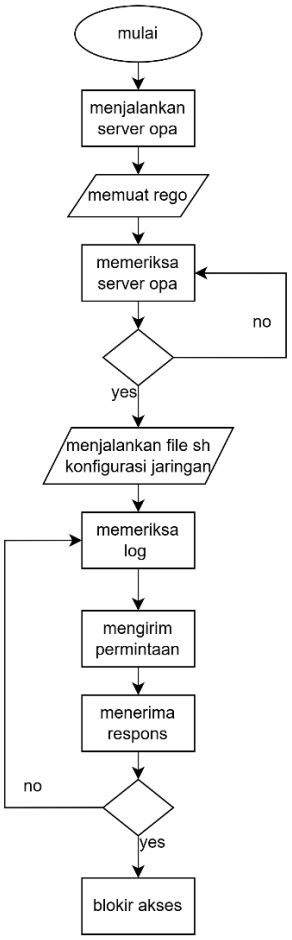
Berikut adalah deskripsi berdasarkan gambar 2:

1. Admin membuat file rego pada server pusat.
2. Admin kemudian memuat file rego tersebut ke dalam API OPA di server pusat agar bisa merespon permintaan evaluasi secara konsisten dari para server cabang.
3. Ketika server cabang diakses oleh client, log mencatat akses pada layanan port http/ssh.
4. Server cabang melakukan permintaan evaluasi ke OPA server pusat untuk menetukan apakah upaya akses tersebut biasa saja atau melebihi upaya yang telah ditetapkan.

5. Kemudian OPA pada server pusat memberikan output respon dari permintaan server cabang berdasarkan jumlah upaya yang terdeteksi dari log yang dilampirkan.
6. Setelah menerima respon, keputusan pemblokiran atau tidaknya akses di server cabang oleh client, ditentukan oleh iptables.

**b. Flowchart**

Flowchart adalah cara penulisan algoritma dengan menggunakan notasi grafis. Flowchart merupakan diagram yang menampilkan langkah-langkah dan keputusan untuk melakukan sebuah proses dari suatu program. Setiap langkah digambarkan dalam bentuk diagram dan dihubungkan dengan garis atau arah panah. Fungsi utama dari flowchart adalah memberi gambaran jalannya sebuah program dari satu proses ke proses lainnya.




Gambar 3. Flowchart

Berdasarkan alur kerja dari gambar flowchart pada gambar 3 diatas, dapat dijelaskan sebagai berikut:

1. Pada server pusat, opa dijalankan oleh admin menggunakan ansible agar server cabang dapat menjalankan opa secara otomatis
2. Setelah server opa berhasil dijalankan di setiap server, kemudian admin memuat file rego ke dalam opa hanya di server pusat sebagai pedoman untuk evaluasi ketika ada permintaan dari server cabang.
3. Setelah dimuat, beralih ke server cabang. Di server cabang terdapat ketika server opa sudah jalan, otomatis file sh konfigurasi jaringan dijalankan untuk memantau log pada port ssh dan http
4. Ketika ada upaya akses yang terjadi pada port ssh atau http, maka dilakukan permintaan evaluasi ke opa server pusat
5. Server pusat langsung memeriksa akses yang diterima dari server cabang dengan membandingkan jumlah akses atau permintaan tersebut dengan file rego yang telah dimuat

6. Server pusat memberikan respons, dan respon yang diterima oleh server cabang terdapat 2 jenis, jika false maka eksekusi oleh iptables diabaikan, jika true maka eksekusi oleh iptables dilakukan, yaitu pemblokiran akses

### 4.3 Implementasi Sistem

**a. Membuat file rego pada server pusat**

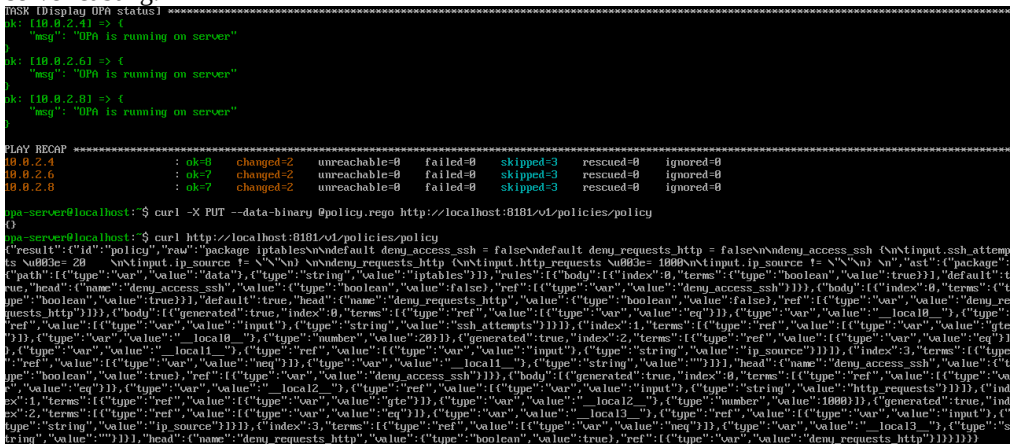
Pada server pusat, admin membuat sebuah file rego yang berfungsi sebagai kebijakan yang nantinya akan menjadi pedoman untuk evaluasi permintaan dari server cabang. Isi file rego ini dapat diubah tergantung kebutuhan pada sistem tanpa harus mematikan server OPA. Sebelum rego dimuat ke dalam OPA melalui API, rego tersebut dapat diperiksa untuk mengetahui apakah ada kesalahan penulisan dalam filenya.

```
opa-server@localhost:~$ cat policy.rego
package iptables

default deny_access_ssh = false
default deny_requests_http = false

deny_access_ssh {
        input.ssh_attempts >= 20
        input.ip_source != ""
}

deny_requests_http {
        input.http_requests >= 1000
        input.ip_source != ""
}
opa-server@localhost:~$ opa check policy.rego
opa-server@localhost:~$ _
```

Gambar 4. Membuat File Rego

**b. Memuat file rego ke dalam OPA melalui API**

setelah tidak ada pesan kesalah pada file rego yang dibuat di server pusat, admin kemudian menjalankan OPA dalam mode server. Lalu memuat file rego ke dalam OPA melalui API dan setelahnya file rego yang dimuat dapat dilihat pada gambar dalam bentuk formt JSON. File rego yang dimuat ke API, digunakan oleh OPA sebagai logika untuk mengevaluasi permintaan dari server cabang.

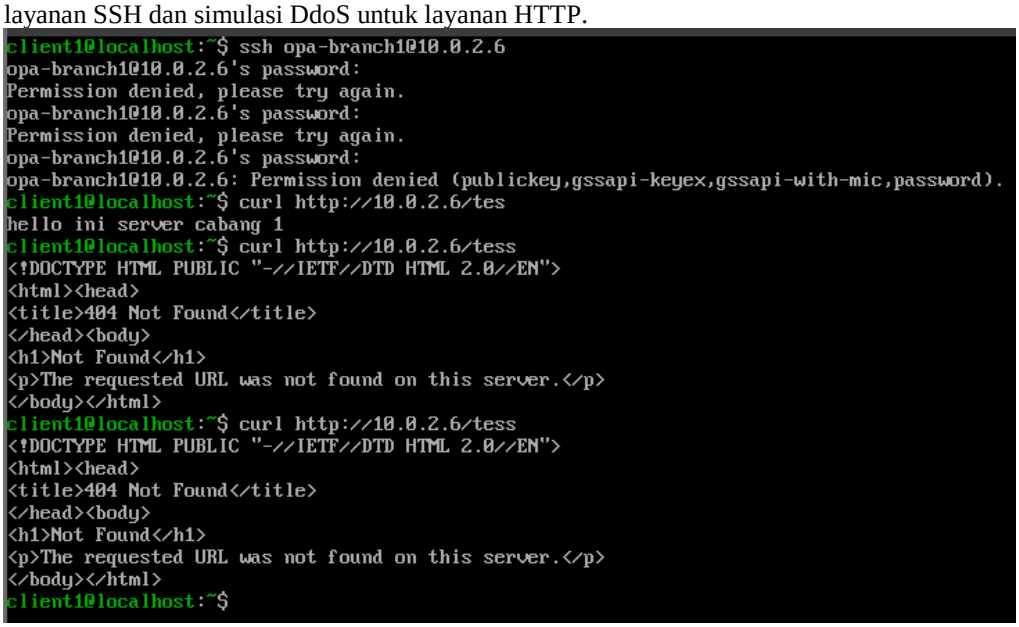



Gambar 5. File Rego dimuat Melalui API

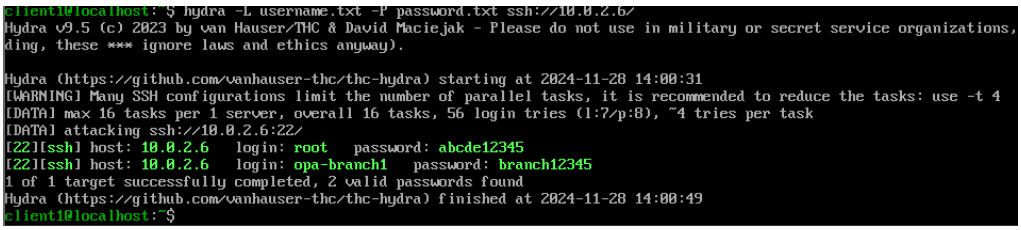
**c. Implementasi akses oleh client ke server cabang**

Setelah file rego dimuat oleh admin di server pusat, kemudian untuk menguji apakah rego mampu menjadi evaluator, server client inilah yang berperan untuk mengakses layanan pada server cabang, layanan yang tersedia ialah layanan port SSH dan HTTP. Pada gambar dibawah ini, dilakukan akses sederhana oleh client dengan melakukan gagal ssh dan permintaan http yang

tidak tersedia sebanyak 3 kali ke server cabang. Kemudian melakukan simulasi brute force untuk layanan SSH dan simulasi DdoS untuk layanan HTTP.




Gambar 6. Akses Sederhana oleh Client




Gambar 7. Simulasi Brute Force dengan Hydra

```
Completed 500 requests
Completed 1000 requests
Completed 1500 requests
Completed 2000 requests
Completed 2500 requests
Completed 3000 requests
Completed 3500 requests
Completed 4000 requests
Completed 4500 requests
Completed 5000 requests
Finished 5000 requests


Server Software:        Apache/2.4.62
Server Hostname:        10.0.2.6
Server Port:            80

Document Path:          /
Document Length:        8474 bytes

Concurrency Level:      1000
Time taken for tests:   55.822 seconds
Complete requests:      5000
Failed requests:        0
Non-2xx responses:      5000
Total transferred:      43760000 bytes
HTML transferred:       42370000 bytes
Requests per second:    89.57 [#/sec] (mean)
Time per request:       11164.416 [ms] (mean)
Time per request:       11.164 [ms] (mean, across all concurrent requests)
Transfer rate:          765.55 [Kbytes/sec] received

Connection Times (ms)
              min  mean[+/-sd] median   max
Connect:        2  527 760.6     31    3118
Processing:   931 9312 7676.2   5711   53180
Waiting:        4 7360 6816.4   5377   53028
Total:       2595 9839 7840.8   5946   54937

Percentage of the requests served within a certain time (ms)
  50%   5946
  66%   7936
  75%  14129
  80%  15935
  90%  17987
  95%  20656
  98%  30284
  99%  54602
 100%  54937 (longest request)
client1@localhost:~$ ab -n 5000 -c 1000 http://10.0.2.6/_
```

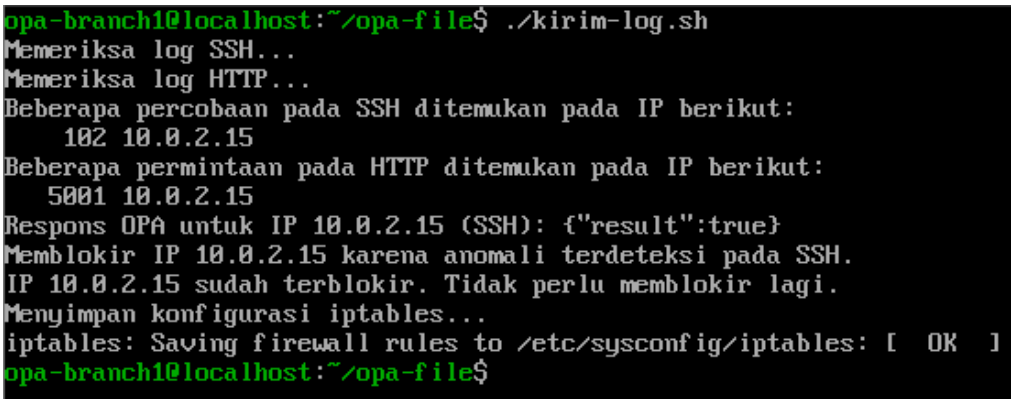
Gambar 8. Simulasi DdoS dengan apache benchmark

**d. Implementasi file bash script**

Untuk memudahkan proses otomatisasi permintaan evaluasi, admin membuat file bash script yang dijalankan di server cabang yang isinya dimulai dengan memeriksa apakah IP sudah pernah diblokir, pemeriksaan log HTTP dan SSH, lalu mengirimkan data log yang sudah diproses ke OPA untuk dievaluasi, dan yang terakhir menyimpan konfigurasi iptables jika terjadi pemblokiran. Langkah sebelumnya, client melakukan 2 cara untuk akses layanan pada server cabang. Dapat dilihat, terdapat 2 output yang berbeda setelah menjalankan file bash scriptnya, output gambar 8 adalah hasil dari akses normal yang dilakukan client dan output gambar 9 adalah akses simulasi serangan yang dilakukan client.

```
opa-branch1@localhost:~/opa-file$ ./kirim-log.sh
Memeriksa log SSH...
Memeriksa log HTTP...
Beberapa percobaan pada SSH ditemukan pada IP berikut:
      3 10.0.2.15
Beberapa permintaan pada HTTP ditemukan pada IP berikut:
      3 10.0.2.15
Respons OPA untuk IP 10.0.2.15 (SSH): {"result":false}
Respons OPA untuk IP 10.0.2.15 (HTTP): {"result":false}
Menyimpan konfigurasi iptables...
iptables: Saving firewall rules to /etc/sysconfig/iptables: [  OK  ]
opa-branch1@localhost:~/opa-file$ _
```

Gambar 9. Respon False yang Tidak Memblokir




Gambar 10. Respon True yang Memblokir Akses

### 4.4 Hasil Pengujian Peforma

Pengujian pada sistem yang dikembangkan menggunakan metode black-box. Pengujian metode black-box merupakan metode pengujian yang berfokus pada pengujian fungsionalitas sistem tanpa memperhatikan detail implementasi atau struktur internalnya. Pengujian ini bertujuan untuk memastikan bahwa sistem berfungsi sesuai dengan spesifikasi dan menghasilkan keluaran yang benar berdasarkan masukan tertentu [16]. Hasil dari pengujian dapat dilihat melalui tabel dibawah ini.

Tabel 4. Hasil Pengujian BlackBox

| No | Kondisi awal | Skenario Pengujian | Idnikator Keberhasilan | Hasil pengujian |
|---|---|---|---|---|
| 1. | Menjalankan server OPA | Admin Menjalankan server opa pada server pusat | Server merespon dengan status HTTP 200 . | Berhasil |
| 2. | Menghentikan server OPA | Admin memeriksa proses yang berjalan di sistem dan kemudian menghentikan prosesnya | Server opa menampilkan output server shutdown. | Berhasil |
| 3. | Menglola file rego | File rego dibuat dan disunting di server pusat, kemudian admin memeriksa syntax rego | Tidak ada output error pada proses pengecekan. | Berhasil |
| 4. | Memuat file rego | Admin memuat rego ke dalam OPA melalui API | Rego yang dimuat mengembalikan output {}. | Berhasil |
| 5. | Mengelola file bash script | Admin membuat dan menyunting file bash script yang disimpan di server cabang dan menambahkan logika dasar ke dalam file script | Saat dijalankan di server cabang, file tidak menampilkan pesan error dan berfungsi sesuai logika yang diinginkan. | Berhasil |
| 6. | Memantau log aktivitas | Server cabang memeriksa log SSH dan HTTP setiap menit | Menampilkan aktivitas pada port SSH dan HTTP. | Berhasil |
| 7. | Permintaan evaluasi | Server cabang mengirim permintaan evaluasi akses ke OPA di server pusat | Server cabang menerima respon OPA dari server pusat. | Berhasil |
| 8. | Penetapan iptables | Server cabang menerapkan iptables berdasarkan respon server pusat | Jika respon true, iptables otomatis memblokir, jika false iptables akan mengabaikan. | Berhasil |
| 9. | Mengakses layanan server cabang | Client melakukan akses ssh dan permintaan http | Akan terblokir jika upaya melebihi akses yang diterapkan. | berhasil |

## 5. KESIMPULAN DAN SARAN

Sistem yang dikembangkan dalam proyek ini mampu memberikan solusi untuk pengelolaan kebijakan akses secara terpusat di lingkungan multi-server. Sistem ini secara otomatis memantau dan mengirimkan log SSH dan HTTP dari server cabang ke server pusat untuk dievaluasi oleh OPA. Evaluasi kebijakan dilakukan secara otomatis, dengan hasil keputusan yang diterapkan menggunakan iptables untuk memblokir akses. Namun automatisasi pengelolaan log dan respons kebijakan masih mengandalkan bash script yang dijalankan dengan cronjob, yang dapat dioptimalkan dengan menggunakan alat otomasi lainnya seperti Ansible atau SaltStack di masa depan.

## DAFTAR PUSTAKA


[1] A. Z. Y. M. and A. H. Jatmika, "OPTIMASI PORT KNOCKING DAN HONEYPOT MENGGUNAKAN IPTABLES SEBAGAI KEAMANAN JARINGAN PADA SERVER," *Jurnal Teknologi Informasi,* vol. 3, no. 2, pp. 189-199, September 2021.

[2] V. V, S. and G. S, "APPLICATION GATEWAY DAN STATEFUL INSPECTION METHOD," *Jurnal Pengembangan Rekayasa dan Teknologi,* vol. 4, no. 2, pp. 87-97, November 2020.

[3] I. Novikov, "Security Boulevard," Techstrong Group, 10 January 2024. [Online]. Available: https://securityboulevard.com/2024/01/what-is-policy-as-code/...

[4] M. Gunasekaran, "Implementing Policy as Code through Open Policy Agent," 15 September 2001. [Online]. Available: https://insights.sei.cmu.edu/library/implementing-policy-as-code-through-open-policy-agent/..

[5] Urvil38, "open-policy-agent," Github, 29 July 2019. [Online]. Available: https://github.com/open-policy-agent/contrib/blob/main/opa-iptables/docs/tutorial.md.

[6] J. Moedjahedy, "Implementasi Cron Job Linux Sebagai Bel Pergantian Kelas Otomatis Di Universitas Klabat," *Cogito Smart Journa,* vol. 4, no. 1, pp. 1-10, June 2018.

[7] N. Ehrman, "What is Open Policy Agent (OPA)? Best Practices + Applications," WIZ, 18 September 2024. [Online]. Available: https://www.wiz.io/academy/open-policy-agent-opa.

[8] K. Wilhelmus, "Perancangan Sistem Informasi Perpustakaan Berbasis WebMenggunakkanMetode Waterfall," *Jurnal Sains Teknologi dan Sistem Informasi,* vol. 2, no. 1, pp. 47-51, April 2022.

[9] N. H, N. L. Azizah and S. B, "SISTEM INFORMASI DESA BERBASIS WEB DENGAN MENGGUNAKAN METODE WATERFALL," *Jurnal Ilmiah Penelitian dan Pembelajaran Informatika,* vol. 9, no. 1, pp. 264-271, March 2024.

[10] S. N. Adzimi, H. A. Alfasih, F. N. G. Ramadhan, S. N. Neyman and A. Setiawan, "Implementasi Konfigurasi Firewall dan Sistem Deteksi Instruksi menggunakan Debian," *Journal of Internet and Software Engineering,* vol. 4, no. 1, pp. 1-12, 2024.

[11] E. Lila, "ANALISIS KEBUTUHAN FUNGSIONAL APLIKASI PENANGANANKELUHAN MAHASISWA STUDI KASUS:STMIK ROSMA KARAWANG," *Jurnal Inovasi Pendidikan dan Teknologi Informasi,* vol. 2, no. 1, pp. 8-17, 2021.

[12] A. A. Aziiza and A. N. Fadhilah, "Analisis Metode Identifikasi dan Verifikasi Kebutuhan Non Fungsional," *Journal Technology and Computing Science Journa,* vol. 3, no. 1, pp. 13-21, 2020.

[13] E. Apipah and M. , "ANALISA DAN PERANCANGAN JARINGAN KOMPUTER MENGGUNAKAN TEKNOLOGI NIRKABEL BERBASIS WIFI," *JURNAL TEKNIK INFORMATIKA STMIK ANTARA BANGSA,* vol. 2, no. 1, pp. 92-101, 2016.

[14] C. Prihantoro and H. Witriyono, "Perancangan Client Server Three Tier Pada Pembangunan Web Service Anggota Perpustakaan Universitas Muhammadiyah Bengkulu," *Jurnal Universitas Muhammadiyah Bengkulu,* vol. 2, no. 2, pp. 68-73, 2019.

[15] Maspupah, "LITERATURE REVIEW:ADVANTAGES AND DISADVANTAGES OF BLACK BOX AND WHITE BOX TESTING METHODS," *Journal of Computing and Information Technology,* vol. 21, no. 2, pp. 151-162, September 2024.

[16] S. M. A. Satria, K. Mardina, J. and D. Aribowo, "Implementasi Teknologi Komunikasi Data Menggunakan Open System Interconnection(OSI) Untuk Berkirim Pesan Antar Perangkat," *Jurnal Publikasi Rumpun Ilmu Teknik,* vol. 2, no. 3, pp. 124-129, June 2024.

[17] M. Jufri and H. , "PENINGKATAN KEAMANAN JARINGAN WIRELESS DENGAN MENERAPKAN SECURITY POLICY PADA FIREWAL," *Journal Of Information System And Informatics Engineering,* vol. 5, no. 2, pp. 98-108, December 2021.

[18] D. R. Az Zahra, F. P. Ilham, H. N. Ramdhani and A. Setiawan, "Penerapan dan Pengujian Keamanan SSH Pada Server Linux menggunakan Hydra," *Journal of Internet and Software Engineering, vol,* vol. 1, no. 3, pp. 1-10, 2024.

[19] A. R. Nisa, A. D. Wijayanto, A. P. J. Priana and A. Setiawan, "Analisis Log Server untuk mendeteksi Serang DDoS pada Keamaan Jaringan di Website," *Journal of Internet and Software Engineering,* vol. 1, no. 3, pp. 1-17, 2024.

[20] N. R. Damayanti and M. Sobri, "JARINGAN WIRELESS LAN AUTHENTICATION CAPTIVE PORTAL," *SEMINAR SANTIKA,* pp. 97-106, September 2019.

[21] F. Jawad, S. M. R. A. Amalia and T. S. Nadzarudien, "Optimalisasi Keamanan dan Monitoring Jaringan Infrastruktur di Kantor DPRD Bekasi," *Jurnal Aplikasi Riset kepada Masyarakat,* vol. 3, no. 2, pp. 184-189, 15 January 2023.

[22] C. K. Wilujeng and A. Voutama, "IMPLEMENTASI FIREWALL FILTER RULES SEBAGAI FILTERING CONTENT PADA JARINGAN KOMPUTER MENGGUNAKAN MIKROTIK," *Jurnal Mahasiswa Teknik Informatika,* vol. 8, no. 3, pp. 2680-2685, June 2024.