Enhanced Security of Linux Server-Based Servers with a Combination of iptables and Knocking Ports

Peningkatan Keamanan *Server* Berbasis *Linux Server* dengan Kombinasi *iptables* dan *Port Knocking*

Surya Tri Atmaja Ramadhani[1], Fiyas Mahananing Puri[2], Amirudin Khorul Huda[3]
[1]Teknik Informatika, Fakultas Ilmu Komputer, Universitas Amikom Yogyakarta
[2]Sistem Informasi, Fakultas Ilmu Komputer, Universitas Amikom Yogyakarta
[3]Informatika, Fakultas Ilmu Komputer, Universitas Amikom Yogyakarta
E-mail: *[1] surya@amikom.ac.id, [2]fiyas@amikom.ac.id, [3]amirudinkh@amikom.ac.id

Abstract – *Ensuring server security is crucial for effectively managing information technology infrastructure, particularly in the current era of growing cyber threats. Server security is a critical component in the management of information technology infrastructure, especially in the face of increasingly sophisticated cyber threats. This research aims to enhance the security of Linux-based servers by implementing a combination of iptables and port knocking techniques as an additional layer of protection against attacks targeting open ports, particularly the SSH port. The methodology used in this study is the Network Development Life Cycle (NDLC), involving stages of analysis, design, simulation, implementation, and monitoring. Simulations were conducted on Ubuntu Linux servers with attack scenarios targeting the SSH port using port scanning and brute force techniques. The results of the study indicate that the combination of iptables and port knocking significantly improves server security by hiding ports from scanning and preventing unauthorized access. Testing showed that after the implementation of this combination, the success rate of port scanning attacks decreased from 100% to 0%, and brute force attacks from 60% to 0%. In conclusion, this approach is effective in protecting critical ports like SSH without compromising server performance, making it a practical and easily implementable solution for enhancing server security*.

Keywords *— firewall, iptables, server security, Linux, port knocking*

Abstrak – Menjamin keamanan *server* sangat penting untuk mengelola infrastruktur teknologi informasi secara efektif, terutama di era saat ini ancaman *cyber* meningkat. Keamanan *server* merupakan komponen kritis dalam manajemen infrastruktur teknologi informasi, terutama dalam menghadapi ancaman siber yang semakin canggih. Penelitian ini bertujuan untuk meningkatkan keamanan *server* berbasis *Linux* dengan mengimplementasikan kombinasi antara *iptables* dan teknik *port knocking* sebagai lapisan tambahan perlindungan terhadap serangan yang menargetkan *port* terbuka, khususnya *port SSH*. Metodologi yang digunakan dalam penelitian ini adalah *Network Development Life Cycle (NDLC)*, yang melibatkan tahapan analisis, desain, simulasi, implementasi, dan monitoring. Simulasi dilakukan pada *server Ubuntu Linux* dengan skenario serangan yang menargetkan *port SSH* menggunakan teknik *port scanning* dan *brute force*. Hasil penelitian menunjukkan bahwa kombinasi *iptables* dan *port knocking* secara signifikan meningkatkan keamanan *server* dengan menyembunyikan *port* dari pemindaian dan mencegah akses tidak sah. Pengujian menunjukkan bahwa setelah implementasi kombinasi tersebut, tingkat keberhasilan serangan *port scanning* menurun dari 100% menjadi 0%, dan serangan brute force dari 60% menjadi 0%. Kesimpulannya, pendekatan ini efektif dalam melindungi *port* kritis seperti *SSH* tanpa mengorbankan kinerja *server*, menjadikannya solusi yang praktis dan mudah diimplementasikan untuk meningkatkan keamanan *server*.

Kata Kunci — *firewall, iptables*, keamanan *server*, *Linux, port knocking*

 doi: https://doi.org/10.32503/jtecs.v4i2.5541

1. PENDAHULUAN

Menjamin keamanan *server* adalah elemen penting dalam mengelola infrastruktur teknologi informasi secara efektif, terutama di era saat ini dari kemajuan digital ketika ancaman *cyber* meningkat dalam frekuensi dan kompleksitas. Direktorat Operasi Keamanan Siber Id-SIRTII/CC pada kuartal tahun 2023 melaporkan bahwa salah satu serangan terhadap *server* yang sering digunakan oleh peretas adalah serangan melalui *port SSH* (22) yaitu tercatat sebanyak 21552 serangan [1],[2]. Peretas memiliki kemampuan untuk memasang pemindai *port* untuk melacak sistem lain atau menjual alamat *IP* dan kredensial akun yang telah dilanggar di *darkweb*, pelaku ancaman menggunakan serangan *bruteforce* untuk mencoba menebak kredensial *SSH server* [3], setelah berhasil menggunakan *server SSH* yang tidak diamankan dengan baik, peretas memiliki banyak pilihan untuk melakukan serangan lainnya lebih lanjut [4]. Contohnya pada situs resmi Majelis Ulama Indonesia yang berhasil disusupi peretas, pada situs tersebut ditemukan potensi risiko serangan *Malware, trojan, virus*, dan celah pemindaian *port* yang mengakibatkan layanan pada situs tersebut tidak dapat diakses [5].

Satu langkah yang banyak digunakan untuk memperkuat keamanan *server* adalah implementasi *firewall* [6], yang bertindak sebagai penghalang utama terhadap berbagai kerentanan jaringan. *Iptables* adalah aplikasi *firewall* yang sangat populer untuk sistem berbasis *Linux* karena keunikan dan efektivitasnya dalam mengontrol lalu lintas jaringan. Namun, bergantung hanya pada *iptables* mungkin tidak cukup dalam menangani serangan yang lebih rumit. Penelitian [7] menekankan pentingnya langkah-langkah dasar keamanan seperti *firewall* dan teknik tambahan untuk memperkuat perlindungan.

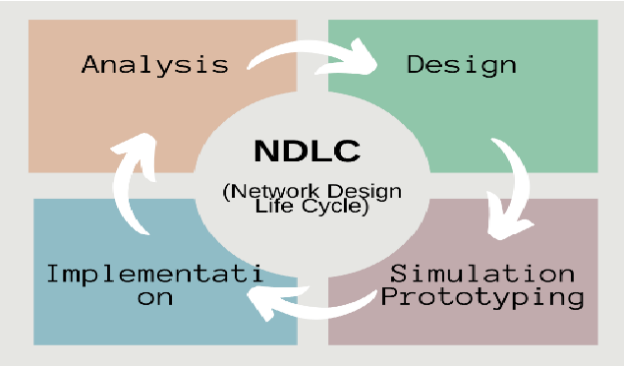
Dengan permasalahan tersebut penulis penerapkan konsep *port knocking* yang dapat meningkatkan keamanan dengan menambahkan lapisan perlindungan tambahan bersama-sama dengan *iptables* untuk meningkatkan keamanan terhadap serangan yang tidak sah sebagai teknik tambahan keamanan *firewall* untuk memperkuat perlindungan serangan terhadap *port SSH* [8], [9]. *Port knocking* [10] adalah metode dimana *server* secara selektif mengizinkan akses ke *port* tertentu hanya ketika menerima urutan sinyal atau ketukan tertentu di *port* lain. Sederhananya, akses ke *port* tertentu hanya diizinkan setelah urutan koneksi yang telah ditentukan sebelumnya telah diterima [11]. Dengan mengintegrasikan *iptables* dengan *port knocking*, adalah mungkin untuk membangun sistem *firewall* yang lebih kuat yang menyembunyikan *port* yang rentan dari pemindaian dan memungkinkan akses hanya kepada mereka yang memiliki urutan yang diperlukan "ketukan" [12]. Meskipun teknik dasar seperti penggunaan *iptables* sebagai *firewall* sangat penting, ada kebutuhan yang meningkat untuk solusi keamanan tambahan dan lebih kompleks untuk menangani ancaman yang terus berkembang. Kombinasi *iptables* dengan *port knocking* menawarkan pendekatan yang lebih holistik dan efektif untuk meningkatkan keamanan *server* dengan menyembunyikan *port* yang rentan dari pemindaian dan hanya mengizinkan akses setelah urutan koneksi tertentu diterima [13]. Penelitian tentang penerapan *port* knocking juga pernah dilakukan SMAN 5 Surakarta [14] menggunakan teknik zona demilitarisasi *(DMZ)* untuk mendapatkan akses aman ke *server* lokal yang digunakan dalam ujian daring. Selain itu, teknik *Port Knocking* digunakan untuk membuka *port* akses yang terfilter dan berhasil untuk menangkal serangan *DDoS*, yang merupakan serangan *DoS* yang menggunakan berbagai sumber daya serangan yang terdistribusi.

Studi ini berfokus pada penggunaan kombinasi *iptables* dan *port knocking* untuk meningkatkan keamanan *server* berbasis *Ubuntu Server Linux*. Tujuan utama dari penelitian ini adalah untuk mengembangkan dan mengimplementasikan pendekatan keamanan yang lebih kuat dan efisien dalam mengatasi ancaman jaringan yang terus berkembang terutama pada *port SSH* dengan mengintegrasikan dua teknik yang sudah terbukti efektif secara individu. Penelitian ini mengklarifikasi kompleksitas dan prosedur yang terlibat dalam implementasi kombinasi *iptables* dan *port knocking* pada *server Ubuntu Linux* serta mengevaluasi efektivitas kombinasi ini dalam mencegah serangan yang berfokus pada eksploitasi *port* terbuka. Hasil penelitian ini diharapkan dapat memberikan kontribusi yang signifikan terhadap pengembangan strategi keamanan siber, khususnya dalam konteks pengamanan infrastruktur teknologi informasi yang semakin kritis. Studi ini membahas implementasi kombinasi *Iptables* dan mengetuk *Port* pada *server Linux Ubuntu Server*.

2. METODE PENELITIAN

2.1. NDLC (Network Development Life Cycle)

Penelitian ini menggunakan metodologi *Network Development Life Cycle (NDLC)*, sebuah pendekatan sistematis yang dirancang untuk pengembangan dan implementasi jaringan komputer. *NDLC* dipilih karena kemampuannya untuk menyediakan kerangka kerja yang komprehensif dalam merancang, mengembangkan, dan menguji sistem jaringan yang kompleks, seperti *server* keamanan berbasis Linux. *NDLC* adalah proses sistematis untuk merancang dan mengembangkan jaringan komputer. Ini melibatkan serangkaian langkah atau fase yang menghasilkan penciptaan output yang disesuaikan agar dapat mengoptimalkan fungsi jaringan yang ada melalui pendekatan *Network Development Life Cycle* (NDLC) [15]. Untuk penelitian tentang penerapan kombinasi *iptables* dan *port knocking* sebagai *firewall* pada *server* berbasis *Linux Ubuntu Server* adalah sebagai berikut:



Gambar 1. Metode *NDLC*

Pada Gambar 1, Tahapan dalam *NDLC* dilakukan untuk merancang, mengembangkan, dan menguji sistem pada *server* keamanan berbasis *Ubuntu Linux Server*. Tahapan yang diterapkan dalam penelitian ini meliputi:

2.1.1. Analysis

Pada tahap ini, dilakukan pendataan kondisi topologi jaringan agar dapat dilakukan analisis kebutuhan, pendataan kondisi lingkungan *server* sebelum implementasi untuk menganalisis masalah yang muncul, pendataan proyek jaringan untuk analisis keinginan pengguna, dan dokumentasi yang diperlukan setelah sistem selesai dibangun serta analisis topologi jaringan saat ini dilakukan.

2.1.2. Design

Berdasarkan hasil analisis kebutuhan, dilakukan perancangan desain konfigurasi *iptables* dan skenario *port knocking*. model prototipe jaringan dirancang menggunakan alat simulasi seperti *Netcat* untuk memvalidasi aturan-aturan keamanan yang diimplementasikan. Pemilihan alat simulasi ini didasarkan pada kemampuannya untuk meniru kondisi jaringan secara akurat, sehingga memungkinkan pengujian awal sebelum implementasi pada *server* sebenarnya.

2.1.3. Simulation Prototyping

Pada tahap simulasi, sistem keamanan yang terdiri dari kombinasi *iptables* dan *port knocking* diimplementasikan pada *server Ubuntu Linux*. Simulasi ini mencakup uji coba berbagai konfigurasi *iptables* dan urutan *port knocking* untuk memastikan efektivitas dalam menyembunyikan dan mengamankan *port* kritis seperti *SSH*. Tahap ini melibatkan penerapan konfigurasi *firewall* dengan *bash scripting* untuk memastikan keamanan yang optimal. Pemilihan *iptables* dan *port knocking* didasarkan pada bukti empiris dan literatur sebelumnya yang menunjukkan efektivitas kombinasi ini dalam mencegah akses tidak sah ke *server*.

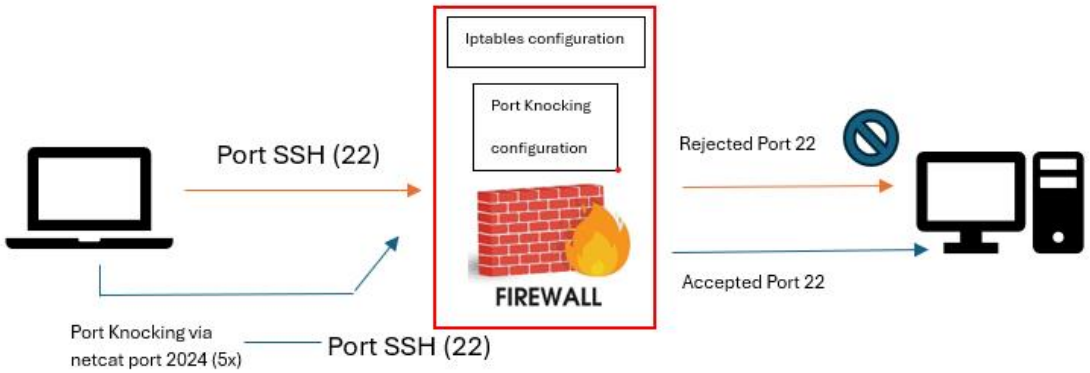
 doi: https://doi.org/10.32503/jtecs.v4i2.5541

2.1.4. Implementation and Monitoring

Setelah sistem diimplementasikan, dilanjutkan dengan implementasi dan pemantauan kinerja dan keamanan *server* untuk menilai efektivitas konfigurasi yang telah diinstal. Implementasi *firewall Server* dipantau secara intensif selama beberapa waktu untuk mengevaluasi kinerja dan keamanan sistem yang telah dikonfigurasi. Monitoring dilakukan dengan menggunakan alat seperti *iptables-logs, port scanning, dan bruteforce* untuk mendeteksi aktivitas jaringan dan potensi serangan. Selain itu, pengujian penetrasi dilakukan untuk menilai ketahanan *server* terhadap serangan dari luar. Pemantauan ini penting untuk memastikan bahwa sistem berfungsi sesuai dengan yang diharapkan tanpa mengganggu kinerja *server*.

2.2. Skenario Kasus

Skenario kasus yang akan di ujicobakan ini bersifat simulasi serangan dengan melakukan akses *remote server* melalui *port ssh* didalam sebuah jaringan yang saling terhubung antara komputer penyerang dan *server* target, konfigurasi *firewall* akan diterapkan pada komputer *server* dengan menghapus semua konfigurasi *default* terlebih dahulu, kemudian mengkombinasikan antara *iptables* dan *port knocking* sebagai *firewall* pada *server*. Skenario yang akan digunakan dalam penelitian ini adalah sebagai berikut :



Gambar 2. Skenario Kasus

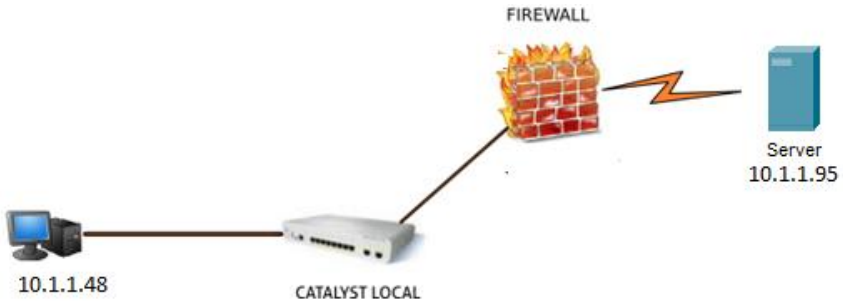
Gambar 2 mengilustrasikan dari kasus serangan yang nantinya akan di ujicobakan dengan kombinasi *iptables* dan *port knocking*. Ilustrasi tersebut melibatkan 2 komputer yang terhubung pada sebuah jaringan dimana komputer penyerang menggunakan *ip address* 10.1.1.48, dan *server* menggunakan *ip address* 10.1.1.95. *Firewall server* akan dikonfigurasikan menggunakan kombinasi antara *iptables* dan *port knocking*, penyerang akan mencoba melakukan akses remote *server* ke *server* dengan menggunakan perintah ”*ssh -l <username><spasi><ip address server*>”. Umumnya akses remote *server linux* di akses melalui *ssh* secara normal, ketika penyerang berhasil menginputkan *username* dan *password* secara benar, maka penyerang akan mendapatkan akses terhadap *server* tersebut. Oleh karena itu kombinasi *iptables* dan *port knocking* akan diterapkan untuk mencegah penyerang leluasa untuk mengakses *port ssh* (22) dengan mengharuskan *user* tertentu yang akan mengakses *ssh* pada *server* harus mengetahui pola ketukan yang sudah ditentukan untuk dapat mengakses *port ssh* pada *linux server* tersebut.

Untuk mengukur efektifitas dari metode tersebut, monitoring dan pengujian akan dilakukan dengan dua metode serangan yaitu metode *port scanning* dan *bruteforce* yang akan dilakukan dengan 3 skenario yaitu sebelum implementasi, setelah implementasi menggunakan *iptables*, dan setelah implementasi menggunakan kombinasi *iptables* dan *port knocking*. Jumlah percobaan yang dilakukan pada masing-masing skenario adalah 100 percobaan. Ada 3 hasil yang akan menjadi catatan pada monitoring menggunakan *port scanning* yaitu berapa persen yang dideteksi oleh penyerang, berapa persen akses yang tidak sah, dan berapa waktu yang dibutuhkan. Sementara pada monitoring dan pengujian menggunakan teknik serangan *bruteforce* hanya akan mencatatkan berapa persen percobaan berhasil dan berapa waktu untuk akses pertama berhasil dalam satuan detik.

3. HASIL DAN PEMBAHASAN

3.1. *Analysis*

Hasil Analisa berdasarkan pendataan kondisi topologi jaringan yang terimplementasikan secara default, kondisi lingkungan *server* sebelum implementasi masih menggunakan konfigurasi bawaan dari iptables. Gambaran topologi jaringan yang digunakan dalam mengimplementasi *IPTABLES* dan *Port Knokcing* pada *Linux Server* berbasiskan ubuntu adalah seperti Gambar 3 sebagai berikut :



Gambar 3 Topologi Jaringan Awal

Gambar 3 secara umum memperlihatkan topologi yang digunakan dengan konfigurasi awal, dimana *firewall* secara *default* sudah terpasang dengan konfigurasi *default* juga. Untuk mengetahui *port* umum yang terbuka pada konfigurasi *default* dapat menggunakan perintah *"sudo ss -ltn*" pada terminal, outputnya akan memberitahu *port* yang sedang *listen* untuk koneksi. Untuk *port default* yang terbuka dapat dilihat pada Tabel 1berikut :

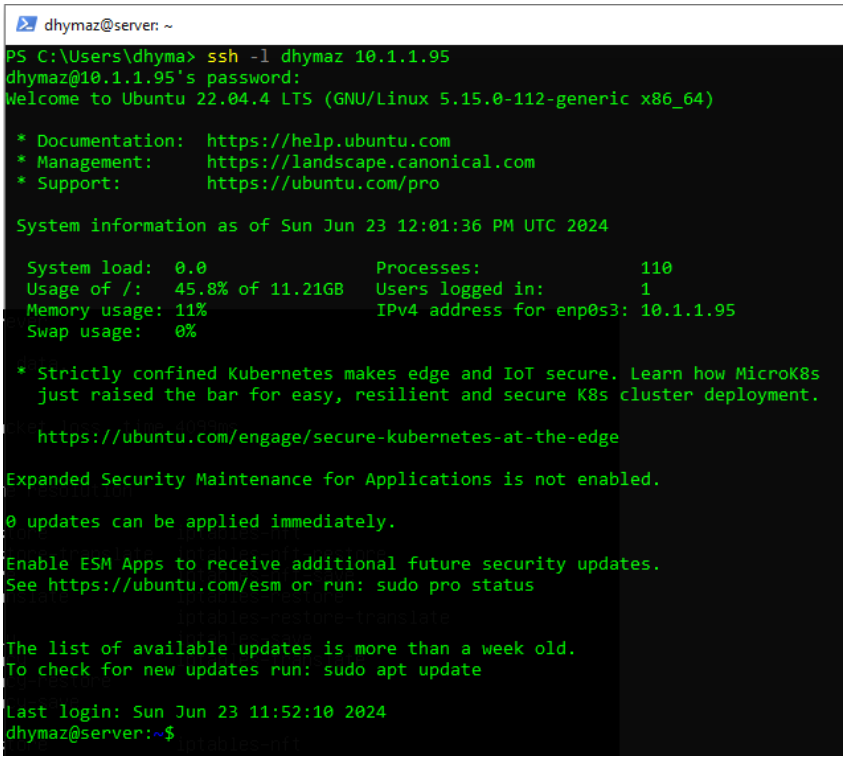
Tabel 1. *Port* Terbuka Secara *Default*

Port	Status
80	*Listen to nginx*
53	*Listen to systemd-resolver*
22	*Listen to sshd*
3000	*Listen to docker-proxy*
443	*Listen to nginx*
1993	*Listen to docker-proxy*

Pada Tabel 1 menunjukkan ada 6 *port* yang terbuka secara default yang bebas di akses secara umum termasuk *port* 22 yang digunakan untuk akses SSH. Skenario *IP address server Linux Ubuntu* beserta perangkat lunak *port knocking* dengan konfigurasi *IP address* 10.1.1.95 adalah *IP* dari *server*, *subnet mask* 255.255.255.0 dan *gateway* 10.1.1.111 dengan *network interface* "enp0s3" yang digunakan sebagai *firewall* yang berisi aturan-aturan yang akan diterapkan untuk membatasi hak akses ketika ada seseorang yang ingin mengakses komputer *server*.

Port-knocking adalah metode yang digunakan untuk memverifikasi pengguna yang ingin mengakses *port* tertentu, seperti *port 22/SSH*, di jaringan *server*. Otentikasi ini didasarkan pada kombinasi yang telah ditentukan sebelumnya yang telah dicatat oleh *port-knocker*. Ini hanya memungkinkan pengguna yang sah untuk mengakses komputer *server*. Skenario pengujian sistem dilakukan dengan menggunakan *remote login via SSH*. Berikut adalah contoh pengujian pra-implementasi yang dilakukan oleh klien untuk mengakses *port* 22 atau SSH:

 doi: https://doi.org/10.32503/jtecs.v4i2.5541



Gambar 4 Ujicoba *port* 22

Gambar 4 merupakan skenario yang menunjukkan bahwa *port* 22 atau *SSH* dapat diakses oleh semua orang, yang menyajikan kerentanan potensial bagi peretas untuk mengeksploitasi melalui serangan *brute force*.

Sebelum diimplemetasikan, peneliti menghapus *rule-rule firewall* yang ada pada *server* (*rule* bawaan dari *Linux Ubuntu*), sedangkan *iptables -P INPUT DROP* adalah mengatur *default policy* untuk *chain INPUT* menjadi *DROP* yaitu menutup semua *port* yang ada pada *server* dengan perintah-perintah sebagai berikut :

```
root@server:~# iptables -F
root@server:~# iptables -F -t nat
root@server:~# iptables -F -t mangle
root@server:~# iptables -X
root@server:~# iptables -P INPUT DROP
```

Gambar 5. Baris Perintah Menutup Semua *port* yang Ada Pada *server*

Selanjutnya, lakukan perintah berikut: ***"iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT"*** Perintah ini mendefinisikan bahwa komunikasi dengan status "ESTABLISHED" dan "RELATED" diperbolehkan atau diizinkan. Selanjutnya, beberapa modifikasi konfigurasi diimplementasikan untuk memastikan bahwa *firewall* hanya memungkinkan untuk membuka *port SSH (22)* ketika komputer klien telah terdaftar di daftar *sshlist*. Pendaftaran ini didasarkan pada persyaratan untuk melakukan setidaknya 5 upaya koneksi dalam interval 15 detik. Aktifkan konfigurasi *Port knokcing* dengan menerapkan aturan yang mencatat alamat IP dari setiap komputer yang mencoba terhubung ke *port* 2024 ke dalam daftar yang disebut "*sshlist*". Untuk melihat daftar alamat IP, gunakan perintah *"cat /proc/net/ipt_recent/sshlist"*. Selain itu, jika ada upaya koneksi untuk *port* 1102, hapus alamat IP mesin yang mencoba mengakses *port* tersebut.

3.2. *Design*

Selama fase ini, arsitektur jaringan akan dibuat menggunakan alat simulasi yang spesifik untuk bidang jaringan, seperti *Netcat*. Jaringan yang di desain untuk di implementasikan sejak awal di rancang pada tahap ini, apakah kinerjanya normal atau tidak. *Setup iptables* dengan aturan dasar dan memasukkan *port knokcing* sebagai langkah tambahan perlindungan. Mengintegrasikan *firewall* ke dalam komputer *server* berdasarkan aturan yang ditetapkan untuk memastikan bahwa *firewall* beroperasi dan berfungsi dengan benar selama pengujian sistem. Pada tingkat ini, sistem ini diimplementasikan dengan menggunakan *IPTables* sebagai *firewall* yang terintegrasi ke dalam bahasa pemrograman *Linux bash script*. *Bash script* adalah *file* yang berisi kumpulan program yang dapat dijalankan. Untuk menjalankan *script bash*, gunakan tanda "." sebelum nama *file*, yang menunjukkan eksekusi *shell*, dan tanda "./". Ini berarti bahwa *file bash script* ada di direktori saat ini atau dapat dijalankan dengan menggunakan perintah "*bash <namefile>"*. Dalam skenario ini, *script Bash* menyederhanakan implementasi aturan *IPtables* yang dirancang untuk administrator jaringan. Ini menghilangkan kebutuhan untuk secara manual mengetik setiap sintaks dari aturan *IPTables* yang akan diterapkan pada komputer *server*. Sebagai gantinya, administrator hanya dapat menjalankan satu *file firewal*l yang sudah diinstal di *server*.

Setelah itu tahap selanjutnya ini adalah tahap yang krusial dikarenakan dengan konfigurasi inilah *firewall* pada *server* di bangun, dengan menggunakan *syntax-syntax* pada *IPTABLES*, *server FOSS* dapat menentukan kebijakan paket data mana saja yang dibolehkan lewat maupun di-*block* atau di-*drop*. Berikut konfigurasi *IPTABLES* yang diterapkan pada *server*.

```
# Setting ke policy default
iptables -P INPUT ACCEPT
iptables -P OUTPUT ACCEPT
iptables -P FORWARD ACCEPT

# Hapus semua rule yang ada (Flush)
iptables -F

# Membolehkan akses localhost
iptables -I INPUT -i lo -j ACCEPT
iptables -I OUTPUT -o lo -j ACCEPT

# Mengizinkan semua koneksi ke luar
iptables -A INPUT -i eth0 -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -o eth0 -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT

# Mengizinkan port http / https (open port 80 / 443) untuk diakses
iptables -A INPUT -i eth0 -p tcp --destination-port 80 -j ACCEPT
iptables -A INPUT -i eth0 -p tcp --destination-port 443 -j ACCEPT

# Memungkinkan ICMP ping pong stuff
iptables -A INPUT -i eth0 -p icmp --icmp-type 8 -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -o eth0 -p icmp --icmp-type 0 -m state --state ESTABLISHED,RELATED -j ACCEPT

# Mengizinkan port 53 tcp/udp (DNS Server) diakses
iptables -A INPUT -i eth0 -p udp --dport 53 -m state --state NEW,ESTABLISHED,RELATED -j ACCEPT
iptables -A OUTPUT -o eth0 -p udp --sport 53 -m state --state ESTABLISHED,RELATED -j ACCEPT

iptables -A INPUT -i eth0 -p tcp --destination-port 53 -m state --state NEW,ESTABLISHED,RELATED  -j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 53 -m state --state ESTABLISHED,RELATED -j ACCEPT

# Mengaktifkan fungsi logging supaya mudah menganalisa troubleshoot
iptables -A INPUT -m limit --limit 2/m --limit-burst 2 -j LOG --log-prefix '** INPUT DROP ** '
iptables -A FORWARD -m limit --limit 2/m --limit-burst 2 -j LOG --log-prefix '** FORWARD DROP ** '
iptables -A OUTPUT -m limit --limit 2/m --limit-burst 2 -j LOG --log-prefix '** OUTPUT DROP ** '
iptables -A INPUT -j DROP

# Terakhir mengatur default policy menjadi drop
iptables -P INPUT DROP
iptables -P OUTPUT DROP
iptables -P FORWARD DROP

#### Save konfigurasi iptables
service iptables save
```

Gambar 6. Konfigurasi *iptables* Yang Diterapkan

Setelah konfigurasi dilakukan, akan di-*review* kembali *rule port-knocking* dan *iptables* yang telah dijelaskan pada konfigurasi sistem dengan menggunakan *bash script* sehingga menjadi lebih mudah dalam implementasi pada *server* yang disimpan dengan nama *file firewall.sh* untuk di eksekusi pada *system*.

 doi: https://doi.org/10.32503/jtecs.v4i2.5541

```
root@server:/home/dhymaz# bash firewall.sh
Setting sysctl IPv4 settings...
net.ipv4.ip_forward = 0
net.ipv4.conf.all.send_redirects = 0
net.ipv4.conf.default.send_redirects = 0
net.ipv4.conf.all.accept_source_route = 0
net.ipv4.conf.all.accept_redirects = 0
net.ipv4.conf.all.secure_redirects = 0
net.ipv4.conf.all.log_martians = 1
net.ipv4.conf.default.accept_source_route = 0
net.ipv4.conf.default.accept_redirects = 0
net.ipv4.conf.default.secure_redirects = 0
net.ipv4.tcp_syncookies = 1
net.ipv4.conf.all.rp_filter = 1
net.ipv4.conf.default.rp_filter = 1
sysctl: cannot stat /proc/sys/kernel/exec-shield: No such file or directory
kernel.randomize_va_space = 1
Setting IPv4 Firewall...
root@server:/home/dhymaz# _
```

Gambar 7. Implementasi *File firewall.sh*

3.3. *Simulation Prototyping*

Tahapan *monitoring* sangat penting untuk memastikan bahwa apa yang dirancang pada tahap simulasi dapat berjalan sesuai dengan semestinya dan sesuai dengan apa yang dikonfigurasikan. Diawali dengan Melakukan implementasi konfigurasi *firewall* dengan melakukan *Iptables-persistent* agar konfigurasi tersimpan pada *file* tertentu dan tidak terhapus ketika *system* melakukan *restart* dan melakukan serangkaian pengujian *port SSH* dengan melakukan *login remote server* melalui *port SSH* dengan teknik *port knocking* yang telah ditentukan.

Setelah itu gunakan perintah *iptables -L* untuk melihat apakah *script* tersebut sudah terpasang pada komputer *server*. Berikut hasil *output* setelah mengetikkan perintah *iptables -L*.

```
root@server:/home/dhymaz# iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination
blokip     all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere
LOG        tcp  --  anywhere             anywhere            tcp flags:!FIN,SYN,RST,ACK/SYN state NEW limit: avg 5/min burst 7 LOG level warning prefix "Drop Sync"
DROP       tcp  --  anywhere             anywhere            tcp flags:!FIN,SYN,RST,ACK/SYN state NEW
LOG        all  -f  anywhere             anywhere            limit: avg 5/min burst 7 LOG level warning prefix "Fragments Packets"
DROP       all  -f  anywhere             anywhere
DROP       tcp  --  anywhere             anywhere            tcp flags:FIN,SYN,RST,PSH,ACK,URG/FIN,PSH,URG
DROP       tcp  --  anywhere             anywhere            tcp flags:FIN,SYN,RST,PSH,ACK,URG/FIN,SYN,RST,PSH,ACK,URG
LOG        tcp  --  anywhere             anywhere            tcp flags:FIN,SYN,RST,PSH,ACK,URG/NONE limit: avg 5/min burst 7 LOG level warning prefix "NULL Packets"
DROP       tcp  --  anywhere             anywhere            tcp flags:FIN,SYN,RST,PSH,ACK,URG/NONE
DROP       tcp  --  anywhere             anywhere            tcp flags:SYN,RST/SYN,RST
LOG        tcp  --  anywhere             anywhere            tcp flags:FIN,SYN/FIN,SYN limit: avg 5/min burst 7 LOG level warning prefix "XMAS Packets"
DROP       tcp  --  anywhere             anywhere            tcp flags:FIN,SYN/FIN,SYN
LOG        tcp  --  anywhere             anywhere            tcp flags:FIN,ACK/FIN limit: avg 5/min burst 7 LOG level warning prefix "Fin Packets Scan"
DROP       tcp  --  anywhere             anywhere            tcp flags:FIN,ACK/FIN
DROP       tcp  --  anywhere             anywhere            tcp flags:FIN,SYN,RST,PSH,ACK,URG/FIN,SYN,RST,ACK,URG
ACCEPT     all  --  anywhere             anywhere            state RELATED,ESTABLISHED
ACCEPT     tcp  --  anywhere             anywhere            state NEW tcp dpt:ssh recent: CHECK seconds: 15 hit_count: 5 name: sshlist side: source mask: 255.255.255.255
LOG        tcp  --  anywhere             anywhere            state NEW tcp dpt:2024 recent: SET name: sshlist side: source mask: 255.255.255.255 LOG level warning prefix "knock ssh: "
LOG        tcp  --  anywhere             anywhere            state NEW tcp dpt:1102 recent: REMOVE name: sshlist side: source mask: 255.255.255.255 LOG level warning prefix "knock ssh: "
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:http
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:https
ACCEPT     icmp --  anywhere             anywhere            icmp echo-request state NEW,RELATED,ESTABLISHED
ACCEPT     udp  --  anywhere             anywhere            udp dpt:domain state NEW,RELATED,ESTABLISHED
ACCEPT     tcp  --  anywhere             anywhere            tcp dpt:domain state NEW,RELATED,ESTABLISHED
LOG        all  --  anywhere             anywhere            limit: avg 2/min burst 2 LOG level warning prefix "**INPUT DROP ** "
DROP       all  --  anywhere             anywhere

Chain FORWARD (policy DROP)
target     prot opt source               destination
blokip     all  --  anywhere             anywhere
LOG        all  --  anywhere             anywhere            limit: avg 2/min burst 2 LOG level warning prefix "**FORWARD DROP ** "

Chain OUTPUT (policy DROP)
target     prot opt source               destination
blokip     all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere            state NEW,RELATED,ESTABLISHED
ACCEPT     icmp --  anywhere             anywhere            icmp echo-reply state RELATED,ESTABLISHED
ACCEPT     udp  --  anywhere             anywhere            udp spt:domain state RELATED,ESTABLISHED
ACCEPT     tcp  --  anywhere             anywhere            tcp spt:domain state RELATED,ESTABLISHED
LOG        all  --  anywhere             anywhere            limit: avg 2/min burst 2 LOG level warning prefix "**OUTPUT DROP ** "

Chain blokip (3 references)
target     prot opt source               destination
root@server:/home/dhymaz#
```

Gambar 8. *Output firewall* yang Diimplementasikan

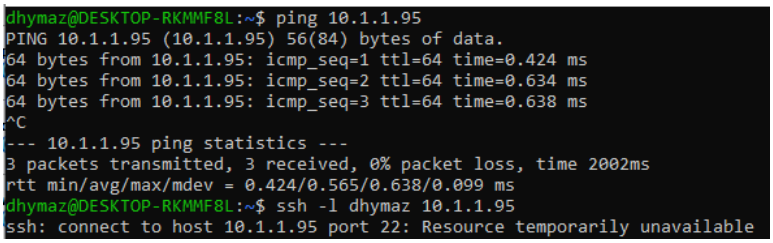
Output tersebut memperlihatkan konfigurasi yang dibangun pada *server* dan menggunakan pendekatan *positif* yaitu menutup seluruh *port* secara *default*, terlihat dari ketiga *chain* (*INPUT*, *OUTPUT*, dan *FORWARD*) menerapkan *policy DROP*.

Konfigurasi *firewall* yang telah diterapkan harus bersifat permanen. Untuk melakukannya, gunakan perintah "*apt install iptables-persistent*". Ini akan memungkinkan konfigurasi yang telah dibuat dan dimuat kembali ketika *server direstart*.

3.4. *Implementation and Monitoring*

Berdasarkan konfigurasi *port-knocking* dan *IPTables* yang telah diimplementasikan pada *server*, walaupun semua *port* secara *default* ditutup dan hanya mengizinkan beberapa *service* saja yang

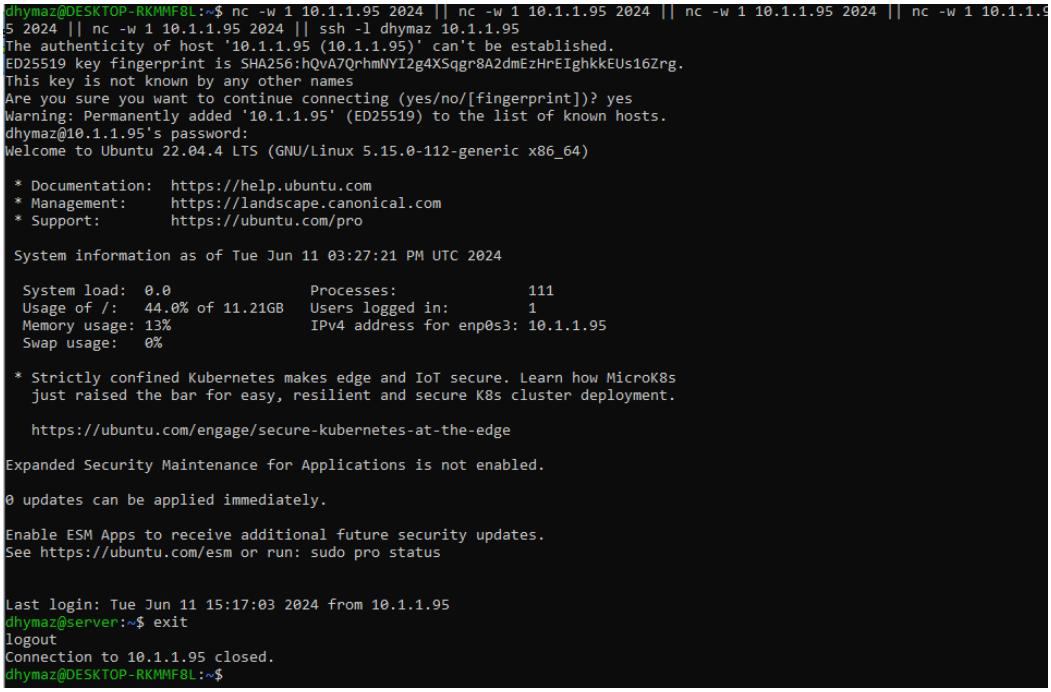
boleh diakses, *server* masih dapat diakses dari jaringan, *user* masih dapat melakukan *ping* ke *server* tersebut, namun tidak dapat melakukan *remote server* pada *port SSH* seperti pada gambar dibawah ini :



Gambar 9. Ping ke *ip server*

Setelah mencoba *login* SSH ke IP *server* 10.1.1.95, muncul pesan *ssh: connect to host 10.1.1.95 port 22: Connection timed out*. Ini menunjukkan bahwa pengguna tidak dapat melakukan *login SSH* ke komputer *server*. Ini menunjukkan bahwa metode *port-knocking* yang digunakan sebelumnya telah berhasil.

Karena metode *port-knocking server* berfungsi, seorang user dapat mengakses komputer *server* dengan mengetahui kombinasi *port* yang telah diterapkan pada *iptables*. Penelitian ini menguji *port* tertentu (*port 2024*) lima kali menggunakan *tool netcat*, dan setelah itu, *port* tersebut telah terbuka untuk *user* yang diizinkan, seperti yang ditunjukkan pada gambar di bawah ini :



Gambar 10. Ujicoba *port knocking*

Tool alat *netcat* dengan opsi *-w* menunjukkan waktu atau *timeout* dalam melakukan percobaan ke *port 2024* dalam 15 detik berdasarkan *rule iptables* yang diimplementasikan ke dalam *server*, dan angka 1 menunjukkan *timeout -w* bernilai positif. Jika percobaan login berhasil membuka *port 22*, alamat IP yang melakukan login akan dimasukkan ke dalam *file /proc/net/ipt_recent/sshlist*.

Monitoring yang telah dilakukan untuk mengukur efektifitas keamanan *server* menggunakan kombinasi *iptables* dan *port knocking* dengan membandingkan tingkat keberhasilan serangan pada *server* sebelum dan sesudah implementasi. Pengujian dilakukan dalam dua skenario utama: *port scanning* dan *brute force* pada *port SSH* pada Tabel 2 dan 3.

 doi: https://doi.org/10.32503/jtecs.v4i2.5541

Tabel 2. Hasil Pengujian *Port Scanning*

Skenario	Jumlah Percobaan	Deteksi oleh Penyerang	Akses Tidak Sah	Waktu Akses (detik)
Sebelum implementasi	100	100%	80%	< 2
Setelah implementasi *iptables*	100	20%	0%	Tidak ada akses
Setelah implementasi *iptables + port knocking*	100	0%	0%	Tidak ada akses

Tabel 2 menampilkan perbandingan antara tingkat keberhasilan serangan *port scanning* pada tiga kondisi: sebelum implementasi, setelah implementasi *iptables*, dan setelah implementasi kombinasi *iptables* dan *port knocking.*

Tabel 3. Hasil Pengujian *brute force* pada *port ssh*

Skenario	Jumlah Percobaan	Jumlah Percobaan Berhasil	Waktu Untuk Akses Pertama Berhasil (detik)
Sebelum implementasi	100	60%	< 5
Setelah implementasi *iptables*	100	10%	> 60
Setelah implementasi *iptables + port knocking*	100	0%	Tidak ada akses

Tabel 3 menunjukkan perbandingan jumlah percobaan yang berhasil dalam serangan *brute force* sebelum dan sesudah implementasi, serta perbedaan waktu yang dibutuhkan untuk mendapatkan akses pertama. Kombinasi *iptables* dan *port knocking* telah ditunjukkan untuk meningkatkan keamanan *server* karena menyembunyikan *port ssh* dari pemindaian dan mencegah akses yang tidak sah. Dalam skenario pengujian, metode ini berhasil menunjukkan bahwa itu sangat efektif dalam mencegah serangan *port scanning* dari 100% menjadi 0% dan *serangan brute force* dari 60% menjadi 0%. Ini menunjukkan bahwa metode ini sangat efektif dalam mencegah serangan ke *port* penting seperti *SSH*.

4. KESIMPULAN

Studi ini telah mengevaluasi implementasi kombinasi *iptables* dan *port knocking* sebagai *firewall* pada *server Ubuntu Server* berbasis *Linux*. Berdasarkan hasil penelitian, dapat disimpulkan bahwa kombinasi iptables dan *port knocking* secara signifikan meningkatkan keamanan *server* berbasis Ubuntu Linux dengan menyembunyikan *port* dari pemindaian dan mencegah akses tidak sah, tanpa mengorbankan kinerja *server*. Metode ini mudah diimplementasikan dan menawarkan solusi praktis untuk meningkatkan keamanan *server*. Pada skenario pengujian, implementasi metode ini berhasil mengurangi tingkat keberhasilan serangan *port* scanning dari 100% menjadi 0%, dan serangan *brute force* dari 60% menjadi 0%. Hal ini menunjukkan bahwa metode ini sangat efektif dalam melindungi *port* kritis seperti SSH dari eksploitasi.

Saran untuk penelitian selanjutnya dapat mengembangkan metode yang lebih canggih, seperti integrasi dengan teknik enkripsi dalam urutan *port knocking* dan menguji efektivitas metode ini dalam skala yang lebih besar dan dalam berbagai kondisi jaringan.

DAFTAR PUSTAKA

[1] Direktorat Operasi Keamanan Cyber, "CYBERBLITZ," *Id-SIRTII/CC*, vol. 10, no. 186, 2023.

[2] Pusat Tanggap Darurat Keamanan AhnLab (ASEC), "Statistics Report on Malware Targeting Linux SSH Servers in Q4 2023 – ASEC." Accessed: Aug. 18, 2024. [Online]. Available: https://asec.ahnlab.com/en/78949/

[3] A. Zainal, S. Syaifuddin, and D. Risqiwati, "IMPLEMENTASI ASYMMETRIC ENCRYPTION RSA PADA PORT KNOCKING UBUNTU SERVER MENGGUNAKAN KNOCKD DAN PYTHON," *Jurnal Repositor*, vol. 2, p. 787, Apr. 2020, doi: 10.22219/repositor.v2i6.270.

[4] R. Ernawati *et al.*, "Coding : Jurnal Komputer dan Aplikasi IMPLEMENTASI METODE PORT KNOCKING PADA SISTEM KEAMANAN SERVER UBUNTU VIRTUAL BERBASIS WEB MONITORING."

[5] N. A. Santoso, K. B. Affandi, and R. D. Kurniawan, "Implementasi Keamanan Jaringan Menggunakan Port Knocking," *Jurnal Janitra Informatika dan Sistem Informasi*, vol. 2, no. 2, pp. 90–95, Oct. 2022, doi: 10.25008/janitra.v2i2.156.

[6] R. W. Anwar, T. Abdullah, and F. Pastore, "Firewall best practices for securing smart healthcare environment: A review," Oct. 01, 2021, *MDPI*. doi: 10.3390/app11199183.

[7] I. H. Sarker, A. S. M. Kayes, S. Badsha, H. Alqahtani, P. Watters, and A. Ng, "Cybersecurity data science: an overview from machine learning perspective," *J Big Data*, vol. 7, no. 1, Dec. 2020, doi: 10.1186/s40537-020-00318-5.

[8] Ö. Aslan, S. S. Aktuğ, M. Ozkan-Okay, A. A. Yilmaz, and E. Akin, "A Comprehensive Review of Cyber Security Vulnerabilities, Threats, Attacks, and Solutions," Mar. 01, 2023, *MDPI*. doi: 10.3390/electronics12061333.

[9] I. Putri, A. Agita, and S. Soim, "Implementasi Port Knocking, Port Blocking Pada Keamanan Jaringan Komputer Berbasis Mikrotik," *Journal Rekayasa Sistem Komputer*, vol. 6, no. 3, 2023, [Online]. Available: https://s.id/jurnalresistor

[10] V. Tanksale, "Intrusion detection system for controller area network," *Cybersecurity*, vol. 7, no. 1, Dec. 2024, doi: 10.1186/s42400-023-00195-4.

[11] V. Srivastava, A. K. Keshri, A. D. Roy, V. K. Chaurasiya, and R. Gupta, "Advanced port knocking authentication scheme with QRC using AES," *2011 International Conference on Emerging Trends in Networks and Computer Communications (ETNCC)*, pp. 159–163, 2011, doi: 10.1109/ETNCC.2011.5958506.

[12] F. Cremer *et al.*, "Cyber risk and cybersecurity: a systematic review of data availability," *Geneva Papers on Risk and Insurance: Issues and Practice*, vol. 47, no. 3, pp. 698–736, Jul. 2022, doi: 10.1057/s41288-022-00266-6.

[13] M. A. Hossain and M. S. Islam, "Enhanced detection of obfuscated malware in memory dumps: a machine learning approach for advanced cybersecurity," *Cybersecurity*, vol. 7, no. 1, Dec. 2024, doi: 10.1186/s42400-024-00205-z.

[14] Saputro Andik, Saputro Daniel Tunggono, and Remawat Dwi, "Implementasi Port Knocking Untuk Keamanan Jaringan Komputer Dengan Metode Demilitarized Zone," *Jurnal INFORMA Politeknik Indonusa Surakarta*, vol. 8, no. 2, 2022, doi: https://doi.org/10.46808/informa.v8i2.222.

[15] D. Lusi, Y. Suban Belutowe, U. I. Kupang Jl Perintis Kemerdekaan, K. Putih, and K. Kupang, "ANALISIS DAN IMPLEMENTASI DESAIN JARINGAN HOTSPOT BERBASIS MIKROTIK MENGGUNAKAN METODE NDLC (NETWORK DEVELOPMENT LIFE

 doi: https://doi.org/10.32503/jtecs.v4i2.5541

CYCLE) PADA KANTOR BALAI PELAKSANAAN JALAN NASIONAL NTT," *Jurnal Teknologi Informasi*, vol. 7, no. 1, 2023.