**KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET DAN TEKNOLOGI**

**INSTITUT TEKNOLOGI DAN BISNIS PALCOMTECH**

**SKRIPSI**

# PENERAPAN *PORT KNOCKING* DAN *HONEYPOT* PADA EKOSISTEM *SERVER FARM* DINAS KOMUNIKASI DAN INFORMATIKA KOTA PALEMBANG


**Diajukan Oleh :**

**WALFINDO BAYU SETYA**

**011190027P**

**Untuk Memenuhi Sebagian dari Syarat**

**Mencapai Gelar Sarjana Komputer**

**PALEMBANG**

**2023**

**KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET DAN TEKNOLOGI**

**INSTITUT TEKNOLOGI DAN BISNIS PALCOMTECH**

**SKRIPSI**

# PENERAPAN *PORT KNOCKING* DAN *HONEYPOT* PADA EKOSISTEM *SERVER FARM* DINAS KOMUNIKASI DAN INFORMATIKA KOTA PALEMBANG


**Diajukan Oleh :**

**WALFINDO BAYU SETYA**

**011190027P**

**Untuk Memenuhi Sebagian dari Syarat**
**Mencapai Gelar Sarjana Komputer**

**PALEMBANG**

**2023**

**KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET DAN TEKNOLOGI**

**INSTITUT TEKNOLOGI DAN BISNIS PALCOMTECH**

**HALAMAN PENGESAHAN PEMBIMBING SKRIPSI**

| | | |
|---|---|---|
| **NAMA** | **:** | **WALFINDO BAYU SETYA** |
| **NOMOR POKOK** | **:** | **011190027P** |
| **PROGRAM STUDI** | **:** | **S1 INFORMATIKA** |
| **JENJANG PENDIDIKAN** | **:** | **STRATA SATU (S1)** |
| **JUDUL** | **:** | **PENERAPAN *PORT KNOCKING* DAN *HONEYPOT* PADA EKOSISTEM *SERVER FARM* DINAS KOMUNIKASI DAN INFORMATIKA KOTA PALEMBANG** |

**Tanggal: 01 Maret 2023**
**Pembimbing**

**Guntoro Barovih, S.Kom., M.Kom.**
**NIDN: 0201048601**

**Mengetahui,**
**Rektor**

**Benedictus Effendi, S.T., M.T.**
**NIP: 09.PCT.13**

**KEMENTERIAN PENDIDIKAN, KEBUDAYAAN, RISET DAN TEKNOLOGI**

**INSTITUT TEKNOLOGI DAN BISNIS PALCOMTECH**

**HALAMAN PENGESAHAN PENGUJI SKRIPSI**

| | | |
|---|---|---|
| **NAMA** | **:** | **WALFINDO BAYU SETYA** |
| **NOMOR POKOK** | **:** | **011190027P** |
| **PROGRAM STUDI** | **:** | **S1 INFORMATIKA** |
| **JENJANG PENDIDIKAN** | **:** | **STRATA SATU (S1)** |
| **JUDUL** | **:** | **PENERAPAN *PORT KNOCKING* DAN *HONEYPOT* PADA EKOSISTEM *SERVER FARM* DINAS KOMUNIKASI DAN INFORMATIKA KOTA PALEMBANG** |

| | |
|---|---|
| **Tanggal: 01 Maret 2023** | **Tanggal: 01 Maret 2023** |
| **Penguji 1** | **Penguji 2** |
| **D Tri Octafian, S.Kom., M.Kom.** | **Hendra Effendi, S.Kom., M.Kom.** |
| **NIDN: 0213108002** | **NIDN: 0217108001** |

**Menyetujui,**

**Rektor**

**Benedictus Effendi, S.T., M.T.**
**NIP: 09.PCT.13**

**Motto:**

- Ikhlas, sabar dan ikhtiar adalah kunci yang harus kamu pegang untuk menuju kesuksesan.
- Jadikan sabar dan sholat sebagai penolongmu, dan sesungguhnya yang demikian itu sungguh berat, kecuali bagi orang-orang yang khusyu.
- Sesungguhnya sesudah kesulihan itu ada kemudahan. Maka apabila kamu telah selesai (dari suatu urusan), kerjakanlah dengan sungguh-sungguh (urusan yang lain).

(Walfindo Bayu Setya)

**Kupersembahkan kepada:**

- Orang tua tercinta yaitu Ibu, Ayah dan saudara/i yang selalu mendukung
- Kepada bapak Guntoro Barovih, S.Kom., M.Kom selaku dosen pembimbing yang selalu memberikan bimbingan serta arahan kepada peneliti sehingga skripsi ini dapat terselesaikan
- Kepada pihak Dinas Komunikasi dan Informatika Kota Palembang yang telah memberikan kesempatan bagi peneliti untuk dapat melangsungkan penelitian dan memperoleh data, terutama kepada bapak Eko Mulyadi S.T., M.T., Rianda Pratama S.Kom, dan Doni A.Md.Kom.
- Kepada Desti Natalia, Walid Badeges, Danil Reza Mahendra, Erizon Ade Pratama S.Kom para sahabat dan teman yang selalu memberikan semangat serta dukungan.

# KATA PENGANTAR

Dengan mengucapkan Alhamdulillah Puji dan syukur peneliti panjatkan atas kehadirat Allah Yang Maha Esa yang telah memberikan berkat dan rahmat nya dengan kelancaran menyelesaikan penulisan skripsi yang berjudul "**Penerapan *Port Knocking* dan *Honeypot* Pada Ekosistem *Server Farm* Dinas Komunikasi dan Informatika Kota Palembang''** ini dapat diselesaikan guna memenuhi salah satu persyaratan dalam menyelesaikan program studi S1 Informatika Institut Teknologi dan Bisnis PalcomTech Palembang.

Sebagai rasa syukur dan hormat, melalui kesempatan ini peneliti mengucapkan terimakasih banyak kepada semua pihak yang telah membantu, serta memberikan segala saran, motivasi dalam penulisan laporan skripsi ini. Untuk itu peneliti mengucapkan terimakasih kepada kedua orang tua saya tercinta, Kepada Rektor Institut Teknologi dan Bisnis PalcomTech Bapak Benedictus Effendi, S.T., MT. Kepada Dosen Pembimbing Bapak Guntoro Barovih, S.Kom., M.Kom. Kepada seluruh keluarga dan teman-teman seperjuangan, yang telah banyak membantu dan mendukung peneliti sehingga terselesaikan penulisan laporan skripsi.

Demikian kata pengantar dari peneliti, dengan harapan semoga skripsi ini berguna dan bermanfaat bagi semua pihak yang membutuhkan, dengan kesadaran peneliti bahwa penulisan skripsi masih mempunyai beberapa kekurangan dan kelemahan sehingga membutuhkan banyak saran dan kritik yang membangun untuk menghasilkan sesuatu yang lebih baik. Akhir kata, atas perhatinnya peneliti ucapkan terimakasih.

Palembang,  2023

Peneliti

## DAFTAR ISI

**HALAMAN JUDUL..... i**

**HALAMAN PENGESAHAN PEMBIMBING.....ii**

**HALAMAN PENGESAHAN PENGUJI..... iii**

**HALAMAN MOTTO DAN PERSEMBAHAN.....iv**

**KATA PENGANTAR..... v**

**DAFTAR ISI.....vii**

**DAFTAR GAMBAR..... xi**

**DAFTAR TABEL..... xii**

**DAFTAR LAMPIRAN..... xiii**

**ABSTRAK..... xiv**

**BAB I PENDAHULUAN**

1.1. Latar Belakang..... 1

1.2. Rumusan Masalah Penelitian..... 3

1.3. Ruang Lingkup Penelitian..... 3

1.4. Tujuan Penelitian..... 4

1.5. Manfaat Penelitian..... 4

1.5.1. Manfaat Penelitian Bagi Penulis..... 4

1.5.2. Manfaat Penelitian Bagi Dinas Komunikasi dan Informatika Kota Palembang..... 4

1.5.3. Manfaat Penelitian Bagi Akademik..... 4

1.6. Sistematika Penulisan..... 5

**BAB II GAMBARAN UMUM PERUSAHAAN**

2.1. Profile Perusahaan.....6

2.1.1. Sejarah Perusahaan.....6

2.1.2. Visi dan Misi.....7

2.1.3. Struktur Organisasi.....8

2.1.4. Tugas Wewenang.....9

**BAB III TINJAUAN PUSTAKA**

3.1. Teori Pendukung.....32

3.1.1. Keamanan Jaringan.....32

3.1.2. Sistem Operasi.....32

3.1.3. Linux.....33

3.1.4. *Secure shell (SSH)*.....34

3.1.5. *IP Address (Internet Protocol Address)*.....34

3.1.6. *Honeypot*.....35

3.1.7. *Port knocking*.....35

3.1.8. *Vulnerability Scanning*.....36

3.1.9. *Brute force*.....37

3.1.10. *IPTables*.....37

3.2. Hasil Penelitian Terdahulu.....38

3.3. Kerangka Pemikiran.....40

**BAB IV METODE PENELITIAN**

4.1. Lokasi dan Waktu Penelitian.....41

4.1.1. Lokasi.....41

4.1.2. Waktu Penelitian.....41

4.2. Teknik Pengumpulan Data.....42

4.2.1. Observasi.....42

4.2.2. Wawancara.....42

4.2.3. Studi Literatur.....42

4.3. Alat Pengembangan Sistem.....43

4.3.1. *Flowchart*.....43

4.4. Metode Pengembangan Sistem.....44

**BAB V HASIL DAN PEMBAHASAN**

5.1. Hasil dan Pembahasan.....48

5.1.1. *Analysis*.....48

5.1.2. *Design*.....57

5.1.3. *Simulation/Prototyping*.....58

5.1.4. *Implementation*.....59

5.1.5. *Monitoring*.....71

5.1.6. *Management*.....85

**BAB VI PENUTUP**

6.1. Kesimpulan.....86

6.2. Saran.....87

**DAFTAR PUSTAKA.....xvii**

**HALAMAN LAMPIRAN.....xix**

## DAFTAR GAMBAR

Gambar 2.1 Struktur Organisasi Dinas Komunikasi Dan Informatika Kota Palembang..... 8

Gambar 3.1 Kerangka Pemikiran..... 40

Gambar 4.1 Model *Network Development Life Cycle*.....45

Gambar 5.1 Topologi Sebelum Perancangan..... 49

Gambar 5.2 Pengujian *Port Scanning Server* Sebelum Implementasi *Honeypot*, *Port Knocking* dan *Iptables*.....50

Gambar 5.3 Proses Serangan *Brute Force* Pada *Port* 21 *FTP*..... 50

Gambar 5.4 Proses Serangan *Brute Force* Pada *Port* 22 *SSH*..... 51

Gambar 5.5 Serangan *DoS* Menggunakan *Torshammer*..... 52

Gambar 5.6 Proses Memanajemen Yang Sedang Berjalan Sebelum Penyerangan *DoS* Pada *Server*.....52

Gambar 5.7 Proses Memanajemen Yang Sedang Berjalan Sesudah Penyerangan *DoS* Pada *Server*.....53

Gambar 5.8 Proses Hasil *Capture Wireshark*..... 54

Gambar 5.9 *Flowchart* Perancangan..... 57

Gambar 5.10 Topologi Yang Diusulkan..... 58

Gambar 5.11 Bagan Simulasi Keamanan Jaringan..... 59

Gambar 5.12 Install *Port Knocking Server*..... 60

Gambar 5.13 Proses Konfigurasi *Port Knocking*..... 61

Gambar 5.14 Proses Konfigurasi *Honeypot*..... 63

Gambar 5.15 *Client* Melakukan *Port Scanning* Untuk Mengecek *Port* Yang Terbuka Pada *Server*..... 64
Gambar 5.16 *Client* Mencoba Untuk Mengakses *Port* 22 Pada *Server*..... 64
Gambar 5.17 Konfigurasi *Rules Iptables*..... 66
Gambar 5.18 Pengujian *Port Scanning Server* Setelah Melakukan Installasi dan Konfigurasi *Honeypot*..... 71
Gambar 5.19 Proses Penyerangan Pada *Port* 22 Menggunakan *Brute force*..... 73
Gambar 5.20 Proses Penyerangan Pada *Port* 2323 Menggunakan *Brute force* ..... 74
Gambar 5.21 Pengujian *Port Scanning Server* Setelah Melakukan Installasi dan Konfigurasi *Honeypot* dan *Port Knocking*..... 76
Gambar 5.22 Proses Akses *Port* 21 Yang Telah Terlindungi *Port Knocking* dan *Brute Force Attack* Pada *Port* 21 *FTP*..... 77
Gambar 5.23 Proses *Login SSH* Sebelum Melakukan *Port Knocking*..... 78
Gambar 5.24 Proses *Login SSH* Sesudah Melakukan *Port Knocking*..... 78
Gambar 5.25 Pengujian *Brute Force Attack* Pada *Port* 2323 *SSH* Yang Terlindungi *Port Knocking*..... 79
Gambar 5.26 Pengujian *Port Scanning Server* Setelah Melakukan Installasi dan Konfigurasi *Honeypot*, *Port Knocking* dan *Iptables*..... 81
Gambar 5.27 Penyerangan *DoS* Setelah Diterapkan *Iptables*..... 82
Gambar 5.28 Proses Memanajemen Yang Sedang Berjalan Sebelum Penyerangan *DoS* Pada *Server* Yang Terlindungi *Iptables*..... 83
Gambar 5.29 Proses Memanajemen Yang Sedang Berjalan Sesudah Penyerangan *DoS* Pada *Server* Yang Terlindungi *Iptables*..... 83

# DAFTAR TABEL

Tabel 3.1 Penelitian Terdahulu..... 39

Tabel 4.1 Waktu Penelitian..... 41

Tabel 4.2 Simbol Umum *Flowchart*..... 44

Tabel 5.1. Hasil Tabel Pengujian Sebelum Implementasi Keamanan *Honeypot*, *Port Knocking* dan *Iptables*..... 55

Tabel 5.2 Hasil Tabel Pengujian Sesudah Implementasi Keamanan *Honeypot*..... 75

Tabel 5.3 Hasil Pengujian *Honeypot* dan *Port knocking*..... 80

Tabel 5.4 Tabel Uji Coba Skema *Port Knocking*.....81

Tabel 5.5 Hasil Pengujian *Honeypot*, *Port knocking* dan *Iptables*.....84

# DAFTAR LAMPIRAN

Lampiran 1. *Form* Topik dan Judul (*Fotocopy*)

Lampiran 2. Surat Balasan dari Perusahaan (*Fotocopy*)

Lampiran 3. *Form* Konsultasi (*Fotocopy*)

Lampiran 4. Surat Pernyataan (*Fotocopy*)

Lampiran 5. *Form* Revisi Ujian Pra Sidang (*Fotocopy*)

Lampiran 6. *Form* Revisi Ujian Kompre (Asli)

Lampiran 7. *Listing Code*

# ABSTRAK

WALFINDO BAYU SETYA. Penerapan *Port Knocking* dan *Honeypot* Pada Ekosistem *Server Farm* Dinas Komunikasi dan Informatika Kota Palembang.

Keamanan jaringan komputer menjadi bagian terpenting dalam menjaga sebuah jaringan, serangan yang bisa mengganggu bahkan merusak sistem koneksi antar perangkat yang terhubung akan sangat merugikan, maka mengatasi hal tersebut, perlu dibangun sistem keamanan *server* dengan bertujuan mencegah serangan dari luar yang dapat menyebabkan kehilangan data. Salah satu cara yang bisa dilakukan adalah dengan membangun sistem keamanan *server* dengan melakukan penarapan *honeypot*, *port knocking* dan *iptables*. Metode dalam penelitian ini menggunakan metode *NDLC*. Pengujian dilakukan menggunakan metode *penetration testing* yang menerapkan pengujian dua kondisi, yaitu sebelum dan sesudah penerapan keamanan jaringan *server*. Hasil dari penelitian ini adalah penerapan mekanisme *honeypot* yang nantinya menjadi salah satu solusi yang dapat diberikan karena merupakan sebuah sistem umpan yang dapat digunakan untuk memancing penyerang dengan menyamarkan diri sebagai sistem yang rentan. Keamanan jaringan *server port knocking* berfungsi menyembunyikan layanan *port* pada *server* sampai urutan ketukan dilakukan untuk membuka dan menutup *port* tertentu yang diamankan oleh *administrator* dengan menggunakan beberapa *key* atau kode. Selanjutnya *iptables* yang berfungsi untuk melindungi *server* dari serangan dan melindungi *port* pada *server*.

**Kata Kunci :** ***Server, Honeypot, Port Knocking, Iptables.***

## *ABSTRACT*

*WALFINDO BAYU SETYA. Implementation Of Port Knocking And Honeypot In The Server Farm Ecosystem Of Communication And Informatics Department Of Palembang City.*

*Computer network security is the most important part in maintaining a network, attacks that can disrupt and even damage the connection system between connected devices will be very detrimental, so to overcome this, it is necessary to build a server security system with the aim of preventing attacks from outside that can cause data loss. One way that can be done is to build a server security system by implementing honeypot, port knocking and iptables applications. The method in this study uses the NDLC method. Testing is carried out using the penetration testing method which applies two test conditions, namely before and after implementing server network security. The result of this research is the application of a honeypot mechanism which will later become one of the solutions that can be provided because it is a decoy system that can be used to lure attackers by disguising themselves as vulnerable systems. Port knocking server network security functions to hide port services on the server until a sequence of knocks is performed to open and close certain ports that are secured by the administrator using several keys or codes. Furthermore, iptables functions to protect the server from attacks and protect ports on the server.*



***Keywords: Server, Honeypot, Port Knocking, Iptables.***

# BAB I

# PENDAHULUAN

## 1.1. Latar Belakang

Perkembangan teknologi informasi di era modern saat ini sangatlah cepat dan telah menjadi salah satu kebutuhan baik pada lingkup individu maupun organisasi, dengan pesatnya dunia teknologi yang serba digital membutuhkan infrastruktur jaringan yang memadai sebagai pendukung perkembangan dunia yang serba digital.

Jaringan komputer berkembang dengan sangat pesat, akses terhadap internet sangat dibutuhkan oleh semua kalangan sekarang ini. Internet tidak hanya diakses untuk mencari informasi bagi orang–orang yang membutuhkannya, tetapi juga diakses oleh *hacker* atau *cracker* maka dari itu kita harus mengamankan jaringan komputer untuk mencegah serta mengidentifikasi pengguna yang tidak legal (penyusup) berasal dari jaringan komputer. Tujuan keamanan jaringan komputer ialah untuk mengantisipasi resiko jaringan komputer berupa bentuk ancaman fisik maupun logik baik langsung ataupun tidak langsung mengganggu aktivitas yang sedang berlangsung pada jaringan komputer.

Sepanjang tahun 2022 ada lebih dari 700 juta serangan siber terjadi di Indonesia. Data Badan Siber dan Sandi Negara (BSSN) menyebut, total 714.170.967 anomali trafik atau serangan siber yang terjadi di sepanjang 2022, dengan angka serangan paling tinggi terjadi pada Januari dengan angka

serangan 272.962.734, lebih dari sepertiga total serangan selama semester pertama 2022.

Dinas Komunikasi dan Informatika Kota Palembang berperan sebagai penyelenggara urusan pemerintah bidang Komunikasi dan Informatika untuk daerah Kota Palembang. Kantor Dinas Komunikasi dan Informatika Kota Palembang memiliki beberapa bidang yang salah satunya adalah bidang *E-Government* yang terdapat pada bagian jaringan dan *server*.

Berbagai upaya Dinas Komunikasi dan Informatika Kota Palembang untuk menjaga keamanan data *server* dengan menggunakan *firewall* sebagai dinding penghalang pembatasan akses. Penggunaan *firewall* sendiri masih kurang efektif dikarenakan menutup semua akses tanpa memperdulikan siapapun yang sedang terkoneksi dalam jaringan.

Data dan informasi *server* hanya ditujukan untuk *administrator* dan *user* yang berhak mengakses melalui *port* layanan. Membiarkan *port* penting terbuka adalah kesalahan yang dapat mengakibatkan serangan terhadap *server*, umumnya teknik yang sering dilakukan adalah *scanning port* dan *brute force*. Dengan menggunakan *port knocking* sistem autentifikasi menggunakan kombinasi lapisan-lapisan kunci untuk dapat menggunakan *port* komunikasi yang dilindungi *port knocking*. Teknik ini mempertahankan suatu *port* dan hanya dapat terbuka jika menggunakan *sequence of request.*

Selain menggunakan metode *port knocking* dibutuhkannya *honeypot* yang dapat berjalan di *server* Dinas Komunikasi dan Informatika Kota Palembang yang nantinya dapat menyembunyikan *service port SSH* asli yang

biasa diakses dan diserang oleh penyerang dan juga membuat *service port SSH* palsu yang mampu menipu dan memantau penyerang yang mengancam pada *server*. Sehingga dengan *honeypot*, *server* Dinas Komunikasi dan Informatika Kota Palembang dapat meminimalisir dari serangan siber yang dilakukan oleh penyerang.

Berdasarkan latar belakang di atas maka penulis akan mengangkat sebuah judul "**Penerapan *Port Knocking* dan *Honeypot* Pada Ekosistem *Server Farm* Dinas Komunikasi dan Informatika Kota Palembang**".

## 1.2. Rumusan Masalah Penelitian

Rumusan masalah penelitian ini adalah "Penerapan *Port Knocking* dan *honeypot* Pada Ekosistem *Server Farm* Dinas Komunikasi dan Informatika Kota Palembang".

## 1.3. Ruang Lingkup Penelitian

Ruang lingkup penelitian ini adalah sebagai berikut :

1. Sistem Operasi yang digunakan adalah *Linux Ubuntu* yang diinstall pada S*erver Proxmox Virtual Environment*.
2. Mengamankan *server* dengan *port knocking* dan memanfaatkan *honeypot* sebagai *server* tiruan untuk mencegah terjadinya serangan ke *server* utama.
3. Pengujian keamanan jaringan *server honeypot* dan *port knocking* dengan menggunakan metode *Penetration Testing*.

## 1.4. Tujuan Penelitian

Tujuan dari penelitian ini untuk menerapkan keamanan jaringan *server* Dinas Komunikasi dan Informatika Kota Palembang menggunakan *port knocking* dan *honeypot*.

## 1.5. Manfaat Penelitian

Penelitian ini diharapkan bermanfaat bagi tempat penelitian, akademik, dan penelitian sendiri, meliputi :

### 1.5.1. Manfaat Penelitian Bagi Penulis

Penulis dapat menerapkan ilmu pengetahuan yang diperoleh selama waktu perkuliahan khususnya dalam bidang keamanan pada jaringan *server*.

### 1.5.2. Manfaat Penelitian Bagi Dinas Komunikasi dan Informatika Kota Palembang

Penelitian ini diharapkan dapat memberikan manfaat bagi Dinas Komunikasi dan Informatika Kota Palembang dalam keamanan jaringan *server* menggunakan *honeypot* dan *port knocking* dengan sumber daya yang terjangkau untuk keperluan pencegahan serangan siber.

### 1.5.3. Manfaat Penelitian Bagi Akademik

Referensi bagi penelitian selanjutnya dalam pembuatan laporan skripsi, khususnya mahasiswa Institut Teknologi dan Bisnis PalcomTech, dan dapat menjadi bahan perbandingan dalam penelitian bagi pihak yang berkepentingan dalam keamanan jaringan *server*.

## 1.6. Sistematik Penulisan

Penulis menggunakan pembahasan yang sesuai dengan ketentuan yang diberikan, sebagai berikut :

**BAB I PENDAHULUAN**

Bab ini berisi uraian latar belakang, rumusan masalah, batasan masalah, tujuan dan manfaat penelitian.

**BAB II GAMBARAN UMUM PERUSAHAAN**

Bab ini berisi tentang gambaran umum perusahaan.

**BAB III TINJAUAN PUSTAKA**

Bab ini berisi teori berdasarkan penulis yang terdiri dari landasan teori, penelitian terdahulu dan kerangka penelitian.

**BAB IV METODE PENELITIAN**

Bab ini berisi tentang waktu dan jenis penelitian, jenis dan pengumpulan data, pengembangan dan pengujian sistem.

**BAB V HASIL DAN PEMBAHASAN**

Bab ini memuat hasil yang diperoleh dalam penelitian dan pembahasan serta masalah yang telah ditemukan peneliti.

**BAB VI PENUTUP**

Bab ini berisi kesimpulan dan saran dari pembahasan dalam penelitian yang telah dilakukan

# BAB II

# GAMBARAN UMUM PERUSAHAAN

## 2.1. Profil Perusahaan

### 2.1.1. Sejarah Perusahaan

Sebuah kondisi yang kontradiktif terjadi setelah pemerintahan, memutuskan tidak lagi mencantumkan beberapa departemen dalam Kabinet Persatuan Nasional periode 1999-2004 yang diumumkan oleh Presiden RI pada 26 Oktober 1999 salah satunya adalah Departemen Penerangan RI. Pada tingkat pusat keberadaan Departemen Penerangan digantikan Badan Informasi dan Komunikasi Nasional yang ditetapkan berdasarkan keputusan Presiden RI Nomor 153 Tahun 1999 terhitung tahun 1999, sedangkan kelanjutan fungsi serta kelembagaan Departemen Penerangan di daerah dilimpahkan kepada pemerintah daerah. Pembubaran lembaga yang bertugas untuk mengkoordinasikan jalannya informasi dan komunikasi diindonesia.

Dalam menyikapi reformasi sistem pemerintah pusat maka di daerah pun dikeluarkan penyesuaian yang didasarkan pada undang-undang No. 22 tahun 1999 tentang pemerintah daerah, kecuali di bidang Keamanan, Keuangan, Agama, dan Hukum/Peradilan dari luar negeri. Oleh karena itu, pemerintah Kota Palembang dalam merekontruksikan struktur organisasi pemerintah daerah yang disesuaikan dengan tuntutan dan kebutuhan pembangunan maka

dikeluarkan Peraturan Daerah No. 2 Tahun 2001 tentang Pembentukan, Kedudukan Tugas Pokok, Fungsi, dan Struktur Organisasi Dinas termasuk Dinas Informasi dan Komunikasi.

Sejalan dengan dikeluarkan peraturan pemerintahan No. 38 Tahun 2007 tentang pembagian urusan pemerintahan antar pemerintah, pemerintah daerah/ provinsi dan pemerintah daerah kabupaten/kota. Untuk pemerintah kota Palembang hal ini ditindaklanjuti dengan Peraturan Daerah Kota Palembang No. 9 tahun 2008 tentang kedudukan tugas pokok, maka Dinas Informasi dan Komunikasi diubah namanya menjadi Dinas Komunikasi dan Informatika (DISKOMINFO) dengan susunan organisasi.

### 2.1.2. Visi dan Misi

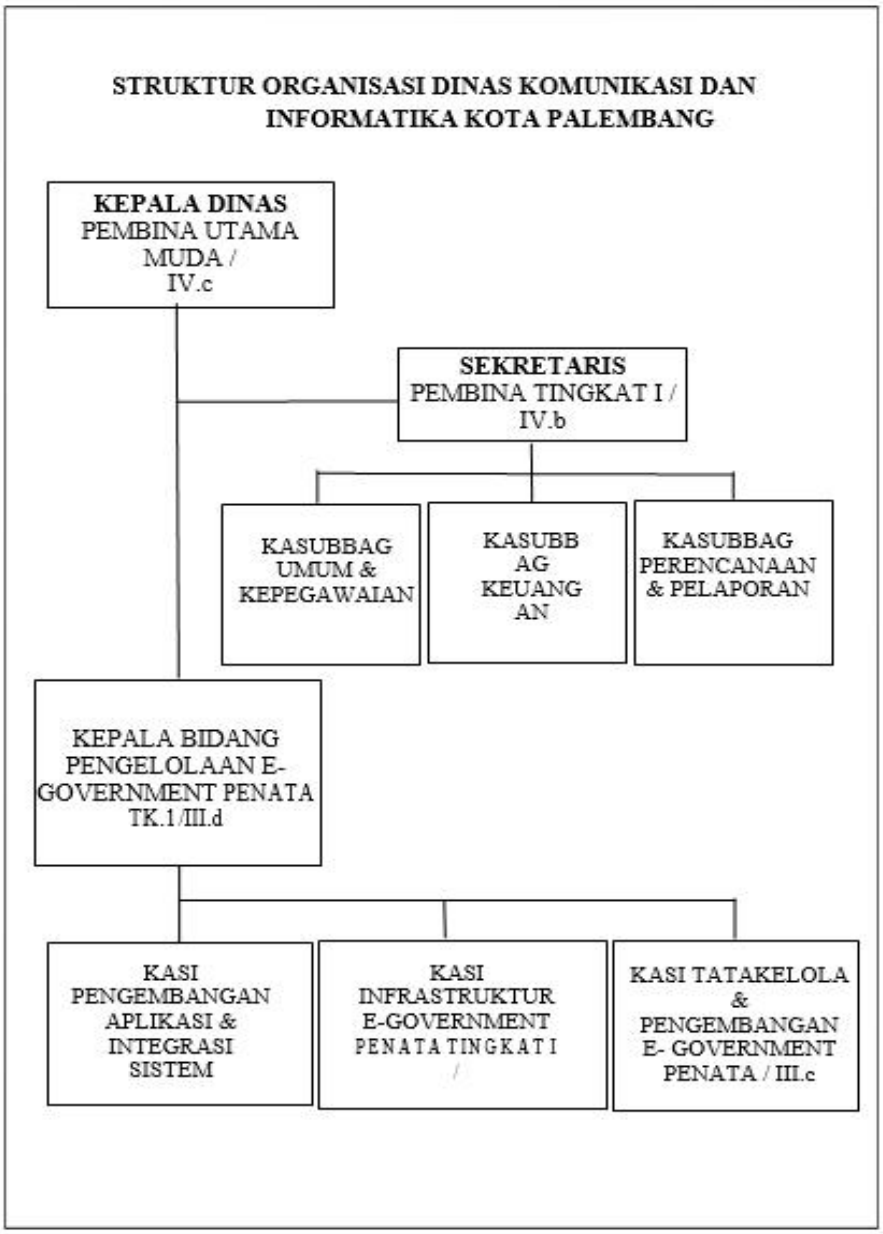
Visi : Adapun Visi Dinas Komunikasi dan Informatika (DISKOMINFO) Kota Palembang sebagai berikut "Terwujudnya Palembang yang Informatif, Maju dan Profesional yang berbasis teknologi dan mass media".

Misi :

a) Meningkatkan kualitas dan kuantitas peran mass media dalam mewujudkan pelayanan informasi publik.
b) Peningkatan SDM dan sarana dan prasarana komunikasi dan informatika yang handal.
c) Peningkatan, pemberdayaan masyarakat terhadap penggunaan TIK.

### 2.1.3. Struktur Organisasi

Struktur Organisasi merupakan kerangka susunan dan hubungan antara tiap bagian serta posisi yang ada pada suatu organisasi dalam menjalankan kegiatan operasional untuk mencapai tujuan yang




**Gambar 2.1 Struktur Organisasi Dinas Komunikasi Dan Informatika Kota Palembang**

### 2.1.4. Tugas Wewenang

**1. Kepala Dinas**

Kepala Dinas mempunyai tugas melaksanakan urusan pemerintahan bidang informatika berdasarkan ketentuan peraturan perundang-undangan yang berlaku dan petunjuk pelaksanaannya.

**2. Sekretaris**

Sekretaris mempunyai tugas membantu Kepala Dinas dalam mengkoordinasikan perencanaan, keuangan dan pelaporan serta menyelenggarakan urusan administrasi umum, perkantoran, kehumasan, dan kepegawaian. Sekretaris mempunyai fungsi sebagai berikut :

a) Koordinasi penyusunan dokumen perencanaan, keuangan, dan pelaporan.
b) Pelaksanaan urusan administrasi umum.
c) Pelaksanaan urusan rumah tangga, perlengkapan, dan perkantoran.
d) Pelaksanaan urusan administrasi kepegawaian.
e) Pelaksanaan urusan kehumasan.
f) Pelaksanaan fasilitas hukum dan perundang – undangan.

**A. Sub Bagian Umum dan Kepegawaian**

Sub Bagian Umum dan Kepegawaian, mempunyai tugas:

a) Menyusun rencana program dan kegiatan sub bagian umum dan kepegawaian.
b) Mengelola administrasi umum dan surat menyurat.
c) Mengelola kearsipan dan kepustakaan.
d) Mengelola administrasi barang, perlengkapan, kendaraan dinas.
e) Mengelola urusan rumah tangga, kehumasan dan keprotokolan dinas.
f) Mengelola administrasi kepegawaian dan perjalanan dinas.
g) Melaporkan hasil kerja dan capaian kinerja.
h) Melaksanakan tugas kedinasan lain sesuai dengan bidang tugasnya.

**B. Sub Bagian Keuangan**

a) Menyusun rencana program dan kegiatan sub bagian Keuangan.
b) Menyusun rencana anggaran kerja dinas
c) Menyusun rencana plafon kebutuhan anggaran dan penggunaan anggaran.
d) Mengelola administrasi keuangan belanja langsung dan belanja Tidak langsung.
e) Menyusun dan menganalisa laporan keuangan.

f) Mengontrol kegiatan perbendaharaan, verifikasi, dan pembukuan/akuntansi.

g) Melaporkan hasil kerja dan capaian kinerja.

h) Melaksanakan tugas kedinasan lain sesuai dengan bidang tugasnya.

**C. Sub Bagian Perencanaan dan Pelaporan**

a) Menyusun rencana program dan kegiatan dinas dan sub bagian perencanaan dan pelaporan.

b) Melaksanakan koordinasi penyusunan program dan kegiatan antar bidang.

c) Menyusun dokumen perencanaan dinas.

d) Mengukur capaian kinerja program dan kegiatan bidang.

e) *Monitoring* dan evaluasi capaian kinerja dinas.

f) Menyusun dokumen pelaporan dinas.

g) Melaporkan hasil kerja dan capaian kinerja.

h) Melaksanakan tugas kedinasan lain sesuai dengan bidang tugasnya.

i) Menyusun dokumen pelaporan dinas.

j) Melaporkan hasil kerja dan capaian kinerja.

k) Melaksanakan tugas kedinasan lain sesuai dengan bidang tugasnya.

**3. Bidang Pengelolaan Opini dan Pelayanan Informasi Publik**

Bidang Pengelolaan Opini dan Pelayanan Informasi Publik mempunyai tugas melaksanakan sebagian tugas dinas di bidang pengelolaan opini dan pelayanan informasi public. Bidang Pengelolaan Opini dan Pelayanan Informasi Publik mempunyai fungsi:

a) Penyiapan bahan perumusan kebijakan di bidang pengelolaan opini dan aspirasi publik di lingkup Pemerintah Kota, pengelolaan informasi untuk mendukung kebijakan nasional dan Pemerintah Kota, serta pelayanan informasi publik di Kota.

b) Penyiapan bahan pelaksanaan kebijakan di bidang pengelolaan opini dan aspirasi publik di lingkup Pemerintah Kota, pengelolaan informasi untuk mendukung kebijakan nasional dan Pemerintah Kota, serta pelayanan informasi publik di Kota.

c) Penyiapan bahan penyusunan norma, standar, prosedur, dan kriteria penyelenggaraan di bidang pengelolaan opini dan aspirasi publik di lingkup Pemerintah Kota, pengelolaan informasi untuk mendukung kebijakan nasional dan Pemerintah Kota, serta pelayanan informasi publik di Kota.

d) Penyiapan bahan pemberian bimbingan teknis dan supervisi di bidang pengelolaan opini dan aspirasi publik

di lingkup Pemerintah Kota, pengelolaan informasi untuk mendukung kebijakan nasional dan Pemerintah Kota, serta pelayananinformasi publik di Kota.

e) Pelaksanaan koordinasi dan kerjasama dengan instansi terkait.

f) Pelaksanaan *monitoring*, evaluasi dan pelaporan pelaksanaan tugas.

g) Pelaksanaan fungsi lain yang diberikan oleh KepalaDinas sesuai dengan tugas dan fungsinya.

**A. Seksi Pengelolaan Opini Publik**

Seksi Pengelolaan Opini Publik, melaksanakan tugas:

a) Menyusun rencana program dan kegiatan seksi pengelolaan opini publik.

b) Menyelenggarakan layanan *monitoring* isu publik di media (media massa dansosial).

c) Melakukan pengumpulan pendapat umum (survei, jajak pendapat).

d) Pembangunan citra pemerintah (pembuatan iklan layanan masyarakat melalui media cetak elektronik).

e) Penguatan opini publik melalui kerjasama kelompok dan tokoh masyarakat dan menjalin aktivitas bersama dengan instansi lain.

f) Pembentukan opini publik melalui pembuatan dan penyebarluasan informasi melalui stiker, brosur, spanduk, banner, baliho, videotron atau sejenis dan turunannya.

g) Mengolah aduan masyarakat di Kota.

h) Melaksanakan program dan petunjuk teknis pada pengelolaan opini publik.

i) Melaksanakan pengawasan, pembinaan dan pengendalian pada pengelolaan opini publik.

j) Melaksanakan koordinasi dan kerjasama dengan lembaga/instansi lain berkairan dengan pengelolaan opini publik.

k) Melaporkan hasil kerja dan capaian kinerja.

l) Melaksanakan tugas kedinasan lain sesuai dengan bidang tugasnya.

**B. Seksi Pengelolaan Informasi Publik**

Seksi Pengelolaan Informasi Publik, mempunyai tugas:

a) Menyusun rencana program dan kegiatan seksi pengelolaan informasi publik.

b) Melaksanakan pencarian dan pengumpulan data dan informasi, baik sektoral maupun lintas sektoral untuk kepentingan publik dan Pemerintah Kota.

c) Mengolah dan menganalisis data informasi untuk mendukung komunikasi publik lintas sektoral lingkup nasional dan daerah di Kota.

d) Melaksanakan penyiapan dan penyajian data dan informasi sektoral maupun lintas sektoral berupa cetakan, CD, Video, Film, Buku, Dokumentasi cetak dan elektronik serta bahan materi paparan selayang pandang Pemerintah Kota.

e) Melaksanakan pengawasan, pembinaan, dan pengendalian pada pengelolaan informasi publik.

f) Mengelola pengaduan masyarakat melalui pelayanan call center.

g) Melaporkan hasil kerja dan capaian kinerja.

h) Melaksanakan tugas kedinasan lain sesuai dengan bidang tugasnya.

**C. Seksi Layanan Informasi Publik**

Seksi Layanan Informasi Publik, mempunyai tugas:

a) Menyusun rencana program dan kegiatan seksi layanan informasi publik.

b) Menyelenggarakan layanan pengelolaan informasi publik untuk implementasi Undang-Undang Nomor 14 Tahun 2008 tentang Keterbukaan Informasi Publik.

c) Memberikan pelayanan informasi publik untuk implementasi UndangUndang Nomor 14 Tahun 2008 tentang Keterbukaan Informasi Publik, dan Layanan Pengaduan Masyarakat di Kota.

d) Menyelenggarakan sosialiasi dan pemeringkatan Keterbukaan Informasi Publik pada Badan Publik lingkup Pemerintah Kota guna mengetahui pelaksanaan keterbukaan informasi di setiap Badan Publik.

e) Melaksanakan penyiapan ketersediaan informasi publik yang terbarukan sesuai dengan jenis data informasi publik (data wajib disediakan, data setiap saat, data serta merta dan data yang dikecualikan, baik secara konveksional dan ataupun inkonveksional (modern dengan memanfaatkan Teknologi Informasi terbarukan).

f) Menyelenggarakan penyiapan sumber daya pelayanan informasi publik pada Badan Publik dalam jajaran Pemerintah Kota.

g) Melakukan penyelesaian sengketa informasi pada Badan Publik dalam jajaranPemerintah Kota.

h) Menyelenggarakan pelayanan pengaduan masyarakat.

i) Melaksanakan program dan petunjuk teknis pada layanan informasi publik.

j) Melaksanakan pengawasan, pembinaan, dan pengendalian pada layanan informasi publik.

k) Melaporkan hasil kerja dan capaian kinerja.

l) Melaksanakan tugas kedinasan lain sesuai dengan bidang tugasnya.

**4. Bidang Pengelolaan Komunikasi Publik**

Bidang Pengelolaan Komunikasi Publik mempunyai tugas melaksanakan sebagian tugas dinas di bidang pengelolaan komunikasi publik. Bidang Pengelolaan Komunikasi Publik mempunyai fungsi :

a) Penyiapan bahan perumusan kebijakan di bidang penyediaan konten lintas sektoral dan pengelolaan media komunikasi publik, layanan hubungan media dan penguatan kapasitas sumber daya komunikasi publik dan penyediaan akses informasidiKota.

b) Penyiapan bahan pelaksanaan kebijakan di bidang penyediaan konten lintas sektoral dan pengelolaan media komunikasi publik, layanan hubungan media dan penguatan kapasitas sumber daya komunikasi publik dan penyediaan akses informasi diKota.

c) Penyiapan bahan penyusunan norma, standar, prosedur, dan kriteria penyelenggaraan di bidang penyediaan konten lintas sektoral dan pengelolaan media komunikasi publik, layanan hubungan media dan penguatan kapasitas sumber daya komunikasi publik dan penyediaan akses informasi di Kota.

d) Penyiapan bahan pemberian bimbingan teknis dan supervisi di bidang penyediaan konten lintas sektoral dan pengelolaan media komunikasi publik, layanan hubungan media dan penguatan kapasitas sumber daya komunikasi publik dan penyediaan akses informasi di Kota.

e) Pemantauan, evaluasi, dan pelaporan di bidang penyediaan konten lintas sektoral dan pengelolaan media komunikasi publik, layanan hubungan media dan penguatan kapasitas sumber daya komunikasi publik dan penyediaan akses informasidi Kota.

f) Pelaksanaan koordinasi dan kerjasama dengan instansi terkait.

g) Pelaksanaan *monitoring*, evaluasi, dan pelaporan pelaksanaan tugas.

h) Pelaksanaan fungsi lain yang diberikan oleh Kepala Dinas sesuai dengan tugas dan fungsinya.

**A. Seksi Pengelolaan Media Komunikasi Publik**

Seksi Pengelolaan Media Komunikasi Publik mempunyai tugas:

a) Menyusun rencana program dan kegiatan seksi pengelolaan media komunikasi publik.
b) Menyelenggarakan layanan perencanaan komunikasi publik dan citra positif Pemerintah Kota.
c) Mengemas ulang konten nasional menjadi konten lokal.
d) Membuat konten lokal.mengelola saluran komunikasi milik Pemerintah Kota/ media internal, diseminasi informasi kebijakan melalui media Pemerintah Kota dan non Pemerintah Kota.
e) Mengkoordinasikan dan melaksanakan talk show, dialog, interaktif, jumpapers.
f) Menyelenggarakan expo, pameran, pekan informasi dan menyiapkan keikut sertaan dalam pameran.
g) Melaporkan hasil kerja dan capaian kinerja.
h) Melaksanakan tugas kedinasan lain sesuai dengan bidang tugasnya

**B. Seksi Pelayanan Media Center dan Hubungan Media**

Seksi Pelayanan Media Center dan Hubungan Media mempunyai tugas :

a) Menyusun rencana program dan kegiatan seksi pelayanan media center dan hubungan media.
b) Menyelenggarakan layanan pengelolaan hubungan dengan media (mediarelations).
c) Menyediakan bahan komunikasi bagi pimpinan Kota (briefing notes, press release, backgrounders) di Kota.
d) Menyiapkan dan menyajikan informasi melalui media audio visual, online, dan cetakan.
e) Menyediakan layanan informasi melalui akses internet.
f) Melaporkan hasil kerja dan capaiankinerja.
g) Melaksanakan tugas kedinasan lain sesuai dengan bidang tugasnya.

**C. Seksi Sumber Daya Komunikasi Publik**

Seksi Sumberdaya Komunikasi Publik mempunyai tugas :

a) Menyusun rencana program dan kegiatan seksi sumber daya komunikasi publik.
b) Menyelenggarakan layanan pemberdayaan dan penyediaan akses informasi bagi media dan lembaga komunikasi publik.
c) Mengembangkan sumber daya komunikasi publik di Kota.
d) Merencanakan pemantauan tingkat produktivitas.
e) Melaporkan hasil kerja dan capaian kinerja.

f) Melaksanakan tugas kedinasan lain sesuai dengan bidang tugasnya.

**5. Bidang Teknologi Informasi dan Persandian**

Bidang mempunyai tugas melaksanakan sebagian tugas dinas di bidang teknologi informasi dan persandian. Bidang Teknologi Informasi dan Persandian mempunyai fungsi:

a) Penyiapan bahan perumusan kebijakan di bidang layanan infrastruktur dasar data center, disaster *recovery center* dan TIK, layanan pengembangan intranet dan penggunaan akses internet, layanan manajemen data dan informasi *E-Government*, integrasi layanan publik dan kepemerintahan, layanan keamanan informasi *E-Government*, layanan sistem komunikasi intra Pemerintah Kota.

b) Penyiapan bahan pelaksanaan kebijakan di bidang layanan infrastruktur dasar data *center*, *disaster recovery center* dan TIK, layanan pengembangan intranet dan penggunaan akses internet, layanan manajemen data dan informasi *E-Government*, integrasi layanan publik dan kepemerintahan, layanan keamanan informasi *E-Government*, layanan sistem komunikasi intra Pemerintah Kota

c) Penyiapan bahan norma, standar, prosedur, dan kriteria penyelenggaraan di bidang layanan infrastruktur dasar data *center*, *disaster recovery center* dan TIK, layanan

pengembangan intranet dan penggunaan akses internet, layanan manajemen data dan informasi *E-Government*, integrasi layanan publik dan kepemerintahan, layanan keamanan informasi *E-Government*, layanan sistem komunikasi intra Pemerintah Kota.

d) Penyiapan bahan pemberian bimbingan teknis dan supervisi di bidang layanan infrastruktur dasar data center, *disaster recovery center* dan TIK, layanan pengembangan internet dan penggunaan akses internet, layanan manajemen data dan informasi *E-Government*, integrasi layanan publik dan kepemerintahan, layanan keamanan informasi *E-Government*, layanan sistem komunikasi intra Pemerintah Kota.

e) Pemantauan, evaluasi, dan pelaporan di bidang layanan infrastruktur dasar data *center*, disaster *recovery center* dan TIK, layanan pengembangan intranet dan penggunaan akses internet, layanan manajemen, data dan informasi *E-Government*, integrasi layanan publik dan kepemerintahan, layanan keamanan informasi *E-Government*, layanan sistem komunikasi intra Pemerintah Kota.

f) Pelaksanaan koordinasi dan kerjasama dengan instansi terkait.

g) Pelaksanaan *monitoring*, evaluasi, dan pelaporan pelaksanaan tugas.

h) Pelaksanaan fungsi lain yang diberikan oleh Kepala Dinas sesuai dengan tugas dan fungsinya

**A. Seksi Teknologi dan Komunikasi**

Seksi teknoloi dan komunikasi mempunyai tugas :

a) Menyusun rencana program dan kegiatan seksi infrastruktur teknologi dan telekomunikasi.

b) Melaksanakan pendataan operator seluler.

c) Menata, memberi rekomendasi, mengawasi, pembinaan dan penataan pengendalian menara telekomunikasi.

d) Menghubungkan layanan kerjasama pemerintah dan provider telekomunikasi.

e) Melaporkan hasil kerja dan capaian kinerja.

f) Melaksanakan tugas kedinasan lain sesuai dengan bidang tugasnya.

**B. Seksi Pengelolaan Data dan Statistik**

Seksi Pengelolaan Data dan Statistik mempunyai tugas:

a) Menyusun rencana program dan kegiatan seksi pengelolaan data dan statistik.

b) Menyelenggarakan layanan penetapan standar format data dan informasi, walidata, dan kebijakan.

c) Menyelenggarakan layanan *recovery* data dan informasi, layanan pengelolaan data elektronik pemerintahan dan non pemerintahan.

d) Menyelenggarakan layanan peningkatan kapasitas Sumber Daya Manusia dalam pemanfaatan Sistem Informasi Pemerintahan dan Sistem Informasi Publik.

e) Menyelenggarakan layanan interoperabilitas, layanan interkonektivitas layanan publik dan kepemerintahan, layanan pusat *Application Programming Interface* Kota

f) Melaporkan hasil kerja dan capaian kinerja.

g) Melaksanakan tugas kedinasan lain sesuai dengan bidang tugasnya.

**C. Seksi Persandian dan Keamanan Informasi**

Seksi Persandian dan Keamanan Informasi mempunyai tugas:

a) Menyusun rencana program dan kegiatan seksi persandiandan keamanan informasi.

b) Menyelenggarakan layanan *monitoring* trafik elektronik.

c) Menyelenggarakan layanan penanganan insiden keamanan informasi, layanan peningkatan kapasitas Sumber Daya Manusia di bidang keamanan informasi.

d) Menyelenggarakan layanan keamanan informasi pada Sistem Elektronik Pemerintah Kota dan audit TIK.
e) Menyelenggarakan internet sehat, kreatif, inovatif, dan produktif serta layanan penyediaan prasarana dan sarana komunikasi pemerintah.
f) Menyelenggarakan layanan bimbingan teknis dalam pemanfaatan sistem komunikasi oleh aparatur pemerintahan.
g) Menyusun perancanaan audit keamanan teknologi informasi.
h) Melaksanakaninternal audit kemanan teknologi informasi.
i) Melakukan identifikasi penyebab gangguan operasional keamanan informasi.
j) Memberikan pemahaman tentang kesadaran keamanan teknologi informasi kepada pengguna.
k) Melakukan instalasi aplikasi komunikasi terpadu (*server*,*client*,peralatan komunikasi).
l) Memonitor kinerja pengamanan jaringan komunikasi sandi.
m) Melakukan pembatasan hak akses terhadap pengelolaan.

n) Melakukan pengkinian (updating) aplikasi komunikasi terpadu.

o) Melakukan pemeliharaan peralatan sandi atau alat pendukung utama.

p) Melakukan pemantauan perkembangan keamanan informasi.

q) Melaporkan hasil kerja dan capaian kinerja.

r) Melaksanakan tugas kedinasan lain sesuai dengan bidang tugasnya

**6. Bidang Pengelolaan *E-Government***

Bidang Pengelolaan *E-Government* mempunyai tugas melaksanakan sebagian tugas dinas di bidang pengelolaan *E-Government*. Bidang Pengelolaan *E- Government* mempunyai fungsi:

a) Penyiapan bahan perumusan kebijakan di bidang Layanan Pengembangan dan Pengelolaan Aplikasi Generik, Spesifik, dan Suplemen yang terintegrasi, Penyelenggaraan Ekosistem TIK *Smart City,* Layanan nama domain dan sub domain bagi lembaga, pelayanan publik dan kegiatan, Penyelenggaraan *Government Chief Information Officer* di Pemerintah Kota dan masyarakat Kota.

b) Penyiapan bahan pelaksanaan kebijakan di bidang Layanan Pengembangan dan Pengelolaan Aplikasi Generik, Spesifik,

dan Suplemen yang terintegrasi, Penyelenggaraan Ekosistem *TIK Smart City*, Layanan nama domain dan sub domain bagi lembaga, pelayanan public dan kegiatan, Penyelenggaraan *Government Chief Information Officer* di Pemerintah Kota dan masyarakat Kota.

c) Penyiapan bahan penyusunan norma, standar, prosedur, dan kriteria penyelenggaraan di bidang Layanan Pengembangan dan Pengelolaan Aplikasi Generik, Spesifik, dan Suplemen yang terintegrasi, Penyelenggaraan Ekosistem TIK *Smart City*, Layanan nama domain dan sub domain bagi lembaga, pelayanan publik dan kegiatan, Penyelenggaraan *Government Chief Information Officer* di Pemerintah Kota dan masyarakat Kota

d) Penyiapan bahan pemberian bimbinga teknis dan supervisi di bidang Layanan Pengembangan dan Pengelolaan Aplikasi Generik, Spesifik, dan Suplemen yang terintegrasi, Penyelenggaraan Ekosistem TIK *Smart City*, Layanan nama domain dan sub domain bagi lembaga, pelayanan publik dan kegiatan, Penyelenggaraan *Government Chief Information Officer* di Pemerintah Kota dan masyarakat Kota.

e) Pemantauan, evaluasi, dan pelaporan di bidang Layanan Pengembangan dan Pengelolaan Aplikasi Generik, Spesifik,

dan Suplemen yang terintegrasi, Penyelenggaraan Ekosistem TIK *Smart City,* Layanan nama domain dan sub domain bagi lembaga, pelayanan publik dan kegiatan, Penyelenggaraan Government Chief Information Officer di Pemerintah Kota dan masyarakat Kota

f) Pelaksanaan koordinasi dan kerjasama dengan instansi terkait.

g) Pelaksanaan *monitoring*, evaluasi, dan pelaporan pelaksanaan tugas.

h) Pelaksanaan fungsi lain yang diberikan oleh Kepala Dinas sesuai dengan tugas dan fungsinya

**A. Seksi Pengembangan Aplikasi dan Integrasi Sistem Informasi**

Seksi Pengembangan Aplikasi dan Integrasi Sistem Informasi mempunyai tugas:

a) Menyusun rencana program dan kegiatan seksi pengembangan aplikasi dan integrasi sistem informasi.

b) Menyelenggarakan layanan pengembangan aplikasi pemerintahan dan pelayanan publik yang terintegrasi.

c) Menyelenggarakan layanan pemeliharaan aplikasi ke pemerintahan dan publik.

d) Melaksanakan inter konektivitas layanan publik dan kepemerintahan.

e) Melaksanakan layanan pusat *Application Program Interface (API)* Kota.

f) Melaksanakan layananan pengembangan *Business Proccess Rengineering* pelayanan di lingkungan pemerintah dan non pemerintah (*stakeholder smart city*).

g) Mengolah layanan sistem informasi *smart city.*

h) Melaporkan hasil kerja dan capaian kinerja.

i) Melaksanakan tugas kedinasan lain sesuai dengan bidang tugasnya

**B. Seksi Infrastruktur *E-Government***

Seksi Infrastruktur *E-Government* mempunyai tugas:

a) Menyusun rencana program dan kegiatan seksi infrastruktur *E- Government.*

b) Menyelenggarakan layanan infrastruktur dasar data *center*.

c) Menyelenggarakan layanan *disaster recovery center* dan TIK Pemeritah Kota.

d) Menyelenggarakan layanan pengembangan intranet dan penggunaan akses internet.

e) Menyelenggarakan layanan manajemen data informasi *E-Government*, integrasi layanan publik dan kepemerintahan.

f) Menyelenggarakan layanan keamanan informasi *E-Government*.

g) Menyelenggarakan layanan sistem komunikasi intra Pemerintah Kota.

h) Melaporkan hasil kerja dan capaian kinerja. Melaksanakan tugas kedinasan lain sesuai dengan bidang tugasnya

**C. Seksi Tata Kelola Pengembangan *E-Government***

Seksi Tata Kelola Pengembangan *E-Government* mempunyai tugas:

a) Menyusun rencana program dan kegiatan seksi tata kelola dan pengembangan *E- Government.*

b) Menyelenggarakan layanan penetapan regulasi dan kebijakan terpadu implementasi *E- Government* Kota.

c) Menyelenggarakan layanan koordinasi kerjasama lintas Organisasi Perangkat Daerah, lintas Pemerintah Kota dan lintas Pemerintah Pusat serta non pemerintah.

d) Menyelenggarakan layanan integrasi pengelolaan TIK dan *E-Government* Pemerintah Kota.

e) Menyelenggarakan layanan peningkatan kapasitas aparatur dan sertifikasi teknis bidang TIK, layanan peningkatan kapasitas masyarakat dalam implementasi *E-Government* dan *Smart City.*

f) Menyelenggarakan layanan implementasi *E-Government* dan *Smart City* dan Promosi pemanfaatan layanan *SmartCity.*

g) Melaporkan hasil kerja dan capaian kinerja.

h) Melaksanakan tugas kedinasan lain sesuai dengan bidang tugasnya.

# BAB III

# TINJAUAN PUSTAKA

## 3.1. Teori Pendukung

### 3.1.1. Keamanan Jaringan

Menurut Purba dan Efendi (2021) keamanan jaringan adalah konsep untuk mencegah pengguna yang tidak sah masuk ke dalam sistem jaringan komputer. Sistem harus tetap dilindungi dari segala macam serangan dan usaha penyusupan atau pemindaian oleh pihak yang tidak memiliki hak. Langkah-langkah pencegahan dapat membantu *administrator* untuk menghentikan pengguna yang tidak sah untuk mengakses sistem jaringan komputer.

Keamanan jaringan komputer berfungsi untuk mengantisipasi resiko-resiko yang akan terjadi pada jaringan komputer yang dapat menggangu aktivitas yang sedang terjadi pada sistem jaringan komputer. Ada tiga hal dalam konsep keamanan jaringan, yaitu tingkat bahaya, ancaman dan kerapuhan sistem.

### 3.1.2. Sistem Operasi

Menurut Muslim (2020) Sistem operasi komputer merupakan perangkat lunak komputer atau *software* yang bertugas melakukan kontrol dan manajemen perangkat keras dan juga operasi dasar sistem, termasuk menjalankan *software* aplikasi seperti program pengolah data yang digunakan mempermudah kegiatan manusia.

Sistem Operasi dalam bahasa Inggris disebut *operating system*, atau biasa disingkat *OS*. Sistem Operasi komputer merupakan *software* pada lapisan pertama diletakkan pada memori komputer (*Hardisk*, bukan memori *ram*) pada saat komputer dinyalakan. Sedangkan *software* lain dijalankan setelah Sistem Operasi Komputer berjalan dan Sistem Operasi melakukan layanan inti umum untuk *software* itu.

### 3.1.3. *Linux*

Menurut Dwiyatno (2020) *Linux* adalah nama yang diberikan kepada sistem operasi komputer bertipe *Unix*. *Linux* merupakan salah satu contoh hasil pengembangan perangkat lunak bebas dan sumber terbuka utama. Seperti perangkat lunak bebas dan sumber terbuka lainnya pada umumnya, kode sumber *Linux* dapat dimodifikasi, digunakan dan didistribusikan kembali secara bebas oleh siapa saja. Nama *"Linux"* berasal dari nama pembuatnya, yang diperkenalkan tahun 1991 oleh *Linus Torvalds*.

Sistemnya, peralatan sistem dan pustakanya umumnya berasal dari sistem operasi *GNU*, yang diumumkan tahun 1983 oleh *Richard Stallman*. Kontribusi *GNU* adalah dasar dari munculnya nama alternative *GNU/Linux*. *Linux* adalah sistem operasi berbasis *GNU/Linux* yang bersifat *Open Source* dan memiliki banyak varian seperti *Debian*, *Slackware*, *Open Suse*, *Arch linux*, *Redhat* dan sebagainya.

### 3.1.4. *Secure shell (SSH)*

Menurut Tohirin (2020) *SSH* atau *secure shell* merupakan protokol jaringan yang terdapat pada lapisan aplikasi pada protokol *model OSI* dan *TCP/IP*. *SSH* berjalan pada *port* standar *TCP* nomor 22. *SSH* difungsikan untuk memfasilitasi sistem komunikasi aman antara dua sistem yang memakai arsitektur klien *server*. Meski demikian, internet tidak sepenuhnya aman. *SSH* menyediakan integritas dan kerahasiaan data melewati teknik enkripsi dan dekripsi dilakukan secara otomatis pada koneksinya.

*SSH* digunakan untuk melakukan *remote* sistem sehingga *administrator* tidak perlu masuk secara langsung, melakukan, *forwarding port TCP*, *tunneling* dan koneksi X11. *SSH* paling banyak dipraktikkan oleh pengguna sistem operasi *Unix* dan *Linux*. Sebagai autentikasi komputer atau *server remote*, digunakan kriptografi jenis kunci publik pada *SSH*. Koneksi dengan *daemon SSH* harus terjadi terlebih dahulu *remote server* dapat dilakukan.

### 3.1.5. *IP Address (Internet Protocol Address)*

Menurut Rahayu Nugraheni Rachmawati (2020) *IP address* atau alamat *IP* yang bahasa awamnya biasa disebut dengan kode pengenal komputer pada jaringan merupakan komponen vital pada *internet*, karena tanpa alamat *IP* seseorang tidak akan dapat terhubung dengan *internet*. Setiap komputer yang terhubung dengan *internet* setidaknya harus memiliki satu buah alamat *IP*

pada setiap perangkat yang terhubung ke *internet* dan alamat *IP* itu sendiri harus unik karena tidak boleh ada *computer* jaringan lainnya yang menggunakan alamat *IP* yang sama di *internet*.

### 3.1.6. *Honeypot*

Menurut Mardiansyah (2021) *honeypot* merupakan sebuah sistem yang dibangun menyerupai atau persis dengan sistem yang sesungguhnya, dengan tujuan agar para *attacker* teralih perhatiannya dari sistem utama yang akan di serang, dan beralih menyerang ke sistem palsu tersebut. Saat ini *honeypot* tidak hanya berfungsi atau bertujuan untuk menjebak *attacker* untuk melakukan serangan ke *server* asli, namun *honeypot* juga bermanfaat untuk para system *administrator* atau *security analyst*, untuk menganalisa aktifitas apa saja yang dilakukan oleh *cracker* yang terdapat di dalam sistem *honeypot* tersebut.

### 3.1.7. *Port knocking*

Menurut Ernawati (2022) *port knocking* merupakan konsep penting untuk mengamankan layanan yang disediakan *server*. kegunaan untuk membuka akses ke *port* tertentu yang telah ditutup *firewall*, dengan cara mengirimkan paket atau koneksi tertentu yang bisa berupa protokol *TCP*, *UDP* maupun *ICMP*. koneksi yang dikirim oleh *host* sudah sesuai dengan *rule knocking* yang diterapkan, maka secara otomatis *firewall* akan memberikan akses *port* yang sudah diblok.

*Port knocking* adalah teknologi baru yang menjanjikan untuk lebih mengamankan layanan jarak jauh. Teknologi ini dapat digunakan untuk menjaga semua *port*, seperti *port* yang ditutup sampai pengguna mengautentikasi dengan urutan *knock port*. Semua *port* tetap tertutup sehingga membuat *server* tidak terlihat oleh pemindai *port*. Cara kerja *port knocking* yang berfungsi untuk mengamankan, dengan cara mengetuk terlebih dahulu untuk bisa masuk pada *server* yang diamankan.

### 3.1.8. *Vulnerability Scanning*

Menurut Alwi (2020) tahapan dilakukan *scanning network* dengan memanfaatkan berbagai *tools network scanning* dan *vulnerability scanner online*. tujuan yang ingin dicapai adalah memperoleh informasi *vulnerability network* tersebut, misal daftar *port* yang terbuka, *bug* pada aplikasi *server*, dan lain-lain yang kadangkala fase ini disebut sebagai *passive attack*.

*Port scanning* merupakan proses memeriksa *service* yang berjalan pada target komputer dengan mengirimkan *sequence of messages*. *Port scanning* melakukan *probing TCP* dan *UDP ports* untuk mengetahui apakah suatu *service* berjalan atau dalam *listening state*. *Listening states* akan memberikan informasi mengenai *OS* dan aplikasi yang digunakan.

### 3.1.9. *Brute force*

Menurut Dawamsyach (2023) *Brute force* adalah teknik serangan terhadap sebuah sistem keamanan dengan menebak atau mencari semua kemungkinan *password* dengan masukan karakter dan panjang *password* tertentu yang tentunya dengan sangat banyak kombinasi *password*. Proses dapat menggunakan menggunakan *software* dan juga cara lain untuk melakukan serangan *brute force*. Beberapa aplikasi yang terdapat dalam sistem operasi *kali linux* biasa digunakan oleh *hacker* untuk melakukan serangan *brute force*. *Brute force* dilakukan dengan cara menebak atau melakukan percobaan terhadap semua kunci *password* dengan harapan dapat memperoleh akses sebuah sistem.

### 3.1.10. *Iptables*

Menurut Santoso (2022) *Iptables* adalah *firewall open source* yang tersedia pada *linux* versi 2.4 atau yang lebih baru. *Iptables* didasarkan pada Modul *Netfilter* yang dapat memeriksa isi dari paket untuk *string* tertentu dengan menggunakan dukungan pencocokan *string* pada *kernel linux*. *Iptables* dapat mendeteksi serangan lapisan aplikasi.

*Iptables* mempunyai 3 *primary tables*, yaitu:

1. *Filter table* : menyaring paket lalu menentukan apakah paket tersebut di *ACCEPT* atau di *DROP*.

2. *Nat table* : berisi *Network Address Translations* dan *demilitarized zone*.
3. *Mangle table* : memeriksa paket dengan parameter *TOS*, *TTL* , dan lain-lain.

Terdapat 3 jenis chains default, yaitu :

1. *INPUT* : rantai ini menangani semua paket yang masuk ke dalam *server*
2. *OUTPUT* : rantai ini berisi aturan untuk lalu lintas yang dibuat oleh *server*
3. *FORWARD* : rantai ini untuk menangani lalu lintas yang dikirimkan ke *server* lain dan yang tidak dibuat di *server* utama.

Ketika *pattern* dari paket sama dengan *rules* yang tersedia maka akan dilakukan action yaitu:

1. *ACCEPT* : mengizinkan paket untuk masuk ke dalam *server*.
2. *DROP* : membuang paket tersebut.
3. *RETURN* : menghentikan paket agar tidak melintasi rantai *(chains)* dan mengirimkan paket kembali ke rantai sebelumnya.

## 3.2. Penelitian Terdahulu

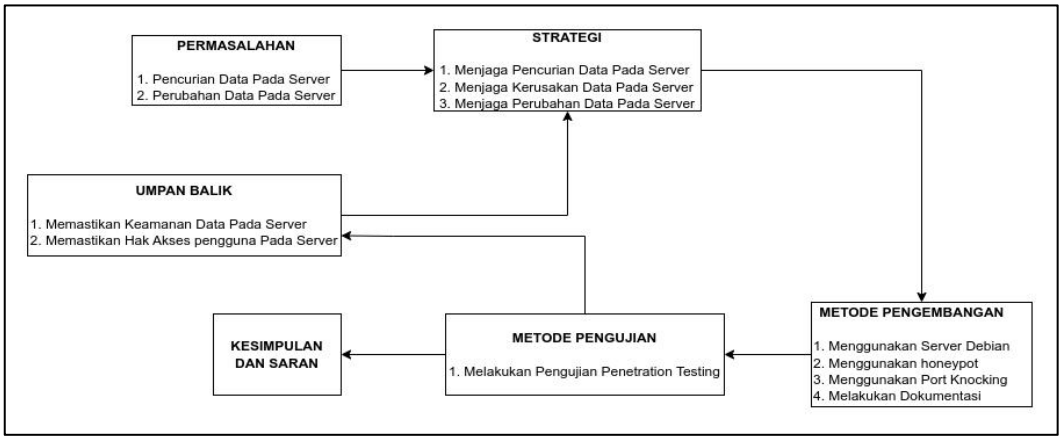
Sebagai bahan pertimbangan dalam penelitian ini akan dicantumkan berbagai hasil penelitian terdahulu, berikut hasil penelitian terdahulu yang dapat dilihat pada tabel 3.1 :

**Tabel 3.1 Penelitian Terdahulu**

| No | Judul | Nama/Tahun | Hasil Penelitian |
|---|---|---|---|
| 1. | Implementasi metode port knocking pada sistem keamanan server ubuntu virtual berbasis web monitoring | Rosalia Ernawati, Ikhwan Ruslianto, Syamsul Bahri Jurnal Komputer dan Aplikasi p-ISSN : 2338-493X, e-ISSN : 2809-574X Tahun : 2022 | Berdasarkan hasil penelitian metode port knocking berhasil mengamankan server dengan layanan jaringan celah port terbuka, dengan cara memblok port 22 dan melakukan knock pada server untuk membuka port SSH (22) sewaktu-waktu diperlukan layanan. |
| 2. | Optimasi port knocking dan honeypot menggunakan iptables sebagai keamanan jaringan pada server | Ahmad Zafrullah Mardiansyah, Yayank Muhammad Abdussyakur, Andy Hidayat Jatmika. Jurnal Teknologi Informasi Komputer dan Aplikasinya, ISSN:2657-0327 Tahun: 2021 | Dapat disimpulkan bahwa dengan adanya penambahan metode IPTables dapat meningkatkan kinerja baik dari segi penggunaan CPU (38,4%) dan Memory (44,2%) pada server maupun keamanan jaringan dibandingkan dengan hanya menggunakan metode Port knocking dan Honeypot saja. |
| 3. | Implementasi honeypot dengan metode honeytrape | Reyzal Hildha Hassan, Setia Juli IrzalI smail ,Periyadi. Jurnal e-Proceeding of Applied Science 6(2). ISSN : 2442-5826 Tahun : 2020 | Setelah melakukan pengujian terhadap implementasi Honeypot dengan metode Honeytrap, maka dapat disimpulkan bahwa:<br>1.Honeytrap yang dibangun berhasil memperkuat keamanan Web Server yaitu dengan cara mengalihkan penyerang ke server palsu.<br>2.Honeytrap yang dibangun dapat mendeteksi dari serangan port SSH, IP Address Target, dan Tanggal Waktu. |

Berdasarkan dari tabel peneletian terdahulu diatas, penelitian yang akan diterapkan adalah membuat *honeypot* menggunakan *server linux ubuntu* dan *port knocking* sebagai keamanan pada *server honeypot* tersebut. Dengan menggunakan cara itu diharapkan *cracker* (penyusup) dapat terjebak pada *server honeypot* yang dibuat. *Port knocking* sebagai keamanan *honeypot* supaya *cracker* (penyusup) akan kesulitan untuk mendapatkan akses pada *server honeypot* hal tersebut akan meyakinkan bahwa yang diserang adalah *server* asli tapi yang sebenarnya adalah *honeypot server*.

## 3.3. Kerangka Pemikiran




Sumber : Peneliti

**Gambar 3.1 Kerangka Pemikiran.**

Kerangka pemikiran diawali bagaimana menerapkan *honeypot* dan *port knocking* pada *server*. Setelah melakukan identifikasi masalah peneliti menggunakan sebuah metode penelitian.

# BAB IV

# METODE PENELITIAN

## 4.1. Lokasi dan Waktu Penelitian

### 4.1.1. Lokasi

Lokasi penelitian dilakukan di Dinas Komunikasi dan Informatika Kota Palembang yang beralamat di Jalan. Nyoman Ratu No. 1271, Sungai Pangeran, Kota Palembang, Sumatera Selatan 30113.

### 4.1.2. Waktu Penelitian

Penelitian ini dilakukan pada bulan Oktober 2022 sampai dengan Maret 2023. Adapun jadwal penelitian dapat dilihat pada tabel 4.1.

**Tabel 4.1 Waktu Penelitian.**

| No | Kegiatan | Tahun 2022 | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Oktober | | | | November | | | | Desember | | | | Januari | | | | Febuari | | | | Maret | | | |
| | | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 | 1 | 2 | 3 | 4 |
| 1 | Analisis | | | | | | | | | | | | | | | | | | | | | | | | |
| 2 | Perancangan | | | | | | | | | | | | | | | | | | | | | | | | |
| 3 | Implementasi | | | | | | | | | | | | | | | | | | | | | | | | |
| 4 | Pengujian | | | | | | | | | | | | | | | | | | | | | | | | |
| 5 | Proposal dan laporan skripsi | | | | | | | | | | | | | | | | | | | | | | | | |

## 4.2. Teknik Pengumpulan Data

### 4.2.1. Observasi

Dalam penelitian ini peneliti melakukan observasi yaitu dengan cara mengambil data secara langsung di lokasi penelitian yang dalam hal ini berarti Dinas Komunikasi dan Informatika Kota Palembang.

### 4.2.2. Wawancara

Wawancara memungkinkan penulis sebagai pewawancara *(interview)* untuk mengumpulkan data secara tatap muka langsung terkait dalam memberikan informasi mengenai pengelolaan jaringan *server* yang dilakukan pada Dinas Komunikasi dan Informatika Kota Palembang.

### 4.2.3. Studi Literatur

Studi literatur dengan mengumpulkan data dengan cara mencari dan mempelajari referensi lain yang berhubungan dengan penulisan laporan penelitian. Literatur yang digunakan dapat berupa jurnal ilmiah penelitian sebelumnya, buku-buku terkait serta referensi lainnya yang mendukung penelitian.

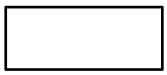
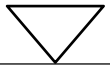
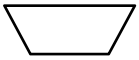
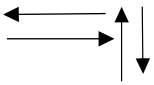
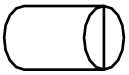
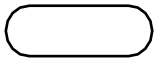
## 4.3. Alat Pengembangan Sistem

### 4.3.1. *Flowchart*

Menurut Susanti (2020), *Flowchart* adalah penggambaran secara grafik dari langkah-langkah dan urut-urutan prosedur dari suatu program. *Flowchart* menolong *analyst* dan *programmer* untuk memecahkan masalah kedalam segmen-segmen yang lebih kecil dan menolong dalam menganalisis alternatif-alternatif lain dalam pengoperasian. *Flowchart* biasanya mempermudah penyelesaian suatu masalah khususnya masalah yang perlu dipelajari dan dievaluasi lebih lanjut.

*Flowchart* adalah bentuk gambar/*diagram* yang mempunyai aliran satu atau dua arah secara sekuensial. *Flowchart* digunakan untuk merepresentasikan maupun mendesain program. Oleh karena itu *flowchart* harus bisa merepresentasikan komponen-komponen dalam bahasa pemrograman. Beberapa simbol-simbol standar diperlihatkan pada tabel 4.2 :

**Tabel 4.2 Simbol Umum *Flowchart***

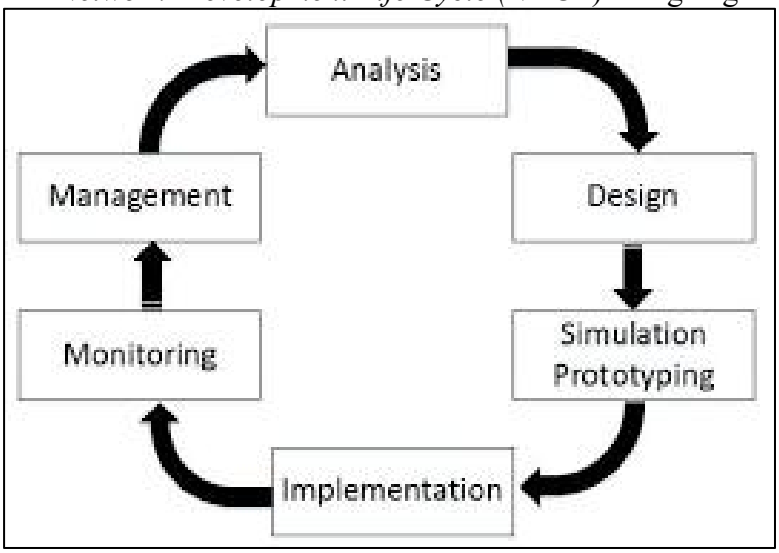
| No | Gambar | Nama | Keterangan |
|---|---|---|---|
| 1 |  | Komputerisasi | Untuk proses pengolahan data secara komputerisasi |
| 2 |  | Penghubung | Digunakan untuk menghubungkan sambungan aliran |
| 3 |  | Dokumen | Digunakan untuk operasi *input* |
| 4 |  | Arsip | Merupakan arsip data yang dihasilkan |
| 5 |  | Proses Manual | Untuk proses pengolahan data secara manual |
| 6 |  | Aliran Sistem | Untuk arah pengaliran data proses |
| 7 |  | Basis Data | Untuk media penyimpanan secara terkomputerisasi |
| 8 |  | Terminator | Mulai dan akhir |
| 9 |  | Pita Kertas | Untuk menunjukan *input/output* menggunakan pita kertas |

Sumber: Susanti (2020)

## 4.4. Metode Pengembangan

Sedangkan pemodelan yang digunakan untuk pengembangan sistem yang dilakukan adalah Model *Network Development Life Cycle (NDCL)*. Menurut Suharto & Irfan, 2019) *Network Development Life Cycle (NDLC)* yang merupakan serangkaian aktivitas yang dilaksanakan oleh profesional dan pemakai sistem jaringan untuk mengembangkan dan mengimplementasikan sistem jaringan. Penulis akan menggunakan metode *Network Development Life Cycle (NDCL)* untuk perancangan keamanan jaringan pada *server* Dinas Komunikasi dan Informatika Kota Palembang.

Konfigurasi dengan menggunakan *honeypot* dan *port knocking*. Metode tersebut terdiri dari *analysis*, *design*, *simulation prototype*, *implementation*, *monitoring* dan *management*. Berikut merupakan tahapan dari metode *Network Development Life Cycle (NDCL*) sebagai gambar 3.1 :




Sumber : Suharto & Irfan (2019:44).

**Gambar 3.1 Model *Network Development Life Cycle*.**

**A. *Analysis***

Setelah mendapatkan data-data pendukung selanjutnya peneliti melakukan analisis proses menganalisa server pada Dinas Komunikasi dan Informatika Kota Palembang yang dilakukannya analisa kebutuhan, analisa permasalahan yang muncul, analisa keinginan pengguna, dan analisa topologi jaringan yang sudah ada saat ini. Spesifikasi kebutuhan perangkat lunak pada tahap ini perlu didokumenasikan.

**B. *Design***

Tahap *design* dilakukan pembuatan *flowchart* dan topologi rancangan berdasarkan dari data yang diperoleh pada saat proses *analysis* dan menentukan skema yang diinginkan dengan menerapkan sistem *server honeypot* dan *port knocking* pada tahap selanjutnya desain yang dihasilkan pada tahap ini juga perlu di dokumentasikan.

**C. *Simulation*/*Prototyping***

Tahap ini peneliti membangun skema alur penerapan keamanan jaringan *server* pada sistem keamanan yang baru menggunakan *software* sistem operasi *ubuntu* untuk simulasi jaringan keamanan *server*.

**D. *Implementation***

Implementasi adalah tahap menerapkan setiap kebutuhan untuk membangun sistem. Implementasi dimulai dari *install* sistem operasi *linux ubuntu* sebagai server *honeypot* berfungi sebagai *server* palsu yang sengaja disusupi oleh penyusup, *install knockd* sebagai memfilter *port* 22 untuk layanan *service SSH*. Sehingga untuk mengakses *port* 22 tidak bisa sembarangan masuk untuk *client* yang memang tidak dikenal.

**E. *Monitoring***

Melakukan pengujian terhadap *server* yang telah dirancang serta menyimpulkan hasil pengujian dan memastikan bahwa semua

bagian sudah diuji. Pengujian dengan menggunakan *tools penetration testing* seperti *nmap*, *wireshark*, *hydra* dan *torshammer*. dan *monitoring* yang dilakukan bertujuan untuk menganalisa dan mendeteksi serangan pada layanan *server* yang telah dibangun.

**F. *Management***

Pada tahap management, dilakukan dengan membuat aturan yang disepakati guna mengatur pengguna sistem yang sudah dikembangkan agar dapat berlangsung lama dan dapat memenuhi harapan.

# BAB V

# HASIL DAN PEMBAHASAN

## 5.1. Hasil dan Pembahasan

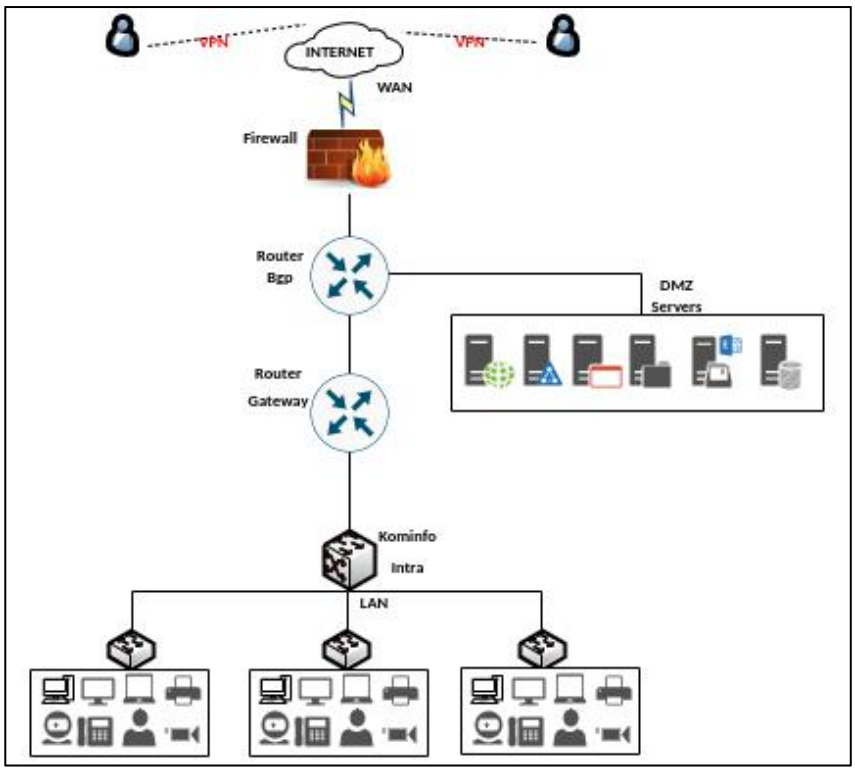
Dalam penerapan *port knocking* dan *honeypot* ini menggunakan metode *Network Development Life Cycle (NDCL),* adapun tahapannya adalah sebagai berikut:

### 5.1.1. *Analysis*

Pada tahapan ini penulis melakukan *analysis* kebutuhan, *analysis* permasalahan yang muncul, *analysis* keinginan pengguna dan *analysis* topologi yang sudah ada saat ini. Metode yang digunakan adalah seperti berikut :

**1) Observasi**

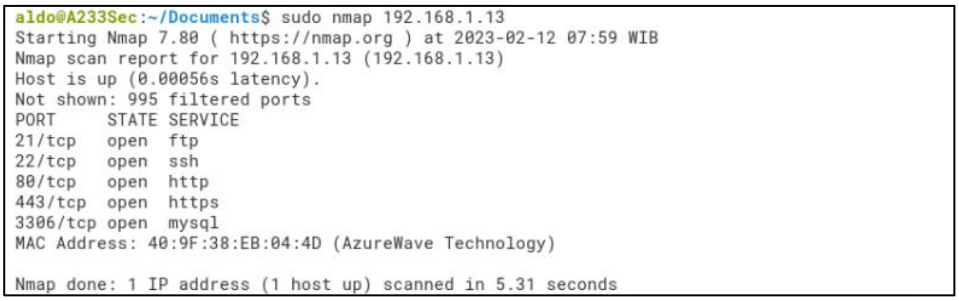
Dalam tahap ini, penulis melakukan observasi langsung pada kantor Dinas Komunikasi dan Informatika Kota Palembang dengan mengambil data secara langsung dengan cara melihat permasalahan disana dan melakukan pengujian *penetration testing* dimana sistem jaringan *server* tersebut belum terdapat *honeypot* sebagai untuk mencegah penyusup masuk ke *server* dan keamanan *port knocking* yang merupakan metode mengakses *port* yang telah diblok dengan mengirimkan paket atau koneksi sesuai dengan aturan *knocking* yang telah dibuat. Gambar 5.1 topologi sebelum perancangan :




Sumber : Peneliti.

**Gambar 5.1 Topologi Sebelum Perancangan.**

Setelah mengamati topologi diatas selanjutnya peneliti akan melakukan pengujian *penetration testing* pada *server* sebelum melakukan penerapan keamanan *honeypot*, *port knocking* dan *iptables*. Berikut adalah proses *server* sebelum menggunakan metode *honeypot*, *port kocking* dan *iptables.* Gambar 5.2 pengujian *port scanning server* sebelum implementasi *honeypot*, *port knocking* dan *iptables* :




Sumber : Peneliti.

**Gambar 5.2 Pengujian *Port Scanning Server* Sebelum Implementasi *Honeypot*, *Port Knocking* dan *Iptables***

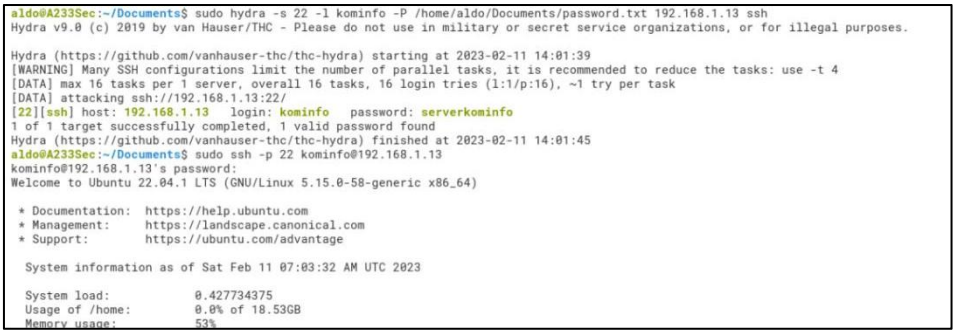
Pada gambar 5.2 dimana peneliti melakukan *port scanning* untuk melihat *vulnerability server* dimana penyerang melihat *port* 21 dan 22 yang *open* dan penyerang berusaha menyerang *port* tersebut untuk mengambil alih *server*. Selanjutnya peneliti akan melakukan *brute force* untuk menyerang *port* 21 dan 22. Gambar 5.3 proses serangan *brute force* pada *port* 21 *FTP* :


Sumber : Peneliti.

**Gambar 5.3 Proses Serangan *Brute Force* Pada *Port* 21 *FTP***

Pada gambar 5.3 peneliti berhasil menyerang *server* pada *port* 21 *FTP* dengan menggunakan serangan *brute force* penyerang tersebut bisa mengakses atau mengeksploitasi *server* melalui *port* 21 *FTP*. Setelah berhasil melakukan serangan terhadap *port* 21 selanjutnya peneliti menyerang *port SSH* yaitu *port* 22. Gambar 5.4 proses serangan *brute*

*force* pada *port* 22 *SSH* :

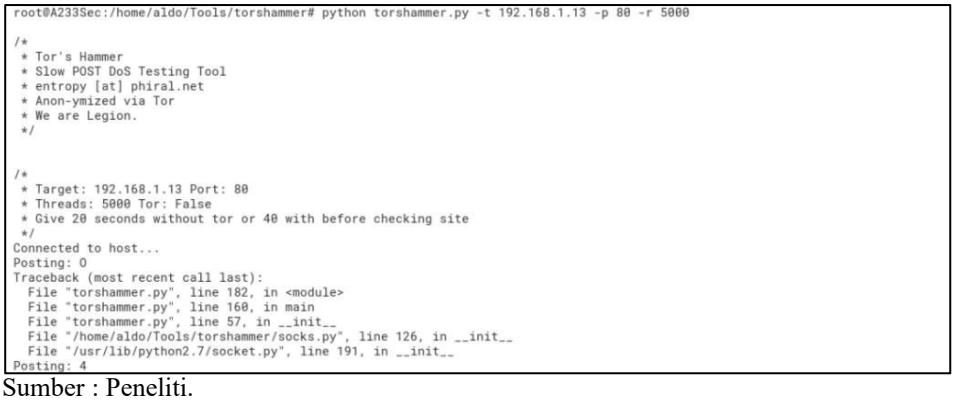



Sumber : Peneliti.

**Gambar 5.4 Proses Serangan *Brute Force* Pada *Port* 22 *SSH***

Pada gambar 5.4 selanjutnya peneliti menyerang *port* 22 *SSH* dengan menggunakan *metode brute force* dan hasilnya peneliti berhasil menyerang *port* 22 *SSH* dengan mendapatkan akses *login username* dan *password* yang sangat berbahaya untuk *server* karena mendapatkan akses *SSH* sepenuhnya untuk mengakses dan mengeksploitasi *server*.

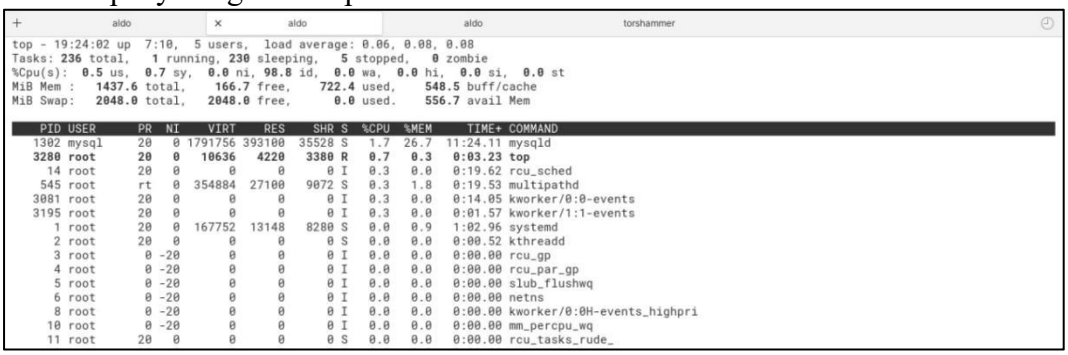
Selanjutnya pada penelitian ini dilakukan juga pengujian penyerangan menggunakan metode *Denial of Service* terhadap *server*. Dengan menggunakan *software torshammer* ini yang merupakan *tool* yang digunakan untuk melakukan penyerangan *DoS*. Dengan tool ini dapat melakukan penyerangan berdasarkan *IP* dan menentukan *port* serangan yang diinginkan. Gambar 5.5 proses serangan *Denial of Service* menggunakan *torshammer*:




Sumber : Peneliti.

**Gambar 5.5 Serangan *DoS* Menggunakan *Torshammer***

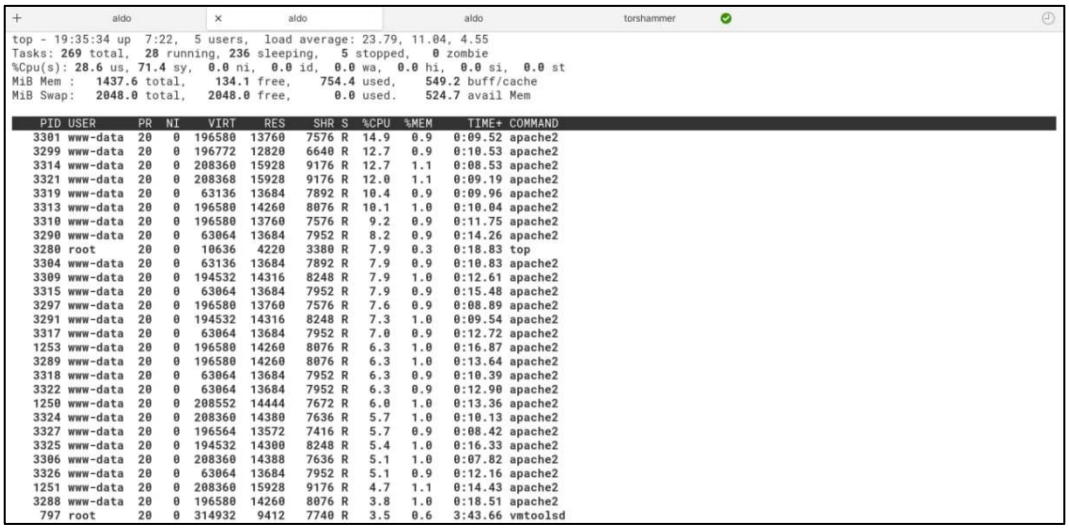
Pada gambar 5.5 menunjukkan penyerangan yang menggunakan *tool torshammer* dimana *server* yang akan diserang adalah dengan *IP address* "192.168.1.13" dengan tujuan *port* penyerangan yaitu *port* 80 dengan *method TCP* dan *thread* yang digunakan sebanyak 5000. Penyerangan akan dilakukan dalam waktu 83 menit dan akan dilihat penggunaan *CPU* dan Memori pada *server* menggunakan perintah *TOP* yang berfungsi sebagai memanajemen proses yang berjalan pada *server*. Gambar 5.6 proses memanajemen yang sedang berjalan sebelum penyerangan *DoS* pada *server* :




Sumber : Peneliti.

**Gambar 5.6 Proses Memanajemen Yang Sedang Berjalan Sebelum Penyerangan *DoS* Pada *Server***

Pada gambar 5.6 terlihat bahwa sebelum penyerangan *DoS atttack* proses manajemen yang sedang berjalan menujukkan hasil proses dari perintah *TOP* pada *server* jumlah waktu yang dihabiskan dalam ruang *kernel (sy)* pada *CPU* sebanyak *(0.7 sy)*. Gambar 5.7 proses memanajemen yang sedang berjalan sesudah penyerangan *DoS* pada *server* :

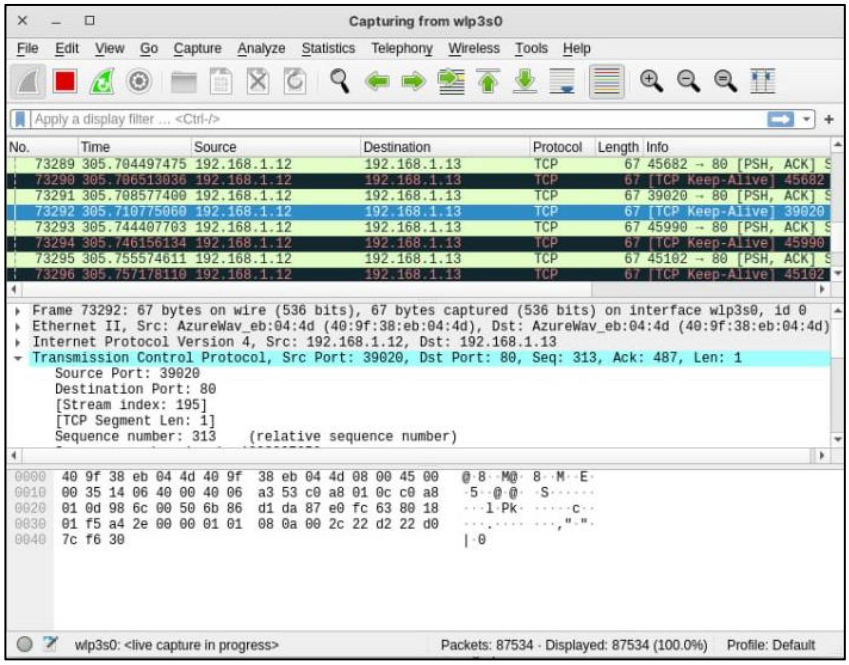



Sumber : Peneliti.

**Gambar 5.7 Proses Memanajemen Yang Sedang Berjalan Sesudah Penyerangan *DoS* Pada *Server***

Setelah dilakukanya serangan pada *server* terlihat bahwa *DoS attack* membanjiri *server* dengan *request* dari *port* 80 sesuai yang dilakukan penyerangan menggunakan *port web server* yaitu *port* 80, terlihat perubahan sesudah penyerangan dalam ruang *kernel (sy)* pada *CPU* sebanyak *(71.4 sy)*. Berikutnya pada memori dapat dilihat jumlah memori sebelum dilakukan penyerangan pada gambar 5.25 Sebesar 166.7 *free* dan pada saat penyerangan pada gambar 5.26 Sebesar 134.1 *free*

menunjukkan penggunaan memori sebesar 32.6 *free*. Gambar 5.8 proses hasil *capture wireshark* :




Sumber : Peneliti.

**Gambar 5.8 Proses Hasil *Capture Wireshark***

Berdasarkan hasil *capture wireshark* pada gambar 5.8 dapat dilihat bahwa paket serangan pada *port* 80 benar menuju target yang akan diserang dari hasil pengujian menggunakan *tool torshammer* sehingga dapat disimpulakan bahwa serangan sukses dilakukan menuju target.

Setelah dilakukannya pengujian sebelum menggunakan metode keamanan *honeypot*, *port knocking* dan *iptables* selanjutnya dilakukan pengamatan hasil dari proses pengujian sebelum menggunakan keamanan *honeypot*, *port knocking* dan *iptables*. Tabel 5.1 hasil tabel pengujian sebelum implementasi keamanan *honeypot*, *port knocking* dan *iptables* :

**Tabel 5.1 Hasil Tabel Pengujian Sebelum Implementasi Keamanan *Honeypot*, *Port Knocking* dan *Iptables***

| Serangan | Port | CPU | Memory | Keterangan | |
|---|---|---|---|---|---|
| | | | | Berhasil | Gagal |
| *Denial of Service* | 21 | 0,7 sy | 273.1 | | ✓ |
| | 22 | 0,5 sy | 266,7 | | ✓ |
| | 80 | 71,4 sy | 134.1 | ✓ | |
| | 443 | 67,0 sy | 132,6 | ✓ | |
| | 3306 | 72,7 sy | 123,4 | ✓ | |
| *Bruteforce* | 21 | 1,6 sy | 182.5 | ✓ | |
| | 22 | 5,1 sy | 194,9 | ✓ | |
| | 80 | 3,6 sy | 182,6 | | ✓ |
| | 443 | 3,2 sy | 182,9 | | ✓ |
| | 3306 | 4,8 sy | 191,7 | | ✓ |

Berdasarkan hasil pengujian penyerangan sebelum implementasi keamanan *honeypot*, *port knocking* dan *iptables* yang menuju *port* 21, 22, 80, 443 dan 3306 pada tabel 5.1, pada penyerangan *Denial of Service* menuju port 80, 443 dan 3306 menyebabkan peningkatan performa *CPU* dan memori pada *server* yang membuat *server* banyak dibanjiri *request* atau *fake traffic* yang membuat *server* menjadi *down*, sedangkan *port* 21, 22 performa *CPU* dan memori tetap stabil. Sedangkan pada penyerangan menggunakan *brute force* menuju *port* 21 *FTP* dan *port* 22 *SSH* berhasil dilakukan penyerang dan penyerang bisa untuk mengakses dan mengeksploitasi *server*.

**2) Wawancara**

Pada tahap wawancara dilakukan dengan bertanya kepada pihak-pihak yang terkait yaitu kepada seksi tata kelola, kepala seksi persandian dan kepala seksi insfrastruktur bidang *E-Govermment* dalam memberikan informasi mengenai pengelolaan jaringan pada server Dinas Komunikasi dan Informatika Kota Palembang.

**3) Studi Literatur**

Peneliti melakukan pengumpulan data dengan cara mencari data, mencari referensi lain yang berhubungan dengan penelitian tentang keamanan *server* dan peneliti mempelajari data-data yang didapatkan setelah melakukan observasi dan wawancara.

**4) Analisis Kebutuhan**

**1. Analisis Kebutuhan Fungsional**

Kebutuhan fungsional adalah jenis kebutuhan yang berisi proses-proses apa saja yang diberikan sistem tersebut. Kebutuhan pengguna untuk melakukan keamanan *server* Dinas Komunikasi dan Informatika Kota Palembang dengan penerapan *honeypot, port knocking* dan *Iptables*.

**2. Analisis Kebutuhan Perangkat Lunak**

Kebutuhan perangkat lunak yang diperlukan untuk mendukung aplikasi yang dibangun adalah sebagai berikut :

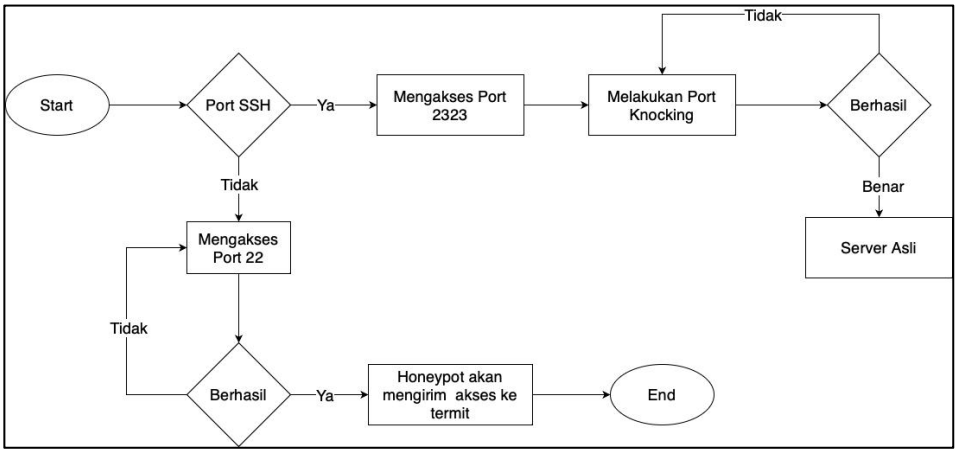
1. Sistem operasi *ubuntu.*

2. *Proxmox.*

3. *Honeypot, Port Knocking* dan *Iptables.*

### 5.1.2. *Design*

Tahap *design* dilakukan pembuatan *flowchart* dan topologi rancangan berdasarkan dari data yang diperoleh pada saat proses *analysis* dan menentukan skema yang diinginkan dengan menerapkan sistem *server honeypot* dimana *server honeypot* terletak pada *port* default *SSH* yaitu 22 yang membuat penyerang akan menyerang *port 22 SSH honeypot* dan jika masuk *port SSH* benar maka harus dilakukan *port knocking*. Gambar 5.9 *flowchart* perancangan:

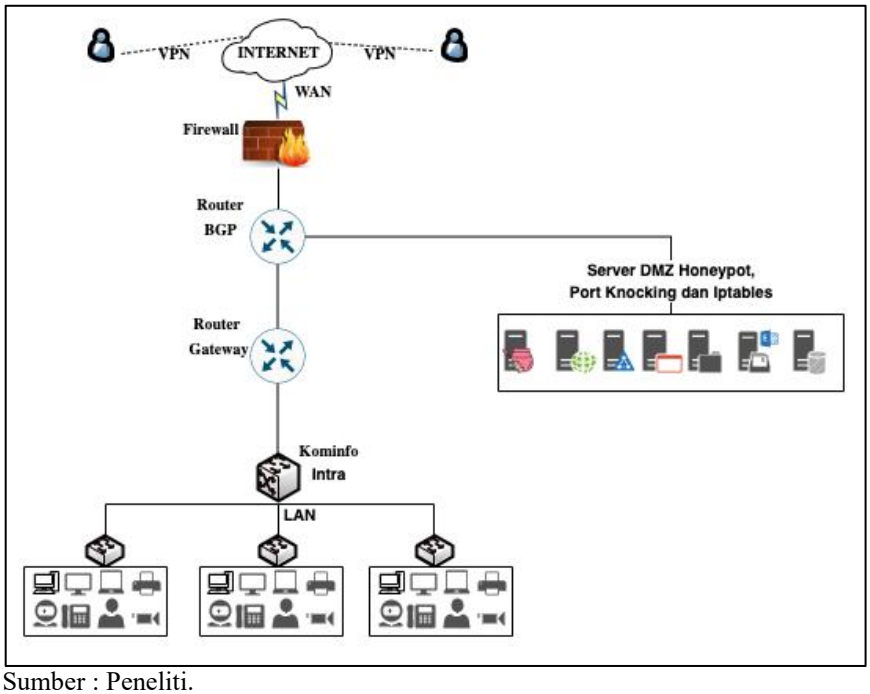



Sumber : Peneliti.

**Gambar 5.9 *Flowchart* Perancangan.**

Berdasarkan *flowchart* diatas dan topologi sebelum perancangan *honeypot platform* dan *port knocking* dapat disimpulkan peneliti untuk membuat hasil peneletian dapat dilihat

dalam topologi sesudah perancangan. Gambar 5.10 topologi yang diusulkan :

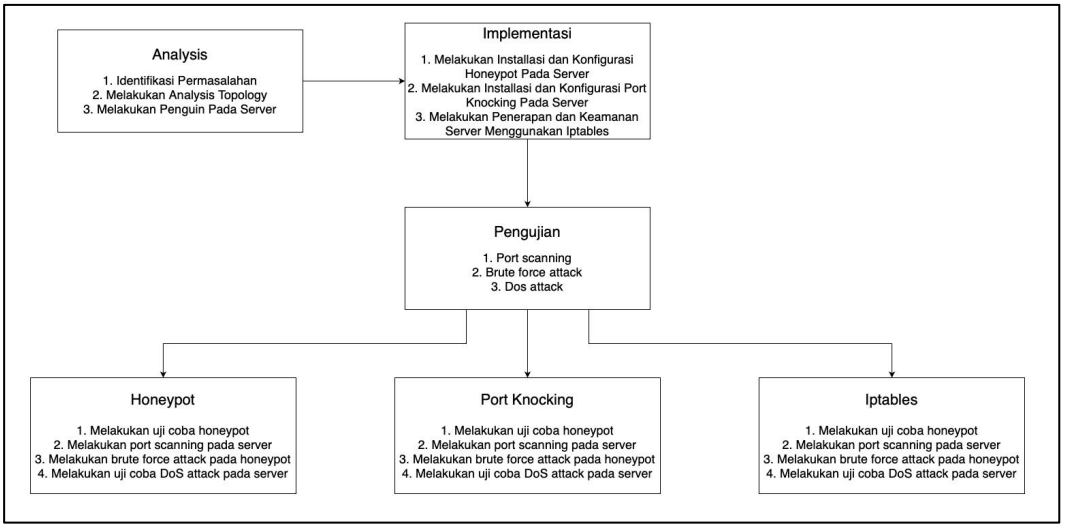



Sumber : Peneliti.

**Gambar 5.10 Topologi Yang Diusulkan**

### 5.1.3. *Simulation*/*Prototyping*

Tahapan ini adalah tahapan simulasi *simulation*/*prototyping* setelah merancang skema alur penerapan pada *server* dan *design* topologi jaringan untuk mendapatkan hasil yang diharapkan dari penerapan *honeypot* dan *port knocking*, selanjutnya dilakukan uji coba dengan melakukan konfigurasi pada *server* Dinas Komunikasi dan Informatika Kota Palembang yang akan menerapkan keamanan jaringan *server* dengan menggunakan *honeypot* dan *port knocking*. Gambar 5.11 bagan simulasi keamanan jaringan :




Sumber : Peneliti.

**Gambar 5.11 Bagan Simulasi Keamanan Jaringan**

Pada gambar 5.11 peneliti melakukan *analysis* identifikasi permasalahan keamanan jaringan *server* setelah permasalahan didapatkan selanjutnya peneliti melakukan implementasi keamanan *server* berdasarkan permasalahan dengan menerapkan *honeypot*, *port knocking* dan *iptables* dan melakukan pengujian hasil implementasi tersebut, selanjutnya peneliti mengambil kesimpulan dan *analysis* hasil pengujian penerapan *honeypot* dan *port knocking*.

### 5.1.4. *Implementation*

Pada tahap ini penulis melakukan konfigurasi terhadap *server* Dinas Komunikasi dan Informatika Kota Palembang sebagai penerapan keamanan dengan menggunakan *honeypot* dan *port knocking* dan *iptables.*

**1) *Port Knocking***

*Port Knocking* merupakan untuk mengontrol akses ke *port* dengan hanya mengizinkan akses pengguna yang sah ke layanan *server*. Proses ini bekerja sedemikian rupa sehingga ketika urutan upaya koneksi yang benar dibuat, *firewall* akan membuka *port* yang ditutup. Logika di balik *port knocking* adalah untuk melindungi sistem *server* dari pemindai *port* otomatis yang akan berusaha untuk menyusup pada *server*. Proses untuk menginstall *port knocking* pada *server* dapat menggunakan *script command* sebagai berikut. Gambar 5.12 install *port knocking* server :

```
sudo apt-get install knockd -y
```

```
kominfo@server:~$ sudo apt-get install knockd -y
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  knockd
0 upgraded, 1 newly installed, 0 to remove and 63 not upgraded.
Need to get 29.1 kB of archives.
After this operation, 111 kB of additional disk space will be used.
Get:1 http://id.archive.ubuntu.com/ubuntu jammy/universe amd64 knockd amd64 0.8-2 [29.1 kB]
Fetched 29.1 kB in 2s (17.4 kB/s)
Selecting previously unselected package knockd.
(Reading database ... 85820 files and directories currently installed.)
Preparing to unpack .../knockd_0.8-2_amd64.deb ...
Unpacking knockd (0.8-2) ...
Setting up knockd (0.8-2) ...
Processing triggers for man-db (2.10.2-1) ...
Scanning processes...
Scanning linux images...
```

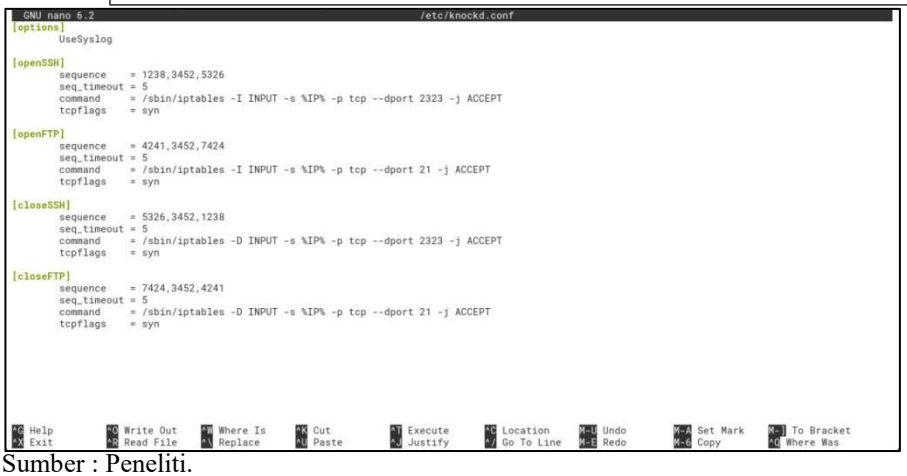
Sumber : Peneliti.

**Gambar 5.12 Install *Port Knocking Server***

Setelah melakukan penginstalan *port knocking* dan selanjutnya melakukan konfigurasi untuk menentukan ketukan yang berfungsi untuk melakukan *remote* pada *server*. Berikut dapat dilihat program yang dijalankan untuk mengakses direktori penyimpanan aktivitas *knock* pada sistem dapat

menggunakan *script command* sebagai berikut. Gambar 5.13 Proses konfigurasi *port knocking* :

```
sudo nano /etc/knockd.conf
```




Sumber : Peneliti.

**Gambar 5.13 Proses Konfigurasi *Port Knocking***

Proses konfigurasi *port knocking* yaitu terdapat beberapa bagian yaitu *options, openSSH, openFTP, closeSSH, closeFTP*. Pada yang pertama *option* dapat digunakan untuk memberikan *log* aktivitas yang berusaha untuk masuk kedalam *server*, kedua berikutnya pada *openSSH* terdapat *sequence* = 1238, 3452, 5326 adalah kumpulan urutan ketukan nomor *port* yang berfungsi sebagai autentikasi untuk membuka pengaksesan layanan *port knocking ssh* pada *server*.

Perintah yang ketiga terdapat juga *seq_timeou*t untuk menentukan berapa lama urutan akan benar, yang keempat terdapat "*command = /sbin/iptables -I INPUT -s %IP% -p tcp – dport 2323 - j ACCEPT*" yang merupakan perintah *iptables* yang menambahkan aturan untuk mengizinkan koneksi *SSH*

masuk dari alamat *IP Address* yang ditentukan. Perintah ini hanya akan dijalankan jika pengguna memulai urutan ketukan yang benar. Perintah ini akan dilakukan saat *client* yang berusaha masuk kedalam *port* 2323 yang dimana apabila ketukan yang dilakukan benar maka akan di terima.

Selanjutnya perintah yang kelima adalah *tcpflags* mendefinisikan jenis paket *TCP* yang akan diterima dalam koneksi ketukan. Paket *TCP* tipe *SYN* ditugaskan dalam kasus ini.

Pada *closeSSH* terdapat *sequence* = 5326, 3452, 1238 adalah kumpulan urutan ketukan nomor *port* yang berfungsi sebagai autentikasi untuk menutup pengaksesan kembali layanan *port knocking SSH* pada *server*.

Perintah yang ketiga terdapat juga *seq_timeou*t untuk menentukan berapa lama urutan akan benar, yang keempat terdapat "*command = /sbin/iptables -D INPUT -s %IP% -p tcp --dport 2323 -j ACCEPT*" merupakan perintah untuk menutup kembali akses *SSH port* 2323 dengan ketukan autentifikasi yang telah di tentukan. Perintah ini hanya akan dijalankan jika pengguna memulai urutan ketukan yang benar. *Port* 2323 merupakan *port SSH* yang dikonfigurasi pada */etc/ssh/sshd_config* yang merupakan *random port* yang dipilih untuk menggantikan *SSH default (port 22)* dikarenakan *port* 22

akan digunakan untuk mengakses *honeypot*. Proses untuk *knocking* pada *port* 21 *FTP* sama seperti proses *port knocking SSH* yang dijelaskan seperti diatas.

**2) *Honeypot***

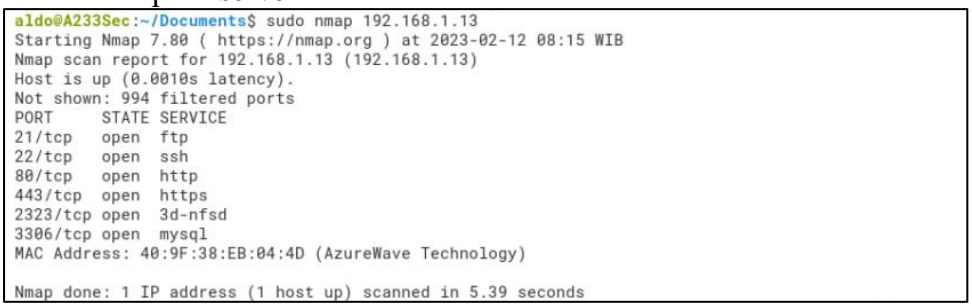
Proses install dan konfigurasi *honeypot* yang dibuat menggunakan *SSH server* palsu untuk mengelabui penyerang sehingga penyerang tidak menyerang *SSH server* yang asli dan dapat mengalihkan penyerang *port service* kepada *fake port* yang dibuat *honeypot.* Gambar 5.14 proses konfigurasi *honeypot* :


Sumber : Peneliti.

**Gambar 5.14 Proses Konfigurasi *Honeypot***

Setelah melakukan konfigurasi *honeypot* selanjutnya konfigurasi *port SSH* yang dimana *honeypot* memiliki akses terhadap *port* 22 awalnya adalah *port* untuk mengakses *Secure Shell (SSH)*, tetapi pada penelitian ini *port SSH* tersebut sudah menjadi *port honeypot* dan *port* yang *Secure Shell (SSH)* yang asli sudah dipindahkan ke *port* 2323. Gambar 5.15 *Client*

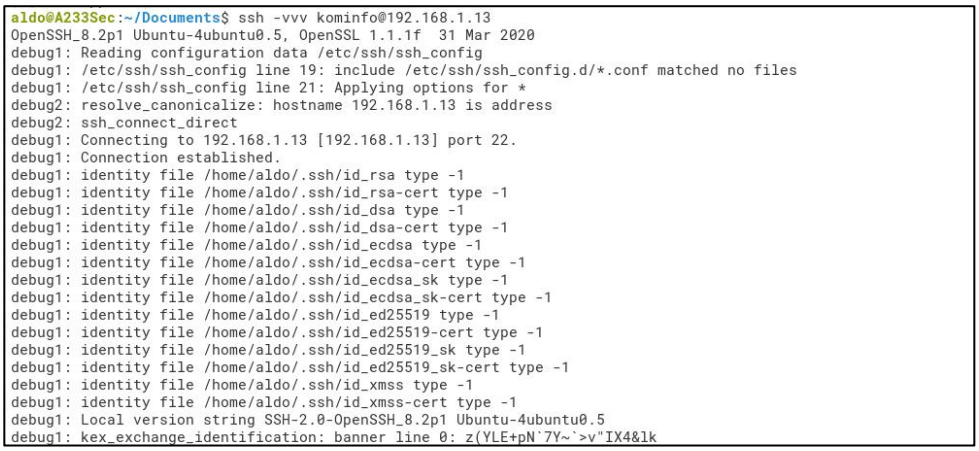
melakukan *port scanning* untuk mengecek *port* yang terbuka pada *server* :




Sumber : Peneliti.

**Gambar 5.15 *Client* Melakukan *Port Scanning* Untuk Mengecek *Port* Yang Terbuka Pada *Server***

Setelah *client* melakukan *port scanning* selanjutnya *client* akan melakukan mencoba akses *port SSH* yang terbuka yaitu *port* 22 dengan mencoba untuk melakukan mengambil akses pada *server* tersebut. Gambar 5.16 *Client* mencoba untuk mengakses *port* 22 pada *server* :




Sumber : Peneliti.

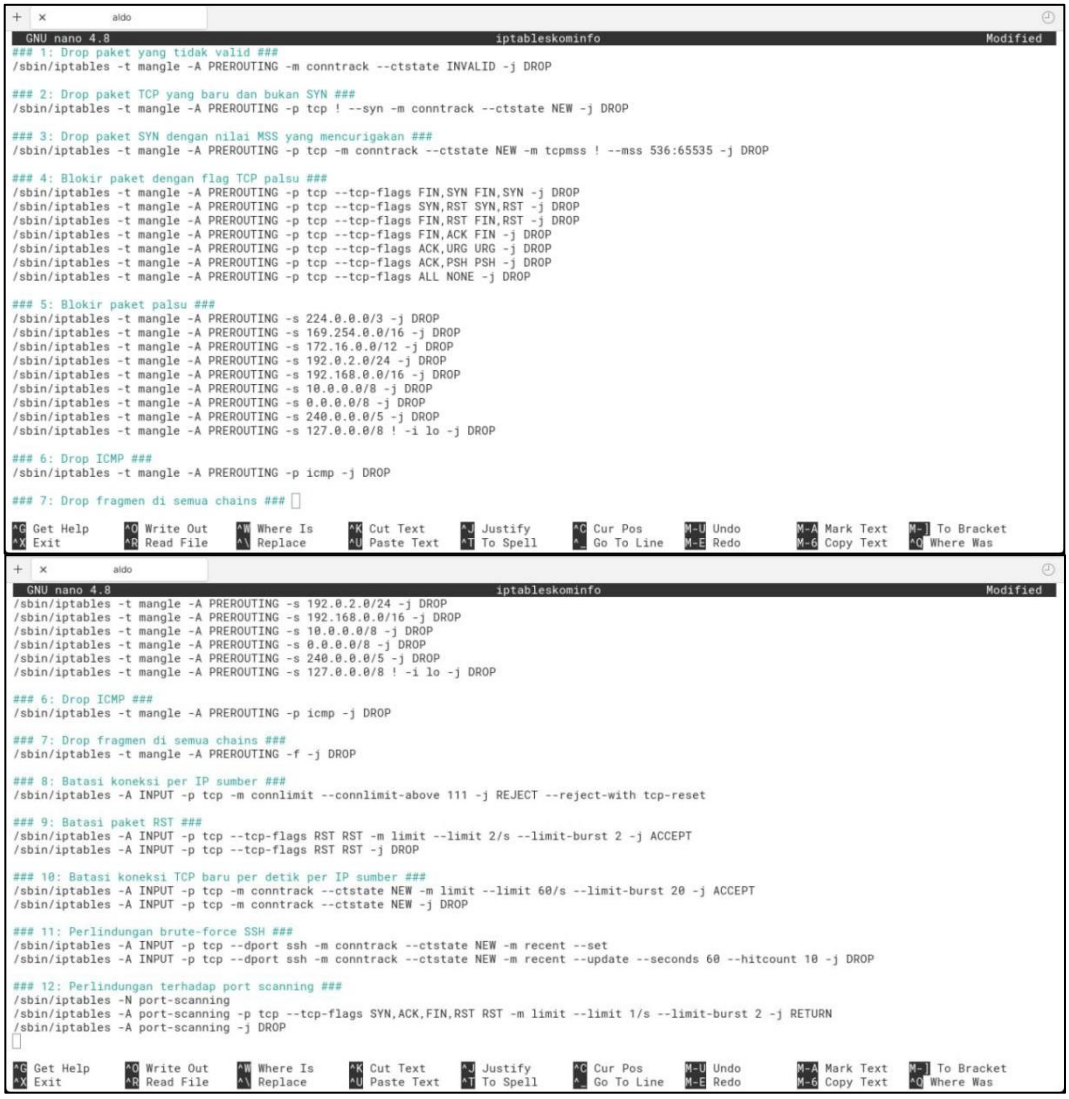
**Gambar 5.16 *Client* Mencoba Untuk Mengakses *Port* 22 Pada *Server***

Pada gambar diatas *client* mencoba untuk mengakses *server* pada *port* 22 yaitu *port* dari *honeypot* yang mana jika

diakses akan tidak bisa masuk ke dalam *server* yang asli dikarenakan *honeypot* akan melakukan perulangan autentikasi akses masuk ke *server* yang menunjukkan bahwa *client* terjebak di *tarpit* yang merupakan sebuah pelayanan atau *service* berupa *server* dalam sistem komputer yang berfungsi untuk menghambat atau menghentikan koneksi *client*.

**3) *Iptables***

Pada pengujian sistem yang telah dilakukan menggunakan *tools brute force attack (Hydra)* dan *DoS attack (torshammer)* didapatkan hasil penyerangan menggunakan *brute force attack (Hydra)* dapat diantisipasi oleh *honeypot* dan *port knocking*, sedangkan penyerangan menggunakan *DoS attack* tidak dapat diantisipasi oleh *server* yang membuat *server* dipenuhi *fake traffic* secara terus menerus dan membuat *server* menjadi *down* karena tidak dapat memenuhi *request*, sehingga untuk mengatasi serangan *DoS Attack* dibuat *rules Iptables* yang berfungsi untuk memblokir dan mengijinkan akses masuk atau keluar *server*. Gambar 5.17 konfigurasi *rules iptables* :




Sumber : Peneliti.

**Gambar 5.17 Konfigurasi *Rules Iptables***

*Iptables* yang pertama berfungsi sebagai perintah ini memblokir semua paket yang bukan paket *SYN* dan bukan milik koneksi *TCP* yang sudah ada. *Script command* sebagai berikut :

```
iptables -t mangle -A PREROUTING -m conntrack --ctstate INVALID -j DROP
```

*Iptables* yang kedua berfungsi sebagai perintah ini memblokir semua paket yang baru (bukan milik koneksi yang sudah ada) dan tidak menggunakan *flag SYN*. Aturan ini mirip dengan aturan "Blokir Paket Tidak *Valid*". *Script command* sebagai berikut :

```
iptables -t mangle -A PREROUTING -p tcp ! --syn -m conntrack --ctstate NEW -j DROP
```

*Iptables* yang ketiga berfungsi sebagai perintah aturan membatasi *MSS (Maximum Segment Size)* pada paket *TCP (Transmission Control Protocol)* yang masuk ke sistem. *MSS* adalah jumlah maksimal data yang dapat dikirimkan dalam satu segmen dalam protokol *TCP*. Dengan membatasi *MSS*, baris perintah ini memastikan bahwa paket *TCP* yang masuk ke sistem memiliki ukuran *MSS* yang tidak melebihi batas yang ditentukan. Baris perintah ini menggunakan modul *"mangle"* untuk memanipulasi paket yang masuk, dan memasukkan mereka ke tabel *"PREROUTING"* sebelum *routing*. Membatasi peraturan hanya untuk paket yang memiliki *MSS* di luar rentang 536 hingga 65535 Ini membantu memblokir *flood SYN* yang menyerang. *Script command* sebagai berikut :

```
iptables -t mangle -A PREROUTING -p tcp -m conntrack --ctstate NEW -m tcpmss ! --mss 536:65535 -j DROP
```

*Iptables* yang keempat berfungsi sebagai perintah memblokir paket yang menggunakan *flag TCP* palsu, yaitu.

*TCP* menandai bahwa paket yang sah tidak akan digunakan. *Script command* sebagai berikut :

```
iptables -t mangle -A PREROUTING -p tcp --tcp-flags FIN,SYN FIN,SYN -j DROP
iptables -t mangle -A PREROUTING -p tcp --tcp-flags SYN,RST SYN,RST -j DROP
iptables -t mangle -A PREROUTING -p tcp --tcp-flags FIN,RST FIN,RST -j DROP
iptables -t mangle -A PREROUTING -p tcp --tcp-flags FIN,ACK FIN -j DROP
iptables -t mangle -A PREROUTING -p tcp --tcp-flags ACK,URG URG -j DROP
iptables -t mangle -A PREROUTING -p tcp --tcp-flags ACK,PSH PSH -j DROP
iptables -t mangle -A PREROUTING -p tcp --tcp-flags ALL NONE -j DROP
```

*Iptables* yang kelima berfungsi sebagai perintah memblokir paket palsu yang berasal dari *subnet* pribadi (lokal). Pada antarmuka jaringan publik yang tidak ingin menerima paket dari *IP address private*. Aturan ini mengasumsikan bahwa antarmuka *loopback* yang menggunakan ruang *IP* 127.0.0.0/8. Lima set aturan ini sudah memblokir banyak serangan *DDoS* berbasis *TCP* dengan kecepatan paket yang sangat tinggi. Dengan pengaturan dan aturan *kernel* yang disebutkan pada *script command* dibawah ini yang dapat memfilter serangan *ACK* dan *SYN-ACK* dengan laju baris. *Script command* sebagai berikut :

```
iptables -t mangle -A PREROUTING -s 224.0.0.0/3 -j DROP
iptables -t mangle -A PREROUTING -s 169.254.0.0/16 -j DROP
iptables -t mangle -A PREROUTING -s 172.16.0.0/12 -j DROP
```

```
iptables -t mangle -A PREROUTING -s 192.0.2.0/24 -j DROP
iptables -t mangle -A PREROUTING -s 192.168.0.0/16 -j DROP
iptables -t mangle -A PREROUTING -s 10.0.0.0/8 -j DROP
iptables -t mangle -A PREROUTING -s 0.0.0.0/8 -j DROP
iptables -t mangle -A PREROUTING -s 240.0.0.0/5 -j DROP
iptables -t mangle -A PREROUTING -s 127.0.0.0/8 ! -i lo -j DROP
```

*Iptables* yang keenam berfungsi sebagai perintah *drop* semua paket *ICMP*. *ICMP* hanya digunakan untuk melakukan *ping* ke sebuah *host* untuk mengetahui apakah *host* tersebut masih bisa. Karena biasanya tidak diperlukan dan hanya mewakili kerentanan lain yang dapat dieksploitasi penyerang, peneliti memblokir semua paket *ICMP* untuk memitigasi *Ping of Death (ping flood)*, *ICMP flood* dan *ICMP fragmentation flood*. *Script command* sebagai berikut :

```
iptables -t mangle -A PREROUTING -p icmp -j DROP
```

*Iptables* yang ketujuh berfungsi sebagai *iptables* ini membantu melawan serangan koneksi. Itu menolak koneksi dari *host* yang memiliki lebih dari 80 koneksi yang dibuat. Jika menghadapi masalah apa pun harus menaikkan batas karena hal ini dapat menyebabkan masalah dengan klien sah yang membuat koneksi *TCP* dalam jumlah besar. *Script command* sebagai berikut :

```
iptables -A INPUT -p tcp -m connlimit --connlimit-above 80 -j REJECT --reject-with tcp-reset
```

*Iptables* yang kedelapan berfungsi sebagai membatasi koneksi *TCP* baru yang dapat dibuat klien per detik. Ini dapat berguna melawan serangan koneksi, tetapi banyak serangan terhadap *SYN* karena biasanya menggunakan *IP address* palsu yang berbeda dalam jumlah tak terbatas. *Script command* sebagai berikut :

```
iptables -A INPUT -p tcp -m conntrack --ctstate NEW -m limit --limit 60/s --limit-burst 20 -j ACCEPT
iptables -A INPUT -p tcp -m conntrack --ctstate NEW -j DROP
```

*Iptables* yang kesembilan perintah ini memblokir paket yang terfragmentasi. Tetapi serangan sering kali membanjiri fragmentasi *UDP* menggunakan *bandwidth* dalam jumlah besar yang cenderung menghabiskan kapasitas kartu jaringan. *Script command* sebagai berikut :

```
iptables -t mangle -A PREROUTING -f -j DROP
```

*Iptables* yang kesepuluh perintah ini membatasi paket *TCP RST* yang masuk untuk mengurangi serangan *TCP RST*. *Script command* sebagai berikut :

```
iptables -A INPUT -p tcp --tcp-flags RST RST -m limit --limit 2/s --limit-burst 2 -j ACCEPT
iptables -A INPUT -p tcp --tcp-flags RST RST -j DROP
```

*Iptables* yang kesebelas berfungsi perintah perlindungan *brute force* attack pada *SSH*. *Script command* sebagai berikut :

```
/sbin/iptables -A INPUT -p tcp --dport ssh -m conntrack --ctstate NEW -m recent --set
```

```
/sbin/iptables -A INPUT -p tcp --dport ssh -m conntrack --ctstate NEW -m recent --update --seconds 60 --hitcount 10 -j DROP
```

*Iptables* yang kesebelas berfungsi perlindungan terhadap *port scanning*. *Script command* sebagai berikut :

```
/sbin/iptables -N port-scanning  /sbin/iptables -A port-scanning -p tcp --tcp-flags SYN,ACK,FIN,RST RST -m limit --limit 1/s --limit-burst 2 -j RETURN

/sbin/iptables -A port-scanning -j DROP
```

### 5.1.5. *Monitoring*

Pada tahap ini, dilakukan pengamatan hasil implementasi untuk melihat dan memastikan konfigurasi berjalan sesuai dengan harapan dengan cara mencoba dan menguji skema yang telah ditentukan.

**a) *Honeypot***

Setelah selesai implementasi dan konfigurasi *honeypot* selanjutnya pengujian serangan *honeypot*, pada tahap ini yang dilakukan pertama adalah melakukan *port scanning* pada *server*. Gambar 5.18 pengujian *port scanning server* setelah melakukan installasi dan konfigurasi *honeypot* :

```
aldo@A233Sec:~/Documents$ sudo nmap 192.168.1.13
Starting Nmap 7.80 ( https://nmap.org ) at 2023-02-12 08:15 WIB
Nmap scan report for 192.168.1.13 (192.168.1.13)
Host is up (0.0010s latency).
Not shown: 994 filtered ports
PORT     STATE SERVICE
21/tcp   open  ftp
22/tcp   open  ssh
80/tcp   open  http
443/tcp  open  https
2323/tcp open  3d-nfsd
3306/tcp open  mysql
MAC Address: 40:9F:38:EB:04:4D (AzureWave Technology)

Nmap done: 1 IP address (1 host up) scanned in 5.39 seconds
```

Sumber : Peneliti.

**Gambar 5.18 Pengujian *Port Scanning Server* Setelah Melakukan Installasi dan Konfigurasi Honeypot**

Pada gambar 5.18 peneliti melakukan serangan *port scanning* pada *server* yang dimana ditemukan *port* baru yaitu *port* 2323 yang dimana *port* tersebut merupakan *port* manipulasi dari *port* 22 *SSH* dimana *port* 22 sudah digunakan untuk *port honeypot* jadi untuk akses *server SSH* berada pada *port SSH* 2323.

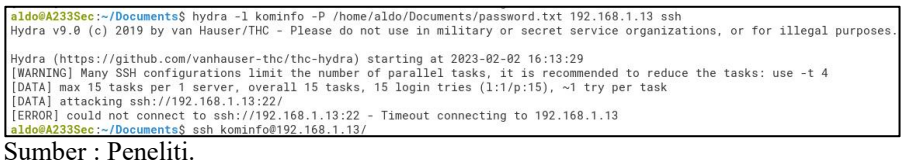
Setelah melakukan *port scanning* selanjutnya tahap penelitian ini dilakukan penyerangan *server* menggunakan metode *brute force hydra* terhadap *port* 22 yang merupakan *port honeypot* dan *port* 2323 yang merupakan *port SSH*, dimana penyerangan ini bertujuan untuk memperoleh *username* dan *password* pada *server*.

Pertama yang dilakukan adalah mencoba membobol akses *port* 22 *SSH* yang telah dimanipulasi menjadi *server honeypot* menggunakan metode *brute force attack*. Perintah yang pertama digunakan dalam penyerangan ini adalah :

```
hydra -l kominfo -P /home/aldo/Documents/password.txt
192.168.1.13 ssh
```

*"sudo hydra"* berfungsi untuk menjalankan *tool hydra* pada *linux, "-s 22 -l kominfo -P /home/aldo/Documents/password.txt 192.168.1.13 ssh" "-s"* merupakan *port* dari *service* yang berjalan sebagai contoh ini menggunakan *default port SSH* 22, *"l"* disini diisi dengan *username* penyerang, berikutnya *"-P"* berisi file *wordlist* dan disini peneliti menggunakan file *wordlist* buatan sendiri, selanjutnya adalah *"/home/aldo/Documents/password.txt"* merupakan dimana

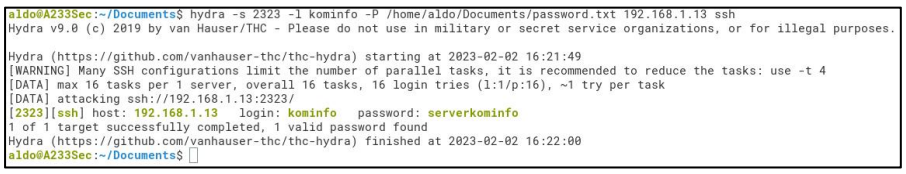
*path* tempat dimana menyimpan *wordlist* yang telah dibuat, terakhir adalah *"192.168.1.13 ssh"* merupakan alamat *IP* target yang akan dilakukan penyerangan *brute force attack.* Gambar 5.19 proses penyerangan pada *port* 22 menggunakan *brute force*:




Sumber : Peneliti.

**Gambar 5.19 Proses Penyerangan Pada *Port* 22 Menggunakan *Brute force***

Setelah melakukan penyerangan *brute force* terhadap *port* 22 *SSH* yang telah diubah menjadi *honeypot* didapatkan serangan menuju *port honeypot* tidak berhasil karena penyerang telah memasuki *termit* dari *honeypot* yang mengulangin proses akses ke *port* 22 yang mengakibatkan *Timeout connecting*. Selanjutnya melakukan serangan *brute force* terhadap *port SSH server* yang asli yaitu *port* 2323 dimana *port server* tersebut belum menggunakan metode *port knocking*. Penyerang akan melakukan serangan *brute force* menggunakan *wordlist* yang sudah diisi dengan *password server* yang asli. Gambar 5.20 proses penyerangan pada *port* 2323 menggunakan *brute force attack*:




Sumber : Peneliti.

**Gambar 5.20 Proses Penyerangan Pada *Port* 2323 Menggunakan *Brute force***

Pada gambar 5.20 ketika melakukan penyerangan terhadap *server SSH port* 2323 sebelum penerapan *port knocking* dengan mencoba serangan metode *brute force attack* yang menggunakan *wordlist* dan begitu juga untuk *port* 21 *FTP* seperti diatas ketika serangan *brute force* akan berhasil mendapatkan mendapatkan bahwa *username* dan *password server* berhasil ditemukan dan itu menjadi sangat bahaya ketika penyerang menemukan *username* dan *password server* yang sesuai dengan *wordlist* yang dibuat penyerang karena *port-port* tersebut karena tidak terlindungin *port knocking* jadi *port* tersebut bisa diserang dengan menggunakan *brute force.*

Selanjutnya untuk serangan *DoS attack* pada *server* belum bisa diantisipasi dikarenakan *port* pada *server* belum terlindungin *iptables* yang berakibatkan terkena serangan *DoS attack*, dikarenakan penyerang menyerang *DoS attack* dengan mengirimkan *fake traffic* untuk melumpuhkan suatu sistem jaringan yang terhubung ke *internet* seperti *port* pada *website* dengan membanjiri *server* dengan jumlah lalu lintas data yang tinggi, atau melakukan *request* data ke sebuah *server* sehingga *server* tidak lagi dapat memberikan layanan dan

menjadi *crash*. Tabel 5.2 hasil tabel pengujian sesudah implementasi keamanan *honeypot* :

**Tabel 5.2 Hasil Tabel Pengujian Sesudah Implementasi Keamanan *Honeypot***

| Serangan | Port | CPU | Memory | Keterangan | |
|---|---|---|---|---|---|
| | | | | Berhasil | Gagal |
| *Denial of Service* | 21 | 0,4 sy | 379.4 | | ✓ |
| | 22 | 0,3 sy | 383,6 | | ✓ |
| | 80 | 88,9 sy | 123.7 | ✓ | |
| | 443 | 81,3 sy | 128.9 | ✓ | |
| | 3306 | 75,4 sy | 138.7 | ✓ | |
| | 2323 | 0,1 sy | 394.2 | | ✓ |
| *Bruteforce* | 21 | 5,1 sy | 183.6 | ✓ | |
| | 22 | 2,7 sy | 174,7 | ✓ | |
| | 80 | 2,6 sy | 171,9 | | ✓ |
| | 443 | 2,1 sy | 170,1 | | ✓ |
| | 3306 | 3,3 sy | 179,3 | | ✓ |
| | 2323 | 0,8 sy | 360,8 | ✓ | |

Berdasarkan hasil pengujian penyerangan sesudah implementasi keamanan *honeypot* yang menuju *port* 21, 22, 80, 443, 3306 dan 2323 pada tabel 5.2, pada penyerangan *Denial of Service* menuju port 80, 443 dan 3306 menyebabkan peningkatan performa *CPU* dan memori pada *server* yang membuat *server* banyak dibanjiri *request* atau *fake traffic* yang membuat *server* menjadi *down*, sedangkan *port* 21, 22 dan 2323 performa *CPU* dan memori tetap stabil. Sedangkan pada penyerangan menggunakan *brute force* menuju *port* 21 *FTP* dan *port* 2323 *SSH* berhasil dilakukan penyerang dan penyerang bisa untuk mengakses dan mengeksploitasi *server* dan pada *port* 22 penyerang belum berhasil masuk *server* dikarenakan *port* tersebut adalah *port honeypot server*.

**b) *Port Knocking***

Pada tahap ini setelah selesai implementasi, konfigurasi dan pengujian *honeypot* selanjutnya peneliti melakukan pengujian terhadap *port knocking* yang telah terpasang *honeypot*. Tahap yang pertama adalah peneliti akan melakukan *port scanning* pada *server*. Gambar 5.21 pengujian *port scanning server* setelah melakukan installasi dan konfigurasi *honeypot* dan *port knocking* :

```
aldo@A233Sec:~/Documents$ sudo nmap 192.168.1.13
Starting Nmap 7.80 ( https://nmap.org ) at 2023-02-12 10:15 WIB
Nmap scan report for 192.168.1.13 (192.168.1.13)
Host is up (0.00077s latency).
Not shown: 996 filtered ports
PORT     STATE SERVICE
22/tcp   open  ssh
80/tcp   open  http
443/tcp  open  https
3306/tcp open  mysql
MAC Address: 40:9F:38:EB:04:4D (AzureWave Technology)

Nmap done: 1 IP address (1 host up) scanned in 5.23 seconds
```

Sumber : Peneliti.

**Gambar 5.21 Pengujian *Port Scanning Server* Setelah Melakukan Installasi dan Konfigurasi *Honeypot* dan *Port Knocking***

Pada gambar 5.21 peneliti melakukan serangan *port scanning* pada *server,* hasil dari serangan *port scanning* tersebut adalah *port* yang terbuka 22, 80, 443 dan 3306 untuk *port* 21 *FTP* dan 2323 *SSH* tidak terdeteksi *NMAP* dikarenakan *port* tersebut telah diblok *firewall*. Gambar 5.22 proses akses *port* 21 yang telah terlindungi *port knocking* dan *brute force attack* pada *port* 21 *FTP*:

```
aldo@A233Sec:~/Documents$ ftp 192.168.1.13
ftp: connect: Connection refused
ftp> exit
aldo@A233Sec:~/Documents$ sudo hydra -L /home/aldo/Documents/username.txt -P /home/aldo/Documents/password.txt 192.168.1.13 ftp
Hydra v9.0 (c) 2019 by van Hauser/THC - Please do not use in military or secret service organizations, or for illegal purposes.

Hydra (https://github.com/vanhauser-thc/thc-hydra) starting at 2023-02-12 10:16:23
[DATA] max 16 tasks per 1 server, overall 16 tasks, 256 login tries (1:16/p:16), ~16 tries per task
[DATA] attacking ftp://192.168.1.13:21/
[STATUS] 55.00 tries/min, 55 tries in 00:01h, 234 to do in 00:05h, 16 active
[STATUS] 48.33 tries/min, 145 tries in 00:03h, 144 to do in 00:03h, 16 active
[STATUS] 46.00 tries/min, 184 tries in 00:04h, 105 to do in 00:03h, 16 active
[STATUS] 44.60 tries/min, 223 tries in 00:05h, 66 to do in 00:02h, 16 active
[STATUS] 43.50 tries/min, 261 tries in 00:06h, 28 to do in 00:01h, 16 active
[STATUS] 41.29 tries/min, 289 tries in 00:07h, 1 to do in 00:01h, 1 active
1 of 1 target completed, 0 valid passwords found
Hydra (https://github.com/vanhauser-thc/thc-hydra) finished at 2023-02-12 10:23:27
aldo@A233Sec:~/Documents$ knock -v 192.168.1.13 4241 3452 7424
hitting tcp 192.168.1.13:4241
hitting tcp 192.168.1.13:3452
hitting tcp 192.168.1.13:7424
aldo@A233Sec:~/Documents$ ftp 192.168.1.13
Connected to 192.168.1.13.
220 ProFTPD Server (Debian) [::ffff:192.168.1.13]
Name (192.168.1.13:aldo):
```

Sumber : Peneliti.

**Gambar 5.22 Proses Akses *Port* 21 Yang Telah Terlindungi *Port Knocking* dan *Brute Force Attack* Pada *Port* 21 *FTP***

Pada gambar 5.22 saat mengakses *port* 21 tidak bisa diakses dikarenakan *port* tersebut sudah terlindungi dengan *firewall* dan *port knocking*, jadi ketika penyerang akan mengakses *port* 21 *FTP* maka akan mendapatkan alert *"Connection refused"*.

Ketika penyerang akan melakukan serangan *brute force* terhadap *port* 21 *FTP* maka penyerang tidak mendapatkan hasil karena *port* 21 *FTP* tersebut sudah terlindungi, jika client ingin mengakses *port* 21 maka harus melakukan *port knocking* dengan urutan dan kode yang benar jika salah satu salah atau urutan tidak sesuai maka *firewall* tidak bisa membuka akses untuk *port* 21, jadi perintah yang dimasukan harus benar dan sesuai dengan konfigurasi.

Setelah melakukan pengujian terhadap *port* 21 *FTP* yang terlindungi *port knocking* selanjutnya pengujian *port* 2323 *SSH* server

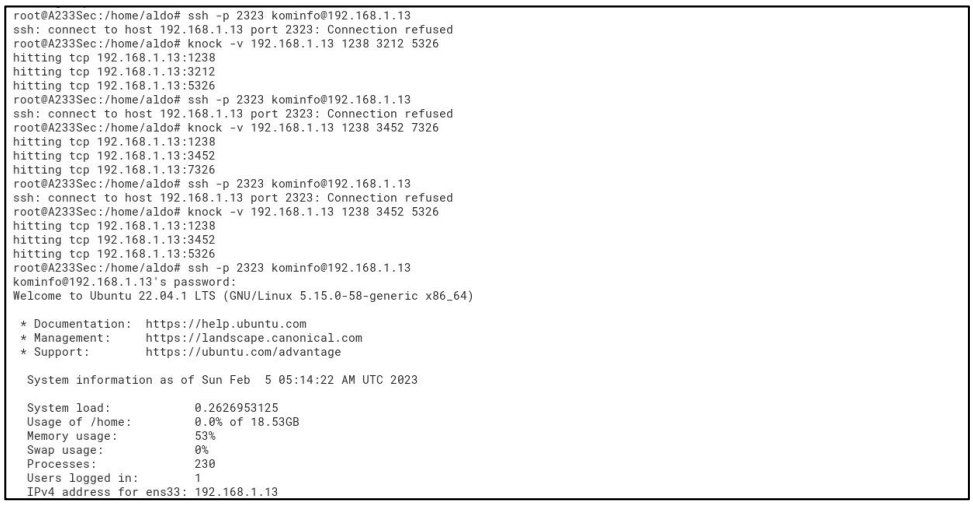
```
root@A233Sec:/home/aldo# ssh -p 2323 kominfo@192.168.1.13
ssh: connect to host 192.168.1.13 port 2323: Connection timed out
```

adalah *script command* untuk melakukan *port knocking* pada *server*. Gambar 5.23 Proses *login ssh* sebelum melakukan *port knocking* :

Sumber : Peneliti.

**Gambar 5.23 Proses *Login SSH* Sebelum Melakukan *Port Knocking***

Pada gambar diatas menjelaskan saat sebelum melakukan proses *port knocking*, pada saat *client* akan melakukan *login* pada *server* yang asli yang terdapat pada *port ssh* 2323, client akan tidak bisa mengakses *server* dikarenakan *server* tersebut telah terlindungin dengan *port knocking* yang memblok *port ssh* 2323 dengan menggunakan menggunakan *iptables* yang membuat akses *ssh* menjadi *connection timed out*. Jika *client* ingin mengakses *port ssh* 2323 *client* diharuskan melakukan installasi *knocked* pada komputer *client* dan melakukan *port knocking*. Berikut dapat dilihat program yang dijalankan untuk mengakses *port ssh* 2323 dengan *port knocking* dapat menggunakan *script command* sebagai berikut. Gambar 5.24 Proses *login ssh* menggunakan *port knocking* :




Sumber : Peneliti.

**Gambar 5.24 Proses *Login SSH* Sesudah Melakukan *Port Knocking***

Proses dimana *client* mencoba untuk terhubung dengan *SSH server* yang telah diketuk pada gambar diatas menggunakan *username*

*server 'kominfo'* dan *IP Address* 192.168.1.13 dengan tujuan *port ssh* yaitu *port* 2323. jika *client* ingin mengakses *port* 2323 maka harus melakukan *port knocking* dengan urutan dan kode yang benar jika salah satu salah atau urutan tidak sesuai maka *iptables* tidak bisa membuka akses untuk port 2323, jadi perintah yang dimasukan harus benar. Apabila *client* mencoba untuk terhubung dengan *SSH server* tanpa menggunakan ketukan maka diberikan hasil *"connection time out"*. Selanjutnya peneliti akan melakukan uji coba *brute force attack* terhadap *port* 2323 *SSH*. Gambar 5.25 pengujian *brute force attack* pada *port* 2323 *SSH* yang terlindungi *port knocking* :

Sumber : Peneliti.

**Gambar 5.25 Pengujian *Brute Force Attack* Pada *Port* 2323 *SSH* Yang Terlindungi *Port Knocking***

Pada gambar 5.25 peneliti melakukan serangan *brute force attack* terhadap *server* pada *port* 2323 *SSH* untuk akses *server*, dari serangan *brute force* tersebut didapatkan bahwa hasilnya adalah serangan *brute force* tidak berhasil karena *port* tersebut dilindungin *port knocking*. Selanjutnya peneliti melakukan pengamatan hasil dari pengujian *honeypot* dan *port knocking*. Tabel 5.3 hasil pengujian *honeypot* dan *port knocking*:

**Tabel 5.3 Hasil Pengujian *Honeypot* dan *Port knocking***

| Serangan | Port | CPU | Memory | Keterangan | |
|---|---|---|---|---|---|
| | | | | Berhasil | Gagal |
| *Denial of Service* | 21 | 0,8 sy | 364.2 | | ✓ |
| | 22 | 0,5 sy | 353,6 | | ✓ |
| | 80 | 82,9 sy | 143.1 | ✓ | |
| | 443 | 79,8 sy | 142,9 | ✓ | |
| | 3306 | 82,8 sy | 143,0 | ✓ | |
| | 2323 | 0,2 sy | 391.8 | | ✓ |
| *Bruteforce* | 21 | 2,8 sy | 239.9 | | ✓ |
| | 22 | 2,3 sy | 234,4 | ✓ | |
| | 80 | 3,6 sy | 182,6 | | ✓ |
| | 443 | 3,2 sy | 182,9 | | ✓ |
| | 3306 | 4,8 sy | 191,7 | | ✓ |
| | 2323 | 1,2 sy | 310,3 | | ✓ |

Berdasarkan hasil pengujian penyerangan menggunakan metode *port knocking* dan *honeypot* menuju port 21, 22, 80, 443, 3306 dan 2323 pada tabel 5.3, pada penyerangan *Denial of Service* menuju port 80, 443, 3306 menyebabkan peningkatan performa *CPU* dan memori pada *server* yang membuat *server* banyak dibanjiri *request* atau *fake traffic* yang membuat *server* menjadi *down*, sedangkan *port* 21, 22 dan 2323 performa *CPU* dan memori tetap stabil. Sedangkan pada penyerangan menggunakan *brute force* menuju *port* 22 berhasil dilakukan penyerang sedangkan *port* 21 dan 2323 gagal dikarenakan dilindungin *port knocking*.

Selanjutnya dilakukan pengamatan hasil implementasi pada *port knocking* untuk melihat dan memastikan konfigurasi berjalan sesuai dengan harapan dengan cara mencoba skema yang telah ditentukan. Tabel 5.4 tabel uji coba skema *port knocking* :

**Tabel 5.4 Tabel Uji Coba Skema *Port Knocking***

| Port Knocking | Masuk | Port |
|---|---|---|

| 1238 | 3452 | 5326 | dalam allow acces | 21 | 22 | 80 | 443 | 3306 | 2323 |
|---|---|---|---|---|---|---|---|---|---|
| x | x | x | x | x | ✓ | ✓ | ✓ | ✓ | x |
| x | ✓ | ✓ | x | x | ✓ | ✓ | ✓ | ✓ | x |
| ✓ | x | ✓ | x | x | ✓ | ✓ | ✓ | ✓ | x |
| ✓ | ✓ | x | x | x | ✓ | ✓ | ✓ | ✓ | x |
| ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ | ✓ |

**c) *Iptables***

Selanjutnya peneliti melakukan pengujian terhadap *iptables* yang telah terpasang *honeypot* dan *port knocking*. Tahap yang pertama adalah peneliti akan melakukan *port scanning* pada *server*. Gambar 5.26 pengujian *port scanning server* setelah melakukan installasi dan konfigurasi *honeypot, port knocking* dan *iptables* :

```
root@A233Sec:/home/aldo/Documents# nmap 192.168.1.13
Starting Nmap 7.80 ( https://nmap.org ) at 2023-02-06 02:42 WIB
Note: Host seems down. If it is really up, but blocking our ping probes, try -Pn
Nmap done: 1 IP address (0 hosts up) scanned in 10.99 seconds
root@A233Sec:/home/aldo/Documents#
```

Sumber : Peneliti.

**Gambar 5.26 Pengujian *Port Scanning Server* Setelah Melakukan Installasi dan Konfigurasi *Honeypot*, *Port Knocking* dan *Iptables***

Pada gambar 5.26 setelah melakukan penerapan *honeypot*, *port knocking* dan *iptables* dimana hasil dari *port scanning* yang dilakukan menggunakan *NMAP* berhasil di *block* oleh *iptables* sehingga tidak menampilkan *port* yang terbuka atau *port* yang aktif pada *server.*

Setelah melakukan konfigurasi *iptables* selanjutnya dilakukan uji coba penyerangan *Denial of Service (DoS)* terhadap *server* dengan menggunakan *tool torshammer*. Gambar 5.27 penyerangan *DoS* setelah diterapkan *iptables* :

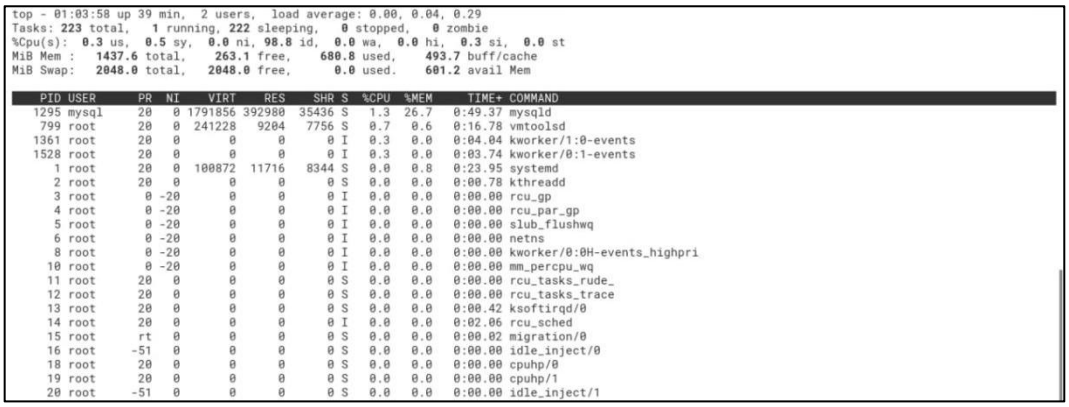
```
root@A233Sec:/home/aldo/Tools/torshammer# python torshammer.py -t 192.168.1.13  -p 80 -r 5000

/*
Connected to host...
Posting: w
Connected to host...
Posting: 2
Connected to host...
 onnected to host...
Posting: 5
Connected to host...
Posting: R
Connected to host...
Connected to host...
Posting: V
Connected to host...
Thread broken, restarting...
Thread broken, restarting...
Traceback (most recent call last):
```

Sumber : Peneliti.

**Gambar 5.27 Penyerangan *DoS* Setelah Diterapkan *Iptables***

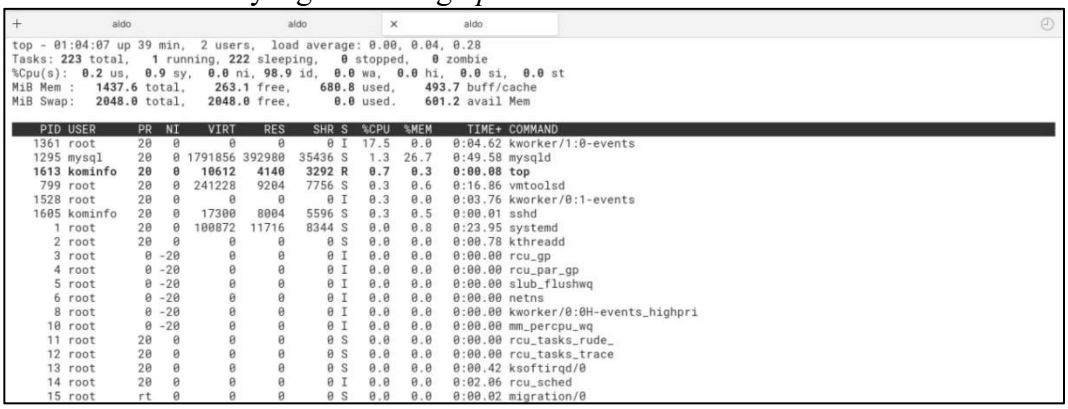
Pada gambar 5.27 Menunjukkan penyerangan menggunakan *tool torshammer* dimana *IP Address* yang dituju adalah "192.168.1.13" dengan tujuan *port* 80 *method TCP* dan *thread* yang digunakan sebanyak 5000. Penyerangan akan dilakukan dalam waktu 83 menit dan akan dilihat penggunaan *CPU* dan Memori pada *server* yang telah dilindungi oleh *firewall iptables*. Gambar 5.28 proses memanajemen yang sedang berjalan sebelum penyerangan *DoS* pada *server* yang terlindungi *iptables* :




Sumber : Peneliti.

**Gambar 5.28 Proses Memanajemen Yang Sedang Berjalan Sebelum Penyerangan *DoS* Pada *Server* Yang Terlindungi *Iptables***

Pada gambar 5.28 terlihat bahwa sebelum penyerangan *DoS atttack* proses manajemen yang sedang berjalan menggunakan *iptables* sebelum penyerangan *DoS attack* menujukkan hasil proses dari perintah *TOP* pada *server* jumlah waktu yang dihabiskan dalam ruang *kernel (sy)* pada *CPU* sebanyak *(0.3 sy)*. Gambar 5.29 proses memanajemen yang sedang berjalan sesudah penyerangan *DoS* pada *server* yang terlindungi *iptables*:




Sumber : Peneliti.

**Gambar 5.29 Proses Memanajemen Yang Sedang Berjalan Sesudah Penyerangan *DoS* Pada *Server* Yang Terlindungi *Iptables***

Pada gambar 5.29 terlihat setelah penyerangan *DoS attack* pada *server* yang terlindingi *honeypot*, *port knocking* dan *iptables* proses dari *CPU* dan memori *server* yang berjalan tetap stabil. Selanjutnya peneliti melakukan pengamatan hasil dari pengujian *honeypot, port knocking* dan *Iptables*. Tabel 5.5 hasil pengujian *honeypot, port knocking* dan *Iptables*:

**Tabel 5.5 Hasil Pengujian *Honeypot , Port knocking* dan *Iptables***

| Serangan | Port | CPU | Memory | Keterangan | |
|---|---|---|---|---|---|
| | | | | Berhasil | Gagal |
| *Denial of Service* | 21 | 0,3 sy | 263,1 | | ✓ |
| | 22 | 0,9 sy | 253,3 | | ✓ |
| | 80 | 0,7 sy | 242,2 | | ✓ |
| | 443 | 0,4 sy | 258,9 | | ✓ |
| | 3306 | 0,2 sy | 263,5 | | ✓ |
| | 2323 | 0,2 sy | 265,2 | | ✓ |
| *Bruteforce* | 21 | 0,5 sy | 261,2 | | ✓ |
| | 22 | 2,6 sy | 182,7 | ✓ | |
| | 80 | 1,6 sy | 192,8 | | ✓ |
| | 443 | 1,1 sy | 200,1 | | ✓ |
| | 3306 | 3,1 sy | 183,2 | | ✓ |
| | 2323 | 0,3 sy | 271,9 | | ✓ |

Berdasarkan hasil pengujian penyerangan menggunakan metode *port knocking, honeypot* menggunakan *iptables* menuju *port* 21, 22, 80, 443, 3306 dan 2323 pada tabel 5.5, pada uji coba serangan *Denial of Service* didapatkan hasilnya pada *server CPU* dan memori tetap stabil dan tidak mengalami peningkatan karena *server* terlindungi *iptables* dan untuk serangan *brute force* menuju *port* 21, 80, 443, 3306 dan 2323 terlindungi *iptables* sehingga jika penyerang menyerang *port* tersebut akan diblok *iptables* dan mendapatkan perintah *connection refused* karena *port* tersebut dilindungi *iptables* kecuali pada *port* 22 yang diubah menjadi *port* untuk *honeypot* yang tidak terlindungi *iptables*.

### 5.1.6. *Management*

Pada tahap *management*, dilakukan pembuatan aturan yang disepakati berdasarkan hasil implementasi dan *monitoring* guna mengatur

penerapan *honeypot* dan *port knocking* yang sudah dikembangkan agar dapat berlangsung lama dan dapat memenuhi harapan.

Adapun aturan yang disepakati sebagai berikut :

a) Pihak yang dapat mengakses *server* adalah pihak yang memiliki akses *server* yaitu pihak Dinas Komunikasi dan Informatika Kota Palembang
b) Dilakukan pemantauan terhadap *server* secara berkala
c) Dilakukan perubahan skema *port knocking* secara berkala
d) Dilakukan *penetration testing* secara berkala untuk memastikan bahwa *server* masih dalam kondisi aman

# BAB VI

# PENUTUP

## 6.1. Kesimpulan

Berdasarkan uraian penelitian skripsi dengan penerapan *port knocking* dan *honeypot* pada ekosistem *server farm* Dinas Komunikasi dan Informatika Kota Palembang, diperoleh kesimpulan sebagai berikut :

1. Sistem pengamanan tersebut dapat mengurangi resiko serangan percobaan akses dan membuat penyerang yang tidak dapat mengakses ke *port* yang telah ditentukan.
2. Sistem pengamanan menggunakan *honeypot* yang dibuat menggunakan *SSH server* palsu untuk mengelabui penyerang sehingga penyerang tidak menyerang *SSH server* yang asli dan dapat mengalihkan penyerang *port service* kepada *fake port* yang dibuat *honeypot.*
3. Pengamanan *server* dengan menggunakan metode *port knocking* berjalan dengan baik, dengan memasukan kode *random* untuk otentikasi sebagai syarat untuk menggunakan *port service*, hanya pengguna yang diperbolehkan saja untuk mengakses *port*.
4. Penggunaan *iptables* dapat memblok *incoming packet* sehingga *port service* tidak akan terbuka, memblokir serangan seperti *DoS attack, brute force attack* dan *port scanning*.
5. Kelebihan dengan menggabungkan metode *port knocking*, *honeypot* dan *iptables* dapat memperkuat pertahanan keamanan jaringan *server* dan membantu dalam melindungi *server* dari serangan yang tidak diinginkan.

6. Kombinasi ketiga metode ini dapat membatasi koneksi yang diterima oleh *server*, memfilter paket, dan mengurangi kerentanan terhadap serangan jaringan.
7. Kekurangan dengan menggabungkan metode *port knocking*, *honeypot*, dan *iptables*, perlu diperhatikan juga bahwa pengaturan yang tidak tepat atau kegagalan sistem dapat mempengaruhi efektivitas keamanan *server*. Jika salah satu komponen dari sistem tidak dikonfigurasi dengan benar atau terbuka, maka dapat memungkinkan *vulnerability* pada *server*.

## 6.2. Saran

Berdasarkan uraian dan kesimpulan yang telah dijelas dalam skripsi mengenai penerapan *Port Knocking* dan *Honeypot* pada ekosistem *server farm* Dinas Komunikasi dan Informatika Kota Palembang, penulis memberikan beberapa saran sebagai berikut :

1. Harus dilakukan pengujian *penetration testing* secara berkala dengan metode yang berbeda untuk mencari kekurangan dan kelemahan dalam sistem.
2. Diharapkan untuk penerapan keamanan *honeynet* yang merupakan *honeypot* yang terdiri dari beberapa jaringan *honeypot.*
3. Diharapakan kedepannya untuk menerapkan keamanan menggunakan *high interaction honeypot*.

# DAFTAR PUSTAKA


Alwi, E. I., Herdianti, H., & Umar, F. (2020). Analisis Keamanan Website Menggunakan Teknik Footprinting dan Vulnerability Scanning. *INFORMAL: Informatics Journal*, *5*(2), 43.

Dawamsyach, F., Ruslianto, I., & Ristian, U. (2023). CESS Implementasi IPS ( Intrusion Prevention System ) Fail2ban Pada Server Terhadap Serangan DDoS dan Brute Force Implementation of IPS ( Intrusion Prevention System) Fail2ban on Server for DDoS and Brute Force Attacks. 8(January), (Journal of Computing Engineering, System and Science) 8(1) January 2023 149-161

Dwiyatno, S., Rachmat, E., Sari, A. P., & Gustiawan, O. (2020). Implementasi Virtualisasi Server Berbasis Docker Container. Jurnal Pengembangan Riset Dan Observasi Sistem Komputer, 7(2), 165–175.

Ernawati, R., Ruslianto, I., Bahri, S., Rekayasa, J., & Komputer, S. (2022). Implementasi Metode Port Knocking Pada Sistem Keamanan Server Ubuntu Virtual Berbasis Web Monitoring. Jurnal Komputer dan Aplikasi 10(01), 158–169.

Hassan, R. H., Juli, S., Ismail, I., Terapan, F. I., Telkom, U., Shell, S., & Server, W. (2020). Implementasi Honeypot Dengan Metode Honeytrap. Jurnal e-Proceeding of Applied Science 6(2).

Mardiansyah, A. Z., Abdussyakur, Y. M., & Jatmika, A. H. (2021). Optimasi Port Knocking dan Honeypot Menggunakan Iptables Sebagai Keamanan Jaringan Pada Server. Security. Jurnal Teknologi Informasi Komputer dan Aplikasinya 3(2).

Muslim, B. (2020). Workshop Instalisasi Sistem Operasi Bagi Operator Dapodik Di Dinas Pendidikan Kec. Dempo Utara. Jurnal Pengabdian Kepada Masyarakat Ngabdimas, 3(1), 1–6.

Purba, W. W., & Efendi, R. (2021). Perancangan dan analisis sistem keamanan jaringan komputer menggunakan SNORT. Jurnal Teknologi Informasi Aiti, 17(2), 143–158.

Rahayu Nugraheni Rachmawati, T. C. (2022). Rancang Bangun Dan Pemanfaatan Mikrotik Dalam Jaringan Rt Rw Net. *Jurnal Publikasi Ilmu Komputer Dan Multimedia*, *1*(1), 31–42.

Santoso, D., Noertjahyana, A., & Andjarwirawan, J. (2022). Implementasi dan Analisa Snort dan Suricata Sebagai IDS dan IPS Untuk Mencegah Serangan DOS dan DDOS. Jurnal Infra, 10(1), 1–6.

Suharto, A., & Irfan. (2019). Analisa dan Perancangan Sistem Jaringan Berbasis VLAN Dengan Metode NDLC pada SMK Boedi Luhur. Jurnal Teknologi Informasi ESIT, 14(3), 42–48.

Sutanti, A., MZ, M. K., Mustika, M., & Damayanti, P. (2020). Rancang Bangun Aplikasi Perpustakaan Keliling Menggunakan Pendekatan Terstruktur. Jurnal Ilmiah Komputer Dan Informatika, 9(1), 1–8.

Tohirin, T. (2020). Penerapan Keamanan Remote Server Melalui Ssh Dengan Kombinasi Kriptografi Asimetris Dan Autentikasi Dua Langkah. Jurnal Teknologi Informasi, 4(1), 133–138.

Tujni, B., & Alfiansyah, A. H. (n.d.). Seminar Hasil Penelitian Vokasi (SEMHAVOK) Perencanaan Pemetaan IP Address Menggunakan Metode VLSM di PT KAI Divre III Palembang Sumatera Selatan (Simulasi Dengan Cisco Packet Tracer).

## *Listing Code*

### *Port Knocking*

```
[options]
UseSyslog

[openSSH]
sequence    = 1238, 3425, 5326
seq_timeout = 5
command     = /sbin/iptables -I INPUT -s %IP% -p tcp --dport 2323-j ACCEPT
tcpflags    = syn

[openFTP]
sequence    = 4241, 3425, 5326
seq_timeout = 5
command     = /sbin/iptables -I INPUT -s %IP% -p tcp --dport 2323-j ACCEPT
tcpflags    = syn


[closeSSH]
sequence    = 5326, 3425, 1238
seq_timeout = 5
command     = /sbin/iptables -D INPUT -s %IP% -p tcp --dport 2323-j ACCEPT
tcpflags    = syn

[closeFTP]
sequence    = 7424, 3452, 4241
seq_timeout = 5
command     = /sbin/iptables -D INPUT -s %IP% -p tcp --dport 2323-j ACCEPT
tcpflags    = syn
```

***Honeypot***

```
/* Endlessh: an SSH tarpit
 *
 * This is free and unencumbered software released into the public domain.
 */
#if defined(__OpenBSD__)
#  define _BSD_SOURCE  /* for pledge(2) and unveil(2) */
#else
#  define _XOPEN_SOURCE 600
#endif

#include <time.h>
#include <errno.h>
#include <stdio.h>
#include <limits.h>
#include <signal.h>
#include <stdarg.h>
#include <stdlib.h>
#include <string.h>

#include <poll.h>
#include <fcntl.h>
#include <unistd.h>
#include <sys/types.h>
#include <sys/socket.h>
#include <arpa/inet.h>
#include <netinet/in.h>
#include <syslog.h>

#define ENDLESSH_VERSION          1.1

#define DEFAULT_PORT              2222
#define DEFAULT_DELAY             10000  /* milliseconds */
#define DEFAULT_MAX_LINE_LENGTH     32
#define DEFAULT_MAX_CLIENTS       4096

#if defined(__FreeBSD__)
#  define DEFAULT_CONFIG_FILE "/usr/local/etc/endlessh.config"
#else
#  define DEFAULT_CONFIG_FILE "/etc/endlessh/config"
#endif

#define DEFAULT_BIND_FAMILY  AF_UNSPEC

#define XSTR(s) STR(s)
```

```
#define STR(s) #s

static long long
epochms(void)
{
   struct timespec tv;
   clock_gettime(CLOCK_REALTIME, &tv);
   return tv.tv_sec * 1000ULL + tv.tv_nsec / 1000000ULL;
}

static enum loglevel {
   log_none,
   log_info,
   log_debug
} loglevel = log_none;

static void (*logmsg)(enum loglevel level, const char *, ...);

static void
logstdio(enum loglevel level, const char *format, ...)
{
   if (loglevel >= level) {
      int save = errno;

      /* Print a timestamp */
      long long now = epochms();
      time_t t = now / 1000;
      char date[64];
      struct tm tm[1];
      strftime(date, sizeof(date), "%Y-%m-%dT%H:%M:%S", gmtime_r(&t, tm));
      printf("%s.%03lldZ ", date, now % 1000);

      /* Print the rest of the log message */
      va_list ap;
      va_start(ap, format);
      vprintf(format, ap);
      va_end(ap);
      fputc('\n', stdout);

      errno = save;
   }
}

static void
logsyslog(enum loglevel level, const char *format, ...)
{
```

```
    static const int prio_map[] = { LOG_NOTICE, LOG_INFO, LOG_DEBUG };

    if (loglevel >= level) {
        int save = errno;

        /* Output the log message */
        va_list ap;
        va_start(ap, format);
        char buf[256];
        vsnprintf(buf, sizeof buf, format, ap);
        va_end(ap);
        syslog(prio_map[level], "%s", buf);

        errno = save;
    }
}

static struct {
    long long connects;
    long long milliseconds;
    long long bytes_sent;
} statistics;

struct client {
    char ipaddr[INET6_ADDRSTRLEN];
    long long connect_time;
    long long send_next;
    long long bytes_sent;
    struct client *next;
    int port;
    int fd;
};

static struct client *
client_new(int fd, long long send_next)
{
    struct client *c = malloc(sizeof(*c));
    if (c) {
        c->ipaddr[0] = 0;
        c->connect_time = epochms();
        c->send_next = send_next;
        c->bytes_sent = 0;
        c->next = 0;
        c->fd = fd;
        c->port = 0;
```

```
        /* Set the smallest possible recieve buffer. This reduces local
         * resource usage and slows down the remote end.
         */
        int value = 1;
        int r = setsockopt(fd, SOL_SOCKET, SO_RCVBUF, &value, sizeof(value));
        logmsg(log_debug, "setsockopt(%d, SO_RCVBUF, %d) = %d", fd, value, r);
        if (r == -1)
            logmsg(log_debug, "errno = %d, %s", errno, strerror(errno));

        /* Get IP address */
        struct sockaddr_storage addr;
        socklen_t len = sizeof(addr);
        if (getpeername(fd, (struct sockaddr *)&addr, &len) != -1) {
            if (addr.ss_family == AF_INET) {
                struct sockaddr_in *s = (struct sockaddr_in *)&addr;
                c->port = ntohs(s->sin_port);
                inet_ntop(AF_INET, &s->sin_addr,
                        c->ipaddr, sizeof(c->ipaddr));
            } else {
                struct sockaddr_in6 *s = (struct sockaddr_in6 *)&addr;
                c->port = ntohs(s->sin6_port);
                inet_ntop(AF_INET6, &s->sin6_addr,
                        c->ipaddr, sizeof(c->ipaddr));
            }
        }
    }
    return c;
}

static void
client_destroy(struct client *client)
{
    logmsg(log_debug, "close(%d)", client->fd);
    long long dt = epochms() - client->connect_time;
    logmsg(log_info,
            "CLOSE host=%s port=%d fd=%d "
            "time=%lld.%03lld bytes=%lld",
            client->ipaddr, client->port, client->fd,
            dt / 1000, dt % 1000,
            client->bytes_sent);
    statistics.milliseconds += dt;
    close(client->fd);
    free(client);
}

static void
```

```
statistics_log_totals(struct client *clients)
{
   long long milliseconds = statistics.milliseconds;
   for (long long now = epochms(); clients; clients = clients->next)
      milliseconds += now - clients->connect_time;
   logmsg(log_info, "TOTALS connects=%lld seconds=%lld.%03lld bytes=%lld",
         statistics.connects,
         milliseconds / 1000,
         milliseconds % 1000,
         statistics.bytes_sent);
}

struct fifo {
   struct client *head;
   struct client *tail;
   int length;
};

static void
fifo_init(struct fifo *q)
{
   q->head = q->tail = 0;
   q->length = 0;
}

static struct client *
fifo_pop(struct fifo *q)
{
   struct client *removed = q->head;
   q->head = q->head->next;
   removed->next = 0;
   if (!--q->length)
      q->tail = 0;
   return removed;
}

static void
fifo_append(struct fifo *q, struct client *c)
{
   if (!q->tail) {
      q->head = q->tail = c;
   } else {
      q->tail->next = c;
      q->tail = c;
   }
   q->length++;
```

```
}

static void
fifo_destroy(struct fifo *q)
{
    struct client *c = q->head;
    while (c) {
        struct client *dead = c;
        c = c->next;
        client_destroy(dead);
    }
    q->head = q->tail = 0;
    q->length = 0;
}

static void
die(void)
{
    fprintf(stderr, "endlessh: fatal: %s\n", strerror(errno));
    exit(EXIT_FAILURE);
}

static unsigned
rand16(unsigned long s[1])
{
    s[0] = s[0] * 1103515245UL + 12345UL;
    return (s[0] >> 16) & 0xffff;
}

static int
randline(char *line, int maxlen, unsigned long s[1])
{
    int len = 3 + rand16(s) % (maxlen - 2);
    for (int i = 0; i < len - 2; i++)
        line[i] = 32 + rand16(s) % 95;
    line[len - 2] = 13;
    line[len - 1] = 10;
    if (memcmp(line, "SSH-", 4) == 0)
        line[0] = 'X';
    return len;
}

static volatile sig_atomic_t running = 1;

static void
sigterm_handler(int signal)
```

```
{
   (void)signal;
   running = 0;
}

static volatile sig_atomic_t reload = 0;

static void
sighup_handler(int signal)
{
   (void)signal;
   reload = 1;
}

static volatile sig_atomic_t dumpstats = 0;

static void
sigusr1_handler(int signal)
{
   (void)signal;
   dumpstats = 1;
}

struct config {
   int port;
   int delay;
   int max_line_length;
   int max_clients;
   int bind_family;
};

#define CONFIG_DEFAULT { \
   .port           = DEFAULT_PORT, \
   .delay          = DEFAULT_DELAY, \
   .max_line_length = DEFAULT_MAX_LINE_LENGTH, \
   .max_clients    = DEFAULT_MAX_CLIENTS, \
   .bind_family    = DEFAULT_BIND_FAMILY, \
}

static void
config_set_port(struct config *c, const char *s, int hardfail)
{
   errno = 0;
   char *end;
   long tmp = strtol(s, &end, 10);
   if (errno || *end || tmp < 1 || tmp > 65535) {
```

```
        fprintf(stderr, "endlessh: Invalid port: %s\n", s);
        if (hardfail)
            exit(EXIT_FAILURE);
    } else {
        c->port = tmp;
    }
}

static void
config_set_delay(struct config *c, const char *s, int hardfail)
{
    errno = 0;
    char *end;
    long tmp = strtol(s, &end, 10);
    if (errno || *end || tmp < 1 || tmp > INT_MAX) {
        fprintf(stderr, "endlessh: Invalid delay: %s\n", s);
        if (hardfail)
            exit(EXIT_FAILURE);
    } else {
        c->delay = tmp;
    }
}

static void
config_set_max_clients(struct config *c, const char *s, int hardfail)
{
    errno = 0;
    char *end;
    long tmp = strtol(s, &end, 10);
    if (errno || *end || tmp < 1 || tmp > INT_MAX) {
        fprintf(stderr, "endlessh: Invalid max clients: %s\n", s);
        if (hardfail)
            exit(EXIT_FAILURE);
    } else {
        c->max_clients = tmp;
    }
}

static void
config_set_max_line_length(struct config *c, const char *s, int hardfail)
{
    errno = 0;
    char *end;
    long tmp = strtol(s, &end, 10);
    if (errno || *end || tmp < 3 || tmp > 255) {
        fprintf(stderr, "endlessh: Invalid line length: %s\n", s);
```

```
        if (hardfail)
            exit(EXIT_FAILURE);
    } else {
        c->max_line_length = tmp;
    }
}

static void
config_set_bind_family(struct config *c, const char *s, int hardfail)
{
  switch (*s) {
      case '4':
          c->bind_family = AF_INET;
          break;
      case '6':
          c->bind_family = AF_INET6;
          break;
      case '0':
          c->bind_family = AF_UNSPEC;
          break;
      default:
          fprintf(stderr, "endlessh: Invalid address family: %s\n", s);
          if (hardfail)
              exit(EXIT_FAILURE);
          break;
  }
}

enum config_key {
    KEY_INVALID,
    KEY_PORT,
    KEY_DELAY,
    KEY_MAX_LINE_LENGTH,
    KEY_MAX_CLIENTS,
    KEY_LOG_LEVEL,
    KEY_BIND_FAMILY,
};

static enum config_key
config_key_parse(const char *tok)
{
    static const char *const table[] = {
        [KEY_PORT]           = "Port",
        [KEY_DELAY]          = "Delay",
        [KEY_MAX_LINE_LENGTH] = "MaxLineLength",
        [KEY_MAX_CLIENTS]    = "MaxClients",
```

```
        [KEY_LOG_LEVEL]       = "LogLevel",
        [KEY_BIND_FAMILY]     = "BindFamily"
    };
    for (size_t i = 1; i < sizeof(table) / sizeof(*table); i++)
        if (!strcmp(tok, table[i]))
            return i;
    return KEY_INVALID;
}

static void
config_load(struct config *c, const char *file, int hardfail)
{
    long lineno = 0;
    FILE *f = fopen(file, "r");
    if (f) {
        char line[256];
        while (fgets(line, sizeof(line), f)) {
            lineno++;

            /* Remove comments */
            char *comment = strchr(line, '#');
            if (comment)
                *comment = 0;

            /* Parse tokes on line */
            char *save = 0;
            char *tokens[3];
            int ntokens = 0;
            for (; ntokens < 3; ntokens++) {
                char *tok = strtok_r(ntokens ? 0 : line, " \r\n", &save);
                if (!tok)
                    break;
                tokens[ntokens] = tok;
            }

            switch (ntokens) {
                case 0: /* Empty line */
                    continue;
                case 1:
                    fprintf(stderr, "%s:%ld: Missing value\n", file, lineno);
                    if (hardfail) exit(EXIT_FAILURE);
                    continue;
                case 2: /* Expected */
                    break;
                case 3:
                    fprintf(stderr, "%s:%ld: Too many values\n", file, lineno);
```

```
            if (hardfail) exit(EXIT_FAILURE);
            continue;
        }

        enum config_key key = config_key_parse(tokens[0]);
        switch (key) {
          case KEY_INVALID:
            fprintf(stderr, "%s:%ld: Unknown option '%s'\n",
                    file, lineno, tokens[0]);
            break;
          case KEY_PORT:
            config_set_port(c, tokens[1], hardfail);
            break;
          case KEY_DELAY:
            config_set_delay(c, tokens[1], hardfail);
            break;
          case KEY_MAX_LINE_LENGTH:
            config_set_max_line_length(c, tokens[1], hardfail);
            break;
          case KEY_MAX_CLIENTS:
            config_set_max_clients(c, tokens[1], hardfail);
            break;
          case KEY_BIND_FAMILY:
            config_set_bind_family(c, tokens[1], hardfail);
            break;
          case KEY_LOG_LEVEL: {
            errno = 0;
            char *end;
            long v = strtol(tokens[1], &end, 10);
            if (errno || *end || v < log_none || v > log_debug) {
              fprintf(stderr, "%s:%ld: Invalid log level '%s'\n",
                      file, lineno, tokens[1]);
              if (hardfail) exit(EXIT_FAILURE);
            } else {
              loglevel = v;
            }
          } break;
        }
      }

      fclose(f);
    }
}

static void
config_log(const struct config *c)
```

```
{
    logmsg(log_info, "Port %d", c->port);
    logmsg(log_info, "Delay %d", c->delay);
    logmsg(log_info, "MaxLineLength %d", c->max_line_length);
    logmsg(log_info, "MaxClients %d", c->max_clients);
    logmsg(log_info, "BindFamily %s",
        c->bind_family == AF_INET6 ? "IPv6 Only" :
        c->bind_family == AF_INET  ? "IPv4 Only" :
                          "IPv4 Mapped IPv6");
}

static void
usage(FILE *f)
{
    fprintf(f, "Usage: endlessh [-vh] [-46] [-d MS] [-f CONFIG] [-l LEN] "
                          "[-m LIMIT] [-p PORT]\n");
    fprintf(f, "  -4        Bind to IPv4 only\n");
    fprintf(f, "  -6        Bind to IPv6 only\n");
    fprintf(f, "  -d INT    Message millisecond delay ["
          XSTR(DEFAULT_DELAY) "]\n");
    fprintf(f, "  -f        Set and load config file ["
          DEFAULT_CONFIG_FILE "]\n");
    fprintf(f, "  -h        Print this help message and exit\n");
    fprintf(f, "  -l INT    Maximum banner line length (3-255) ["
          XSTR(DEFAULT_MAX_LINE_LENGTH) "]\n");
    fprintf(f, "  -m INT    Maximum number of clients ["
          XSTR(DEFAULT_MAX_CLIENTS) "]\n");
    fprintf(f, "  -p INT    Listening port [" XSTR(DEFAULT_PORT) "]\n");
    fprintf(f, "  -v        Print diagnostics to standard output "
          "(repeatable)\n");
    fprintf(f, "  -V        Print version information and exit\n");
}

static void
print_version(void)
{
    puts("Endlessh " XSTR(ENDLESSH_VERSION));
}

static int
server_create(int port, int family)
{
    int r, s, value;

    s = socket(family == AF_UNSPEC ? AF_INET6 : family, SOCK_STREAM,
0);
```

```
    logmsg(log_debug, "socket() = %d", s);
    if (s == -1) die();

    /* Socket options are best effort, allowed to fail */
    value = 1;
    r = setsockopt(s, SOL_SOCKET, SO_REUSEADDR, &value, sizeof(value));
    logmsg(log_debug, "setsockopt(%d, SO_REUSEADDR, true) = %d", s, r);
    if (r == -1)
        logmsg(log_debug, "errno = %d, %s", errno, strerror(errno));

    /*
     * With OpenBSD IPv6 sockets are always IPv6-only, so the socket option
     * is read-only (not modifiable).
     * http://man.openbsd.org/ip6#IPV6_V6ONLY
     */
#ifndef __OpenBSD__
    if (family == AF_INET6 || family == AF_UNSPEC) {
        errno = 0;
        value = (family == AF_INET6);
        r = setsockopt(s, IPPROTO_IPV6, IPV6_V6ONLY, &value, sizeof(value));
        logmsg(log_debug, "setsockopt(%d, IPV6_V6ONLY, true) = %d", s, r);
        if (r == -1)
            logmsg(log_debug, "errno = %d, %s", errno, strerror(errno));
    }
#endif

    if (family == AF_INET) {
        struct sockaddr_in addr4 = {
            .sin_family = AF_INET,
            .sin_port = htons(port),
            .sin_addr = {INADDR_ANY}
        };
        r = bind(s, (void *)&addr4, sizeof(addr4));
    } else {
        struct sockaddr_in6 addr6 = {
            .sin6_family = AF_INET6,
            .sin6_port = htons(port),
            .sin6_addr = in6addr_any
        };
        r = bind(s, (void *)&addr6, sizeof(addr6));
    }
    logmsg(log_debug, "bind(%d, port=%d) = %d", s, port, r);
    if (r == -1) die();

    r = listen(s, INT_MAX);
    logmsg(log_debug, "listen(%d) = %d", s, r);
```

```
    if (r == -1) die();

    return s;
}

/* Write a line to a client, returning client if it's still up. */
static struct client *
sendline(struct client *client, int max_line_length, unsigned long *rng)
{
    char line[256];
    int len = randline(line, max_line_length, rng);
    for (;;) {
        ssize_t out = write(client->fd, line, len);
        logmsg(log_debug, "write(%d) = %d", client->fd, (int)out);
        if (out == -1) {
            if (errno == EINTR) {
                continue;      /* try again */
            } else if (errno == EAGAIN || errno == EWOULDBLOCK) {
                return client; /* don't care */
            } else {
                client_destroy(client);
                return 0;
            }
        } else {
            client->bytes_sent += out;
            statistics.bytes_sent += out;
            return client;
        }
    }
}


int
main(int argc, char **argv)
{
    logmsg = logstdio;
    struct config config = CONFIG_DEFAULT;
    const char *config_file = DEFAULT_CONFIG_FILE;

#if defined(__OpenBSD__)
    unveil(config_file, "r"); /* return ignored as the file may not exist */
    if (pledge("inet stdio rpath unveil", 0) == -1)
        die();
#endif

    config_load(&config, config_file, 1);
```

```
    int option;
    while ((option = getopt(argc, argv, "46d:f:hl:m:p:svV")) != -1) {
        switch (option) {
            case '4':
                config_set_bind_family(&config, "4", 1);
                break;
            case '6':
                config_set_bind_family(&config, "6", 1);
                break;
            case 'd':
                config_set_delay(&config, optarg, 1);
                break;
            case 'f':
                config_file = optarg;

#if defined(__OpenBSD__)
                unveil(config_file, "r");
                if (unveil(0, 0) == -1)
                    die();
#endif

                config_load(&config, optarg, 1);
                break;
            case 'h':
                usage(stdout);
                exit(EXIT_SUCCESS);
                break;
            case 'l':
                config_set_max_line_length(&config, optarg, 1);
                break;
            case 'm':
                config_set_max_clients(&config, optarg, 1);
                break;
            case 'p':
                config_set_port(&config, optarg, 1);
                break;
            case 's':
                logmsg = logsyslog;
                break;
            case 'v':
                if (loglevel < log_debug)
                    loglevel++;
                break;
            case 'V':
                print_version();
```

```
            exit(EXIT_SUCCESS);
            break;
        default:
            usage(stderr);
            exit(EXIT_FAILURE);
    }
}

if (argv[optind]) {
    fprintf(stderr, "endlessh: too many arguments\n");
    exit(EXIT_FAILURE);
}

if (logmsg == logsyslog) {
    /* Prepare the syslog */
    const char *prog = strrchr(argv[0], '/');
    prog = prog ? prog + 1 : argv[0];
    openlog(prog, LOG_PID, LOG_DAEMON);
} else {
    /* Set output (log) to line buffered */
    setvbuf(stdout, 0, _IOLBF, 0);
}

/* Log configuration */
config_log(&config);

/* Install the signal handlers */
signal(SIGPIPE, SIG_IGN);
{
    struct sigaction sa = {.sa_handler = sigterm_handler};
    int r = sigaction(SIGTERM, &sa, 0);
    if (r == -1)
        die();
}
{
    struct sigaction sa = {.sa_handler = sighup_handler};
    int r = sigaction(SIGHUP, &sa, 0);
    if (r == -1)
        die();
}
{
    struct sigaction sa = {.sa_handler = sigusr1_handler};
    int r = sigaction(SIGUSR1, &sa, 0);
    if (r == -1)
        die();
}
```

```
struct fifo fifo[1];
fifo_init(fifo);

unsigned long rng = epochms();

int server = server_create(config.port, config.bind_family);

while (running) {
   if (reload) {
      /* Configuration reload requested (SIGHUP) */
      int oldport = config.port;
      int oldfamily = config.bind_family;
      config_load(&config, config_file, 0);
      config_log(&config);
      if (oldport != config.port || oldfamily != config.bind_family) {
         close(server);
         server = server_create(config.port, config.bind_family);
      }
      reload = 0;
   }
   if (dumpstats) {
      /* print stats requested (SIGUSR1) */
      statistics_log_totals(fifo->head);
      dumpstats = 0;
   }

   /* Enqueue clients that are due for another message */
   int timeout = -1;
   long long now = epochms();
   while (fifo->head) {
      if (fifo->head->send_next <= now) {
         struct client *c = fifo_pop(fifo);
         if (sendline(c, config.max_line_length, &rng)) {
            c->send_next = now + config.delay;
            fifo_append(fifo, c);
         }
      } else {
         timeout = fifo->head->send_next - now;
         break;
      }
   }

   /* Wait for next event */
   struct pollfd fds = {server, POLLIN, 0};
   int nfds = fifo->length < config.max_clients;
```

```
logmsg(log_debug, "poll(%d, %d)", nfds, timeout);
int r = poll(&fds, nfds, timeout);
logmsg(log_debug, "= %d", r);
if (r == -1) {
   switch (errno) {
      case EINTR:
         logmsg(log_debug, "EINTR");
         continue;
      default:
         fprintf(stderr, "endlessh: fatal: %s\n", strerror(errno));
         exit(EXIT_FAILURE);
   }
}

/* Check for new incoming connections */
if (fds.revents & POLLIN) {
   int fd = accept(server, 0, 0);
   logmsg(log_debug, "accept() = %d", fd);
   statistics.connects++;
   if (fd == -1) {
      const char *msg = strerror(errno);
      switch (errno) {
         case EMFILE:
         case ENFILE:
            config.max_clients = fifo->length;
            logmsg(log_info,
                  "MaxClients %d",
                  fifo->length);
            break;
         case ECONNABORTED:
         case EINTR:
         case ENOBUFS:
         case ENOMEM:
         case EPROTO:
            fprintf(stderr, "endlessh: warning: %s\n", msg);
            break;
         default:
            fprintf(stderr, "endlessh: fatal: %s\n", msg);
            exit(EXIT_FAILURE);
      }
   } else {
      long long send_next = epochms() + config.delay;
      struct client *client = client_new(fd, send_next);
      int flags = fcntl(fd, F_GETFL, 0);      /* cannot fail */
      fcntl(fd, F_SETFL, flags | O_NONBLOCK); /* cannot fail */
      if (!client) {
```

```
                fprintf(stderr, "endlessh: warning: out of memory\n");
                close(fd);
            } else {
                fifo_append(fifo, client);
                logmsg(log_info, "ACCEPT host=%s port=%d fd=%d n=%d/%d",
                    client->ipaddr, client->port, client->fd,
                    fifo->length, config.max_clients);
            }
        }
    }
  }

  fifo_destroy(fifo);
  statistics_log_totals(0);

  if (logmsg == logsyslog)
    closelog();
}
```

***Iptables***

```
### 1: Drop paket yang tidak valid ###
/sbin/iptables -t mangle -A PREROUTING -m conntrack --ctstate INVALID -j
DROP

### 2: Drop paket TCP yang baru dan bukan SYN ###
/sbin/iptables -t mangle -A PREROUTING -p tcp ! --syn -m conntrack --ctstate
NEW -j DROP

### 3: Drop paket SYN dengan nilai MSS yang mencurigakan ###
/sbin/iptables -t mangle -A PREROUTING -p tcp -m conntrack --ctstate NEW -m
tcpmss ! --mss 536:65535 -j DROP

### 4: Block paket dengan flag TCP palsu ###
/sbin/iptables -t mangle -A PREROUTING -p tcp --tcp-flags FIN,SYN FIN,SYN
-j DROP
/sbin/iptables -t mangle -A PREROUTING -p tcp --tcp-flags SYN,RST SYN,RST
-j DROP
/sbin/iptables -t mangle -A PREROUTING -p tcp --tcp-flags FIN,RST FIN,RST -j
DROP
/sbin/iptables -t mangle -A PREROUTING -p tcp --tcp-flags FIN,ACK FIN -j
DROP
/sbin/iptables -t mangle -A PREROUTING -p tcp --tcp-flags ACK,URG URG -j
DROP
/sbin/iptables -t mangle -A PREROUTING -p tcp --tcp-flags ACK,PSH PSH -j
DROP
```

/sbin/iptables -t mangle -A PREROUTING -p tcp --tcp-flags ALL NONE -j DROP

### 5: Blokir paket palsu ###
/sbin/iptables -t mangle -A PREROUTING -s 224.0.0.0/3 -j DROP
/sbin/iptables -t mangle -A PREROUTING -s 169.254.0.0/16 -j DROP
/sbin/iptables -t mangle -A PREROUTING -s 172.16.0.0/12 -j DROP
/sbin/iptables -t mangle -A PREROUTING -s 192.0.2.0/24 -j DROP
/sbin/iptables -t mangle -A PREROUTING -s 192.168.0.0/16 -j DROP
/sbin/iptables -t mangle -A PREROUTING -s 10.0.0.0/8 -j DROP
/sbin/iptables -t mangle -A PREROUTING -s 0.0.0.0/8 -j DROP
/sbin/iptables -t mangle -A PREROUTING -s 240.0.0.0/5 -j DROP
/sbin/iptables -t mangle -A PREROUTING -s 127.0.0.0/8 ! -i lo -j DROP

### 6: Drop ICMP ###
/sbin/iptables -t mangle -A PREROUTING -p icmp -j DROP

### 7: Drop fragmen di semua chains ###
/sbin/iptables -t mangle -A PREROUTING -f -j DROP

### 8: Batasi koneksi per IP Address ###
/sbin/iptables -A INPUT -p tcp -m connlimit --connlimit-above 111 -j REJECT --reject-with tcp-reset

### 9: Batasi paket RST ###
/sbin/iptables -A INPUT -p tcp --tcp-flags RST RST -m limit --limit 2/s --limit-burst 2 -j ACCEPT
/sbin/iptables -A INPUT -p tcp --tcp-flags RST RST -j DROP

### 10: Batasi koneksi TCP baru per detik per IP Address ###
/sbin/iptables -A INPUT -p tcp -m conntrack --ctstate NEW -m limit --limit 60/s --limit-burst 20 -j ACCEPT
/sbin/iptables -A INPUT -p tcp -m conntrack --ctstate NEW -j DROP

### Perlindungan brute-force SSH ###
/sbin/iptables -A INPUT -p tcp --dport ssh -m conntrack --ctstate NEW -m recent --set
/sbin/iptables -A INPUT -p tcp --dport ssh -m conntrack --ctstate NEW -m recent --update --seconds 60 --hitcount 10 -j DROP

### Perlindungan terhadap port scanning ###
/sbin/iptables -N port-scanning
/sbin/iptables -A port-scanning -p tcp --tcp-flags SYN,ACK,FIN,RST RST -m limit --limit 1/s --limit-burst 2 -j RETURN
/sbin/iptables -A port-scanning -j DROP