ANALISIS DAN IMPLEMENTASI *SECURE SHELL* PADA *UBUNTU SERVER* DENGAN METODE *PORT KNOCKING*

SKRIPSI

Oleh :
Widy
130210015

PROGRAM STUDI TEKNIK INFORMATIKA
UNIVERSITAS PUTERA BATAM
2017

ANALISIS DAN IMPLEMENTASI *SECURE SHELL* PADA *UBUNTU SERVER* DENGAN METODE *PORT KNOCKING*

SKRIPSI

Untuk memenuhi salah satu syarat
guna memperoleh gelar Sarjana

Oleh :
Widy
130210015

PROGRAM STUDI TEKNIK INFORMATIKA
UNIVERSITAS PUTERA BATAM

2017

PERNYATAAN

Dengan ini saya menyatakan bahwa:

1. Skripsi ini adalah asli dan belum pernah diajukan untuk mendapatkan gelar akademik (sarjana, dan/atau magister), baik di Universitas Putera Batam maupun di perguruan tinggi lain.
2. Skripsi ini adalah murni gagasan, rumusan, dan penelitian saya sendiri, tanpa bantuan pihak lain, kecuali arahan pembimbing.
3. Dalam skripsi ini tidak terdapat karya atau pendapat yang telah ditulis atau dipublikasikan orang lain, kecuali secara tertulis dengan jelas dicantumkan sebagai acuan dalam naskah dengan disebutkan nama pengarang dan dicantumkan dalam daftar pustaka.
4. Pernyataan ini saya buat dengan sesungguhnya dan apabila di kemudian hari terdapat penyimpangan dan ketidakbenaran dalam pernyataan ini, maka saya bersedia menerima sanksi akademik berupa pencabutan gelar yang telah diperoleh, serta sanksi lainnya sesuai dengan norma yang berlaku di perguruan tinggi.

Batam, 11 Februari 2017

Yang membuat pernyataan,

Widy
130210015

ANALISIS DAN IMPLEMENTASI *SECURE SHELL* PADA *UBUNTU SERVER* DENGAN METODE *PORT KNOCKING*

Oleh :
Widy
130210015

SKRIPSI

Untuk memenuhi salah satu syarat
guna memperoleh gelar Sarjana

Telah disetujui oleh Pembimbing pada tanggal
seperti tertera di bawah ini

Batam, 11 Februari 2017

Andi Maslan, S.T., M.SI.
Pembimbing

ABSTRAK

Jaringan komputer memberikan kemudahan antar pengguna komputer, dengan adanya jaringan komputer pertukaran data antar komputer dapat dilakukan dengan mudah dan cepat. Juga bisa melakukan kontrol jarak jauh, salah satunya yaitu *remote* akses dengan menggunakan *telnet*. Namun sekarang ini *telnet* tidak aman lagi dipakai karena proses *remote* akses, dan transmisi data tidak ada enkripsi sehingga rentan terhadap serangan peretas. *Secure Shell* merupakan salah satu solusi untuk masalah keamanan yang ada melalui jaringan. Kedua, tidak hanya mengamankan transfer file, juga membantu dalam *remote login*, *port forwarding* dan mekanisme kontrol akses lainnya. *Port knocking* dapat dijadikan altenatif untuk koneksi pada *server*, yang ingin mempertahankan kondisi semua *port* tertutup sepanjang tidak dibutuhkan. *Secure shell* dengan metode *port knocking* sangat baik diterapkan karena *port* pada layanan ssh di *scanning* oleh *nmap* dengan status *filtered*. *Attacking service* ssh dengan memakai aplikasi *hydra* tidak bisa menemukan *login* dan *password* tesebut. Proses *remote* akses ke *server* menggunakan *secure shell* dengan metode *port knocking* dapat menambah keamanan.

Kata kunci : *Secure Shell*, *Ubuntu Server*, Metode *Port Knokcing*, *Telnet*, *Remote* Akses

ABSTRACT

The computer network provides convenience between computer users, with the computer network data exchange between computers can be done easily and quickly. Also can perform remote control, one of which is remote access using telnet. But now, insecure telnet longer used because the process of remote access, data transmission and no encryption so vulnerable to hacking attacks. Secure Shell is one solution to the security problems that exist over the network. Second, not only securing file transfers, it also helps in remote login, port forwarding and other access control mechanisms. Port knocking can be used as an alternative to the connection on the server, which wanted to keep all the ports closed all unnecessary. Secure shell with excellent port knocking method is applied for the port ssh service in scanning by nmap with the filtered state. Attacking service ssh using hydra application could not find a login and password proficiency level. The process of remote access to the server using the secure shell method port knocking can add security.

Keywords : Secure Shell, Ubuntu Server, Port Knocking Method, Telnet, Remote Access

KATA PENGANTAR

Dengan mengucapkan puji dan syukur kepada Tuhan yang Maha Esa yang telah melimpahkan segala rahmat dan karunianya, sehingga penulis dapat menyelesaikan laporan tugas akhir yang merupakan salah satu persyaratan untuk menyelesaikan program studi strata satu (S1) pada Program Studi Teknik Informatika Universitas Putera Batam.

Penulis menyadari bahwa skripsi ini masih jauh dari sempurna. Karena itu, kritik dan saran akan senantiasa penulis terima dengan senang hati.

Dengan segala keterbatasan, penulis menyadari pula bahwa skripsi ini takkan terwujud tanpa bantuan, bimbingan, dan dorongan dari berbagai pihak. Untuk ini, dengan segela kerendahan hati, penulis menyampaikan ucapan terima kasih kepada:

1. Rektor Universitas Putera Batam.
2. Ketua Program Studi Teknik Informatika Universitas Putera Batam.
3. Andi Maslan, S.T., M.SI. selaku pembimbing Skripsi pada Program Studi Teknik Informatika Universitas Putera Batam.
4. Dosen dan Staff Universitas Putera Batam.
5. Ayah dan Ibu serta saudaraku yang telah mendukung penulis dengan baik.

Batam, 11 Februari 2017

Penulis

DAFTAR ISI

Halaman

DAFTAR TABEL

DAFTAR GAMBAR

Halaman

BAB I
PENDAHULUAN

1.1 Latar Belakang Penelitian

Saat ini perkembangan teknologi yang semakin pesat membawa pengaruh yang cukup besar bagi pengguna yang memafaatkan internet untuk melakukan berbagai hal misalnya transmisi data, transaksi *online*, dan lain-lain. Dalam mengikuti perkembangan teknologi banyak bidang yang membutuhkan jaringan komputer. Jaringan komputer memberikan kemudahan antar pengguna komputer, dengan adanya jaringan komputer pertukaran data antar komputer dapat dilakukan dengan mudah dan cepat. Juga bisa melakukan kontrol jarak jauh, salah satunya yaitu *remote* akses dengan menggunakan *Telnet*.

Server sekarang yang dipakai *ubuntu server*, dan mengontrol jarak jauh antara dua komputer dengan sistem *telnet*. Namun sekarang ini *telnet* tidak aman lagi dipakai karena proses *remote* akses, dan transmisi data tidak ada enkripsi sehingga rentan terhadap serangan peretas maka perlu diterapkan *secure shell*. *Secure shell* merupakan sistem komunikasi yang aman diantara dua sistem yang menggunakan arsitektur *client* dan *server,* serta seorang *user* untuk *login* ke *server* secara *remote*. *Secure shell* mengenkripsi data selama proses komunikasi sehingga menyulitkan peretas yang mencoba membobol *login* dan *password* pengguna.

(Garimella, Kumar: 2015) *Secure Shell* merupakan salah satu solusi untuk masalah keamanan yang ada melalui jaringan. Kedua, tidak hanya mengamankan

transfer *file*, juga membantu dalam *remote login*, *port forwarding* dan mekanisme kontrol akses lainnya. Ini tidak hanya menghilangkan resiko keamanan tetapi juga memberi kita ruang untuk mengembangkan jaringan besar dengan fitur-fitur canggih.

Pada saat ini serangan pada suatu *server* semakin hari semakin meningkat. Terbukanya *port* akan memudahkan peretas untuk menerobos masuk ke dalam *server* melalui *port* tersebut. Para peretas akan mencoba untuk mengeksploitasi berbagai aplikasi yang sedang dijalankan melalui port yang terbuka pada sisi *server*, terutama pada *port* untuk aplikasi *remote server* tentunya akan menjadi titik utama perhatian peretas untuk di eksploitasi. Metode *port knocking* merupakan salah satu metode autentikasi yang dapat digunakan untuk mengatasi masalah tersebut.

(Ricard, Susanto: 2007) *Port knocking* dapat dijadikan altenatif untuk koneksi pada *server*, yang ingin mempertahankan kondisi semua *port* tertutup sepanjang tidak dibutuhkan. Proses *port knocking* lebih tepat digunakan oleh *standalone server*. Yang membutuhkan proses *login* setiap kali hendak mengakses sistem, pembukaan *port* secara spesifik pada pihak tertentu membantu mengontrol akses *port* serta layanannya dari pihak yang tidak sah.

Berdasarkan latar belakang yang dijabarkan diatas tentang rentannya *telnet* terhadap serangan peretas, maka perlu menerapkan *secure shell* dengan metode *port knocking*. Oleh karena itu peneliti akan melakukan **"ANALISIS DAN IMPLEMENTASI *SECURE SHELL* PADA *UBUNTU SERVER* DENGAN METODE *PORT KNOCKING*".**

1.2 Identifikasi Masalah

Berdasarkan pembahasan latar belakang permasalahan dapat di identifikasi beberapa masalah sebagai berikut:

1. *Server* masih menggunakan *TELNET*.
2. *Telnet* rentan terhadap serangan peretas.
3. *Server* belum menerapkan SSH.
4. Masih belum menggunakan metode *PORT KNOCKING*.

1.3 Pembatasan Masalah

Agar pembahasan tidak terlalu meluas pada penelitian ini, maka penelitian ini membatasi permasalahan sebagai berikut:

1. *Secure shell* dan metode *port knocking* diimplementasikan pada jaringan LAN.
2. *Server* menggunakan OS *Ubuntu* 16.04 1 LTS.
3. Tidak membahas sistem operasi secara detail.
4. Hanya menguji keamanan *remote* akses ke *server* dengan aplikasi *Hydra* dan *Nmap*.
5. Menguji keamanan *remote* akses ke *server* dengan melakukan peretasan menggunakan cara *dictionary attack*.

1.4 Perumusan Masalah

Berdasarkan indentifikasi masalah yang telah diuraikan, maka penelitian ini merumuskan masalah sebagai berikut:

1. Bagaimana mengimplementasi *secure shell* pada *ubuntu server* dengan metode *port knocking*?
2. Bagaimana keamanan *server* setelah menerapkan *secure shell* dengan metode *port knocking*?

1.5 Tujuan Penelitian

Tujuan yang ingin dicapai dalam penenlitian ini sebagai berikut:

1. Untuk mengimplementasi *secure shell* pada *ubuntu server* dengan *metode port knocking*.
2. Keamanan *remote* akses ke *server* tidak mudah dibobol oleh para pihak yang tidak berhak.

1.6 Manfaat Penelitian

Dengan dilakukannya penelitian ini diharapkan dapat memberikan manfaat bagi berbagai pihak antara lain:

1.6.1 Aspek Teoritis

1. Agar menambah pengetahuan dalam penelitian tentang *secure shell* pada *unbuntu server* dengan metode *port knocking*.

2. Sebagai tambahan ilmu pengetahuan dan untuk mengembangkan kemampuan dalam melakukan penelitian.

1.6.2 Aspek Praktis

1. Bagi penulis untuk dapat memahami keamanan remote akses ke *server* dengan menggunakan *secure shell* dan metode *port knocking*.
2. Bagi mahasiswa dan masyarakat hasil penelitian ini dapat digunakan sebagai media informasi atau panduan penelitian selanjutnya dan menambah wawasan tentang *secure shell* dengan metode *port knocking*.

BAB II
KAJIAN PUSTAKA

2.1 Teori Dasar

2.1.1 Jaringan computer

Menurut Maslan dan Wangdra (2012: 2) Jaringan komputer adalah sebuah kumpulan dari komputer, printer, dan peralatan lainnya yang terhubung dalam satu kesatuan dan membentuk suatu sistem tertentu. Informasi bergerak melalui kabel atau tanpa kabel sehingga memungkinkan pengguna jaringan komputer dapat saling bertukar informasi (data), mencetak data pada printer yang sama dan dapat secara simultan menggunakan program aplikasi yang sama.

Sedangkan menurut Husda (2012: 77) Jaringan komputer secara sederhana dapat dikatakan sebagai komunikasi antara dua atau lebih komputer yang saling terhubung. Dengan adanya jaringan antar komputer sangat membantu para pengguna komputer dalam bekerja dan berkomunikasi, seperti saling bertukar data, program serta sumber daya komputer seperti media penyimpanan, printer dan lain-lain.

2.1.2 Standar Jaringan Komputer

Maslan (2012: 53) Standarisasi masalah jaringan tidak hanya dilakukan ISO saja, tetapi juga diselenggarakan oleh badah dunia lainnya seperti ITU

(*International Telecommunication Union*), ANSI (*AmSasaan National Standard InstituteI*), NCITS (*National Committee for Information Technology Standardization*), bahkan juga oleh lembaga organisasi profesi IEEE (*Institute of Electrical and Electronics Engineers*) dan ATM-*Forum* di Amerika. Pada prakteknya bahkan vendor-vendor produk LAN bahkan memakai standar yang dihasilkan IEEE. Kita bisa lihat misalnya badan pekerja yang dibentuk oleh IEEE yang banyak membuat standarisasi peralatan telekomunikasi seperti yang tertera pada tabel berikut:

Tabel 2.1 Badan Pekerja di IEEE

Working Group	Bentuk Kegiatan
IEEE802.1	Standarisasi *interface* lapisan atas HILI (*High Level Interface*) dan Data Link termasuk MAC (*Medium Access Control*) dan LLC (*Logical Link Control*).
IEEE802.2	Standarisasi lapisan LLC.
IEEE802.3	Standarisasi lapisan MAC untuk CSMA/CD (10*Base*5, 10*Base*2, 10*Base*T, dll).
IEEE802.4	Standarisasi lapisan MAC untuk Token *Bus*
IEEE802.5	Standarisasi lapisan MAC untuk Token *Ring*
IEEE802.6	Standarisasi lapisan MAC untuk MAN-DQDB (*Metropolitan Area Netwrok-Distributed Queue Dual Bus*).
IEEE802.7	Grup pendukung BTAG (*Broadban Technical Advisory Group*) pada LAN.
IEEE802.8	Grup pendukung FOTAG (*Fiber Optic Technical Advisory Group*).
IEEE802.9	Standarisasi ISDN (*Integrated Services Digital Network*) dan IS (*Integrated Services*) LAN.
IEEE802.10	Standarisasi masalah pengamanan jaringan (LAN *Security*).
IEEE802.11	Standarisasi masalah *wireless* LAN dan CSMA/CD bersama IEEE802.3.
IEEE802.12	Standarisasi masalah 100VG-*Any* LAN.
IEEE802.14	Standarisasi masalah *protokol* CATV.

2.1.3 Jenis Jaringan Komputer

Menurut Husda (2012: 93) Secara umum jaringan komputer terdiri atas lima jenis :

1. *Local Area Network* (LAN)

LAN merupakan jaringan milik pribadi di dalam sebuah gedung atau kampus yang berukuran sampai beberapa kilometer. LAN seringkali digunakan untuk menghubungkan komputer-komputer pribadi dan *workstation* dalam kantor suatu perusahaan atau pabrik-pabrik untuk memakai bersama sumberdaya (*resouce*, misalnya printer) dan saling bertukar informasi.

2. *Metropolitan Area Network* (MAN)

MAN pada dasarnya merupakan versi LAN yang berukuran lebih besar dan biasanya menggunakan teknologi yang sama dengan LAN. MAN dapat mencakup kantor-kantor perusahaan yang letaknya berdekatan atau juga sebuah kota dan dapat dimanfaatkan untuk keperluan pribadi (swasta) atau umum. MAN mampu menunjang data dan suara, bahkan dapat berhubungan dengan jaringan televisi kabel.

3. *Wide Area Network* (WAN)

WAN merupakan jangkauannya mencakup daerah geografis yang luas, sering kali mencakup sebuah negara bahkan benua. WAN terdiri dari kumpulan mesin-mesin yang bertujuan untuk menjalankan program-program (aplikasi) pemakai.

4. *Internet*

Sebenarnya terdapat banyak jaringan di dunia ini, sering kali menggunakan perangkat keras dan perangkat lunak yang berbeda-beda. Orang yang terhubung ke jaringan sering berharap untuk bisa berkomunikasi dengan orang lain yang terhubung ke jaringan lainnya. Keinginan seperti ini memerlukan hubungan antar jaringan yang sering kali tidak compatibel dan berbeda. Biasanya untuk melakukan hal ini diperlukan sebuah mesin yang disebut *gateway* guna melakukan hubungan dan melaksanakan terjemahan yang diperlukan, baik perangkat keras maupun perangkat lunaknya. Kumpulan jaringan yang terkoneksi inilah yang disebut dengan *internet*.

5. *Wireless* (jaringan tanpa kabel)

Wireless (jaringan tanpa kabel), jaringan tanpa kabel merupakan suatu solusi terhadap komunikasi yang tidak bisa dilakukan dengan jaringan yang menggunakan kabel. Misalnya orang yang ingin mendapat informasi atau melakukan komunikasi walaupun sedang berada diatas mobil atau pesawat terbang, maka mutlak jaringan tanpa kabel diperlukan karena koneksi kabel tidaklah mungkin dibuat di dalam mobil atau pesawat. Saat ini jaringan tanpa kabel sudah marak digunakan dengan memanfaatkan jasa satelit dan mampu memberikan kecepatan akses yang lebih cepat dibandingkan dengan jaringan yang menggunakan kabel.

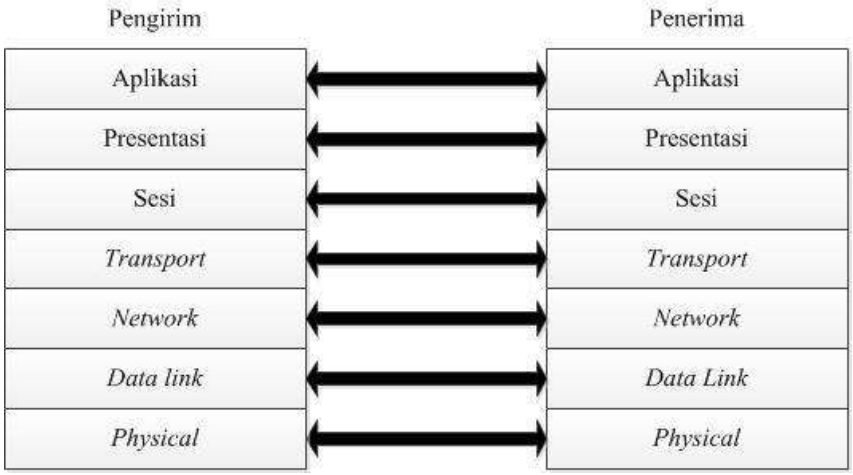
Pada jaringan LAN, MAN, WAN merupakan koneksi antar 2 komputer atau lebih dalam sebuah jaringan. Dalam jaringan tersebut sering menggunakan suatu sistem client *remote* ke *server* dengan jarak jauh. Untuk menghindari serangan

dari peretas pada saat *remote* ke *server* maka sangat bagus diimplementasikan *secure shell* dengan metode *port knocking*.

2.1.4 Model OSI *Layer*

Sukmaaji dan Rianto (2008: 14) OSI memberikan pandangan yang "abstark" dari arsitektur jaringan yang dibagi dalam 7 lapisan. Model ini diciptakan berdasarkan sebuah proposal yang dibuat oleh *International Standard Organization* (OSI) sebagai langkah awal menuju standarisasi protokol internasional yang digunakan pada berbagai layer. Model ini disebut OSI *Refence Model*. *Open System* diartikan sebagai suatu sistem yang terbuka untuk berkomunikasi dengan sistem-sistem lain yang berbeda arsitektur maupun sistem operasi. Prinsip-prinsip yang digunakan bagi ketujuh layer tersebut adalah:

1. Sebuah layer harus dibuat bila diperlukan tingkat abstraksi yang berbeda.
2. Setiap layer harus memiliki fungsi tertentu.
3. Fungsi layer di bawah adalah mendukung fungsi layer di atasnya.
4. Fungsi setiap layer harus dipilih dengan teliti sesuai dengan ketentuan standar protokol internasional.
5. Batas-batas setiap layer diusahakan untuk meminimalkan aliran informasi yang melewati antarmuka.
6. Jumlah layer harus cukup banyak, sehingga fungsi-fungsi yang berbeda tidak perlu disatukan dalam satu layer di luar keperluannya. Akan tetapi jumlah layer juga harus diusahakan sesedikit mungkin sehingga arsitektur jaringan tidak menjadi sulit dipakai.

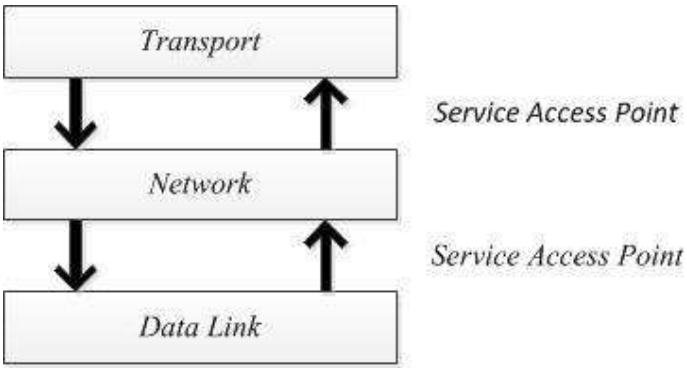


Gambar 2.1 Komunikasi *Peer-to-peer*

Sumber : Sukmaaji dan Rianto

Pada gambar 2.1 tampak bahwa setiap lapisan mempunyai protokol yang saling berkomunikasi (*logic*) dengan protokol pada lapisan yang sama. Data mengalir dari lapisan aplikasi ke bawah hingga lapisan fisik (disebut komunikasi vertikal), kemudian data tersebut dikirim penerima ke atas dari lapisan fisik ke lapisan aplikasi. Masing-masing lapisan berhubungan dengan mekanisme yang disebut sebagai *Service Access Point* (SAP).

Sebagai contoh, antar lapisan *Transport*, *Network*, dan *Data Link*.



Gambar 2.2 *Service Access Point*

Sumber : Sukmaaji dan Rianto

1. *Physical Layer*

Sukmaaji dan Rianto (2008: 16) *Physical layer* atau lapisan fisik melakukan fungsi pengiriman dan penerimaan *bit stream* dalam medium fisik. Dalam lapisan ini kita akan mengetahui spesifikasi mekanikal dan elektrikal dari media transmisi serta antarmukanya. Hal-hal penting yang dapat dibahas lebih jauh dalam lapisan fisik ini adalah:

a. Karakteristik fisik dari media dan antarmuka.
b. Representasi *bit-bit*.

   Dalam hal ini lapisan fisik harus mampu menerjemahkan *bit* 0 atau 1, termasuk pengkodean dan bagaimana mengganti sinyal 0 ke 1 atau sebaliknya.
c. Data *rate* (laju data).
d. *Line configuration* (konfigurasi saluran). Misalnya: *point-to-point* atau *point-to-multipoint configuration*.

Lapisan fisik pada LAN di antarnya:

a. *Internert*/IEEE 802.3, *Baseband* LAN beroperasi 10 Mbps melalui kabel koaksial.
b. 100-Mbps *Ethernet* (*Fast Ethernet*), *High-speed* LAN.
c. 1000-Mbps *Ethernet* (*Gigabit Ethernet*), *Hight-speed* LAN.
d. *Fiber Distributed Digital Interface* (FDDI), 100-Mbps *token-passing*, *dual-ring* LAN menggunakan kabel *fiber optic*.

2. *Data Link*

Sukmaaji dan Rianto (2008: 17) *Data Link Layer* komunikasi data dilakukan dengan menggunakan identitas berupa alamat simpul fisik yang disebut sebagai alamat *hardware* atau *hardware address*. Proses komunikasi antara komputer atau simpul jaringan hanya mungkin terjadi, bila kedua belah pihak mengetahui identitas masing-masing melalui alamat fisik (*physical address*). Bentul topologi yang digunakan ditentukan oleh protokol *Data Link*. Sebagi contoh adalah *BUS* untuk teknologi *Ethernet*, *RING* untuk teknologi *Token Ring* ataupun teknologi FDDI. Selain ketiga bentul topologi tersebut pada komunikasi serial terdapat topologi *point-to-point* atau *point-to-multipoint* pada jaringan yang menggunakan teknologi *Frame Relay dan* ATM.

Tugas utama lapisan *data link* dalam proses komunikasi data adalah:

a. *Framing*: membagi *bit stream* yang diterima dari lapisan *network* menjadi unit-unit data yang disebut *frame*.
b. *Physical addreing*: definisi identitas pengirim dan atau penerima yang ditambahkan data *header*.
c. *Flow control*: melakukan tindakan untuk membuat stabil laju *bit rate* atau

laju *bit stream* berlebih atau berkurang.

d. *Communication control*: menentukan *device* yang harus dikendalikan pada saat tertentu jika ada dua koneksi yang sama.

3. *Network Layer*

Sukmaaji dan Rianto (2008: 19) Pada lapisan ini terjadi proses pendefinisian alamat logis (*logical addressing*), kemudian mengombinasikan *multiple data link* menjadi satu *internetwork*. Lapisan *Network* bertanggung jawab untuk membawa paket dari satu simpul ke simpul lainnya dengan mengacu pada *logical address*. Fungsi lain adalah sebagai *packet forwarder* (penerus). Lapisan *Network* sebagai *packet forwarder* mengantarkan paket dari sumber (*source*) ke tujuan (*destination*) yang disebut dengan istilah *routing.* Ada dua tugas pokok lapisan *network* yaitu:

a. *Logical addressing*: pengalaman secara logis yang ditambahkan pada *header* lapisan *network*. Pada jaringan TCP/IP pengalamatan logis ini populer dengan sebutan IP *Address*.

b. *Routing*: hubungan antar jaringan yang membentuk *internetwork* membutuhkan metode jalur alamat agar paket dapat ditransfer dari satu *device* yang berasal dari jaringan satu menuju *device* lain pada jaringan yang lain. Fungsi *routing* didukung oleh *routing* protokol yaitu protokol yang bertujuan mencari jalan terbaik menuju tujuan dan tukar-menukar informasi tentang topologi jaringan dengan *router* yang lainnya.

4. *Transport Layer*

Sukmaaji dan Rianto (2008: 20) Lapisan *transport* (*end-to-end*) yang dapat dijelaskan sebagai berikut:

a. *Service-point addresing*. Suatu komputer sering menjalankan berbagai macam *program* aplikasi ataupun *services* berlainan pada waktu bersamaan. Karena itu, lapisan *transport* ini tidak hanya menangani pengiriman *source-to-destination* dari komputer satu ke komputer yang lain, namun lebih spesifik kepada *delivery* jenis *message* untuk aplikasi yang berlainan.
b. *Segmentation dan reassembly*. Sebuag *message* dibagi dalam segmen-segmen yang terkirim. Setiap segmen memiliki *sequence number*. *Sequence number* berguna bagi lapisan *transport* untuk merakit (*reassembly*) segmen-segmen yang terpecah menjadi *message* yang utuh.
c. *Flow control*. Seperti halnya lapisan *data link*, lapisan *transport* bertanggung jawab untuk melakukan kontrol aliran (*flow control*). Bedanya dengan *flow control* di lapisan *data link* adalah dilakukan untuk *end-to-end.*
d. *Error control*. Fungsi tugas ini sama dengan tugas *error control* di lapisan *data link*, namun berorientasi *end-to-end*.

5. *Session Layer*

Sukmaaji dan Rianto (2008: 21) Lapisan sesi membuka, merawat, mengendalikan, dan melakukan terminasi hubungan antar simpul. Lapisan aplikasi dan presentasi melakukan *request* dan menunggu *reponse* yang dikoordinasikan oleh lapisan di atasnya misalnya:

a. RPC (*Remote Procedure Call*): protokol yang mengeksekusi *program* pada komputer *remote* dan memberikan nilai balik kepada komputer lokal sebagi hasil eksekusi tersebut.
b. NFS (*Network File System*)
c. SQL (*structured Query Language*)

6. *Presentation Layer*

Sukmaaji dan Rianto (2008: 21) Berfungsi untuk mentranslasikan data yang akan ditransmisikan oleh aplikasi ke dalam format yang dapat ditransmisikan melalui jaringan. Protokol yang berada dalam *level* ini adalah perangkat lunak redirektor (*redirector software),* seperti layanan *Workstation* (dalam *Windows* NT) dan juga *Netwrok shell* (*Virtual Network computing* (VNC)) atau *Remote Desktop Protokol* (RDP)). Lapisan presentasi melakukan *coding* dan konversi data misalnya format data untuk *image* dan *sound* (JPG, MPEG, TIFF, WAV, dan lain-lain).

7. *Application Layer*

Sukmaaji dan Rianto (2008: 21) Aplikasi adalah layanan/*service* yang mengimplementasikan komunikasi antar simpul. *Application Layer* berfungsi sebagai antarmuka dengan aplikasi dengan funsionalitas jaringan mengatur bagaimana aplikasi dapat mengakses jaringan dan membuat pesan-pesan kesalahan. Beberapa hal yang dilakukan oleh lapisan aplikasi mengidenfikasi mitra komunikasi, aplikasi transfer data, *Resource Availability*, dan lapisan aplikasi terkait dengan aplikasi *end-user*.

Secure shell dapat diterapkan pada persentasi *layer*, *session layer*, *application layer*. Pada *application layer* merupakan layanan untuk aplikasi transfer *file* atau akses suatu komputer. Presentasi *layer* ini membuat dua host dapat berkomunikasi sedangkan *session layer* membuat sesi untuk proses dan mengakhiri sesi, lapisan ini dapat menghubungkan lagi jika sesi login terganggu sehingga terputus.

2.2 Teori Khusus

2.2.1 Secure Shell

Menurut Prasetiyo (2015: 30) *Secure Shell* atau SSH adalah protokol jaringan yang memungkinkan pertukaran data melalui saluran aman antara dua perangkat jaringan. SSH terutama banyak digunakan pada sistem berbasis *Linux* dan *Unix* untuk mengakses akun *shell*. SSH dirancang sebagai pengganti *Telnet* dan *Shell Remote* tidak aman lainnya, yang mengirim informasi, terutama kata sandi, dalam bentuk teks sederhana yang membuatnya mudah dicegat. Enkripsi yang digunakan oleh SSH meyediakan kerahasiaan dan integritas data melalui jaringan yang tidak aman seperti internet.

Sedangkan menurut Garimella dan Kumar (2015: 187) SSH *Secure Shell* adalah antarmuka perintah berdasarkan *UNIX* dan protokol jaringan kriptografi digunakan untuk melindungi data dalam transmisi antara perangkat menyediakan otentikasi yang kuat dan membentuk saluran aman melalui jaringan tidak aman dalam arsitektur client-server. *Secure Shell* adalah program untuk *login* ke

komputer lain melalui jaringan, untuk menjalankan perintah di mesin *remote*, dan untuk memindahkan file dari satu mesin ke mesin lainnya. Hal ini terdiri dari rangkaian tiga utilitas, *Slogin*, Ssh dan SCP. Utilitas dasar lainnya termasuk Ssh-*Add*, Ssh-*Agent*, Ssh-*keysign*, Ssh-*KeyScan*, Ssh-*keygen*, dan Sftp-*Server*. Hal ini merupakan pengganti versi sebelumnya dari utilitas *UNIX rlogin*, rsh, rcp, dan *rdist*.

Menggunakan SSH ini *Secure Shell* utilitas di kedua ujung sambungan dikonfirmasi menggunakan sertifikat digital. Berbeda utilitas seperti *Telnet*, password yang dienkripsi. Enkripsi ini membuat sulit bagi seseorang menembus kerahasiaan data atau password dikompromikan. Hal ini dapat diimplementasikan lebih dari sebagian besar sistem operasi (Win, MAC, dan *Unix* atau *Linux*). SSH *Secure Shell* menggunakan RSA kriptografi kunci publik untuk koneksi dan otentikasi. Algoritma enkripsi ini termasuk DES, 3DES, *Blowfish* dan IDEA untuk autentikasi keduanya ujung sambungan, mengenkripsi semua data yang dikirimkan, kerahasiaan, kompresi data, melindungi integritas data, *multiplexing* dengan ssh-2 dan validasikan nilai-nilai yang dikembalikan oleh layanan seperti DNS atau jaringan protokol (tcp). Hal ini biasanya digunakan untuk mengontrol *web*, *aplikasi server* dan peralatan jaringan *remote*.

2.2.2 Ubuntu Server

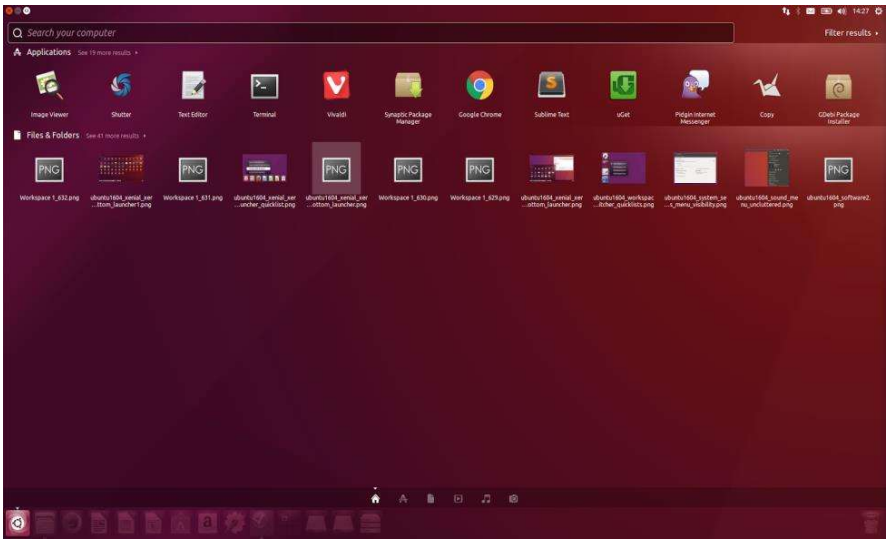
Fajri dkk (2014: 63) Sistem Operasi *Ubuntu* pertama kali ditemukan oleh Mark Shuttleworth. Sistem operasi ini berbasi *Linux* dan merupakan turunan dari Sistem Operasi *Debian*. *Ubuntu* memiliki dua versi yaitu versi *desktop* dan

versi *server*. Sampai saat ini, rilis *ubuntu* telah mencapai versi rilis 12.10 untuk versi *desktop* dan versi *server*. Ada versi baru yang dimiliki *ubuntu* saat ini yaitu versi *cloud* yang digunakan untuk *Cloud Computing*. *Ubuntu* merupakan salah satu sistem operasi yang paling banyak digunakan karena lebih *user friendly*. Oleh sebab itu, *ubuntu* banyak dijadikan sebagai *base* dalam pengembangan sistem operasi baru, seperti *Backtrack*, *Linux Mint*, dan *BlankOn*. Fitur-fitur dalam Os Ubuntu 16.04 1 LTS sebagi berikut:

a. LTS *Kernel*: *Xenial Xerus* akan didasarkan pada *Linux* 4.4 LTS.
b. *Unity spyware*: dalam rilis preseden *Ubuntu Unity Dash* memungkinkan pengguna untuk mencari *online* dari berbagai situs seperti *Amazon* atau Wikipedia.
c. *Unity Launcher* posisi: berkat kerja dari Marco Trevisan, pengguna *Unity* akan dapat memutar *Unity Launcher* dan posisi ke bawah layar.
d. *Snappy*: pengembang *Ubuntu* berusaha untuk membawa tajam ke *Ubuntu* 16.04 1 LTS dengan *Unity* 7.
e. *Ubuntu software center*.

Tim *Ubuntu* dengan bangga mengumumkan rilis *Ubuntu* 16.04.1 LTS (*Long-Term Support*) untuk *Desktop*, *Server*, dan produk *Cloud* nya, serta rasa lain dari *Ubuntu* dengan dukungan jangka panjang. Seperti biasa, rilis saat ini mencakup banyak pembaruan, dan media instalasi diperbarui telah disediakan sehingga update lebih sedikit akan perlu didownload setelah instalasi. Ini termasuk *update* keamanan dan koreksi untuk *bug* berdampak tinggi lainnya, dengan fokus pada menjaga stabilitas dan kompatibilitas dengan *Ubuntu* 16.04 LTS.

Berikut gambar OS *ubuntu* 16.04 1 LTS:



Gambar 2.3 OS *ubuntu* 16.04 1 LTS

.

2.2.3 Port Knocking

Berdasarkan penelitian Fajri dkk (2014 :63) *Port Knocking* adalah sebuah metode sederhana untuk memberikan akses remote tanpa meninggalkan port dalam keadaan selalu terbuka. Hal ini akan memberikan perlindungan kepada *server* dari *port scannig* dan serangan *scripts kiddies*.

Menurut Richard dan Susanto (2007: D-3) *Port knocking* merupakan metode sistem autentikasi yang secara khusus dibuat untuk jaringan. Ide dasar dari sistem autentikasi ini telah lama digunakan namun baru pada tahun 2003, dalam salah satu kolom di majalah *Linux Journal*, seorang pakar jaringan komputer Martin Krzywinski kembali mempopulerkan metode ini dengan beberapa terobosan–terobosan menghadapi serangan yang mungkin terjadi. Pada dasarnya *port knocking* dapat didefinisikan sebagai suatu metode

komunikasi antara dua komputer, dimana informasi yang dikirimkan di *encode* dalam bentuk usaha koneksi ke *port–port* dalam urutan tertentu. Usaha membangun koneksi ini bisa disebut juga ketukan-ketukan. Mekanisme *port knocking* akan menggunakan *file log* yang dibuat oleh *firewall* untuk mengetahui apakah suatu usaha koneksi telah dibuat oleh suatu *host* atau tidak.

Menurut Krzywinski (2009), pola kerja metode *port knocking* ini memiliki beberapa tahap, sebagai berikut:

a. Tahap pertama klien melakukan koneksi ke komputer *server* ke salah satu *port* di komputer *server*, misal *port* 22, namun koneksi tersebut di blok oleh *firewall* komputer *server*.
b. Tahap kedua klien melakukan koneksi ke *port–port sequences* yang telah didefinisikan dalam *file* konfigurasi *daemin port knocking* ke komputer *server* dengan mengirimkan paket SYN didalamnya. Selama fase ini, klien tidak akan mendapatkan respon apa–apa.
c. Tahap ketiga *daemon port knocking* mencatat adanya percobaan koneksi dan kemudian melakukan autentikasi terhadap percobaan tersebut. Apabila autentikasi sesuai dengan yang didefinisikan pada *daemon port knocking* dalam hal ini adalah *port sequences* yang didefinisikan, maka *daemon port knocking* akan melakukan *overwrite* terhadap *rule* yang telah didefinisikan didalam *firewall* agar membuka *port* yang ingin dituju oleh klien.

d. Tahap keempat setelah melakukan autentikasi klien telah bisa melakukan koneksi ke *port* yang dituju menggunakan aplikasi seperti pada umumnya.

Setelah selesai, klien memutuskan koneksi dengan *port* dan kemudian mengirimkan paket SYN kembali agar *daemon port knocking* menulis ulang *rule* pada *firewall* agar tidak bisa dilakukan koneksi kembali ke *port* 22. Fajri dkk (2014 :63).

2.3 Tools

1. Komputer *Intel*(R) *Pentium*(R) CPU G2030 3.00Ghz
2. OS *ubuntu* 16.04 1 LTS.
3. *Putty* merupakan aplikasi *open source* untuk *remote* akses pada komputer melalui jaringan dapat digunakan pada ssh, *telnet*, *rlogin*, raw, dan *serial*.
4. *Opehssh* sebuah aplikasi *open source* untuk *remote* akses.
5. *Knockd* merupakan aplikasi untuk melakukan konfigurasi *port knocking*. *knockd* adalah *daemon* yang mendengarkan permintaan masuk ke kotak anda, dan bereaksi ketika kombinasi tertentu tercapai. Setelah *knockd* diinstal dan berjalan, anda mengubah aturan *firewall* anda (misalnya *iptables*) untuk menolak semua lalu lintas masuk ke *port* 22. Tidak ada *break* dalam upaya yang mungkin, dan *log* keamanan anda tetap bagus. Bila anda ingin menghubungkan ke kotak melalui SSH, anda pertama kali mengirimkan serangkaian pukulan ke kotak. Jika kombinasi yang tepat diterima, *knockd* akan membuka lubang di *firewall* untuk IP anda pada *port* 22.

6. *Hydra* adalah aplikasi untuk melakukan penyerangan pada *username* dan *password*.
7. *Nmap* sebuah aplikasi untuk melakukan *scanning* pada *port* yang terbuka.

2.4 Penelitian Terdahulu

Berdasarkan teori yang telah diuraikan, maka penelitian ini mengacu pada penelitian terdahulu untuk memperkuat hasil penelitian yang dilakukan. Penelitian terdahulu dapat dijabarkan sebagi berikut :

1. **Sembiring.dkk. 2009. Analisa dan Implementasi Sistem Keamanan Jaringan Komputer dengan *Iptables* sebagai *Firewall* Menggunakan Metode *Port Knocking*.**

   Penelitian Sembiring dkk terdapat masalah dalam *port* terbuka tetap menjadi kerentanan, mereka mengizinkan koneksi untuk aplikasi tetapi juga dapat berubah menjadi pintu terbuka untuk serangan. Metode *port kncoking* sangat berguna diterapkan pada sistem keamana jaringan komputer. Metode *port knocking* sangat berguna bagi administrator jaringan atau *server* yang harus mengurusi jaringan atau *server* secara terus menerus dari mana saja, karena *port knocking* aman digunakan untuk membuat komunikasi antar komputer pada jaringan komputer. Dengan metode *port knocking*, komunikasi antar komputer dapat dilakukan meskipun *port* yang tertutup.

2. **Burande.dkk. 2014. Wireless Network Security By SSH Tunneling.**

   Dari hasil penelitian Burande dkk memberikan kesimpulan *SSH tunneling* ini sederhana dan solusi efektif untuk pengiriman konten yang aman melalui

internet. Solusi ini secara efektif akan mengamankan browsing internet dari paket *sniffing*. Dibandingkan dengan link lain, jaringan, dan aplikasi keamanan langkah-langkah seperti WEP, IPSEC dan PGP, bersama dengan menginstal dan mengkonfigurasinya, *Secure Shell* relatif aman, handal, cepat dan mudah. Dengan menerapkan *Secure Shell*, perusahaan membuat tujuan umum yang luas tunneling platfrom yang dapat digunakan untuk menerapkan berbagai kebijakan keamanan, menjaga privasi, autentisitas, autorisasi dan integritas berbagai aplikasi.

3. **Richard.Susanto. 2007. Sistem Autentikasi Port Knocking Pada Sistem Closed Port.**

Penelitian Ricard dan Susanto bahwa *program port knocking* dapat dijadikan alternatif untuk koneksi pada *server*, yang ingin mempertahankan kondisi semua *port* tertutup sepanjang tidak dibutuhkan, proses *port knocking* lebih tepat digunakan oleh *standalone* s*erver*. Yang membutuhkan proses *login* setiap kali hendak mengakses sistem, pembukaan *port* secara spesifik pada pihak tertentu membantu mengontrol akses *port* serta layanannya dari pihak yang tidak sah.

4. **Fajri.dkk. 2014. Analisa Port Knocking Pada Sistem Operasi Linux Ubuntu Server 12.04 LTS.**

Dalam penelitian Fajri dkk Autentikasi standar layanan *SSH Server*, *FTP Server*, dan *MySQL Server* tidak aman bila tidak diikuti dengan password yang unik. Contoh password standar: *root*, *server,admin@root*, dan lain-lain. Sedangkan contoh password unik: @dm1n@5erv3r, myPa55w0rd$y$4dmIn, dan lain-lain. Penerapan autentikasi *Port Knocking* untuk layanan *SSH Server*, *FTP*

Server, dan *MySQL Server* dapat meningkatkan keamanan akses. Waktu untuk mengakses komputer *server* dengan autentikasi *port knocking* dua kali lebih lama dibanding waktu akses komputer *server* tanpa autentikasi *port knocking*. Penggunaan *firewall* UFW lebih baik dibanding *firewall IPTables*, karena ufw menutup akses kesemua layanan kecuali yang diizinkan. Sedangkan *IPTables* tetap memperlihatkan *port* layanan namun dengan status *Filtered*.

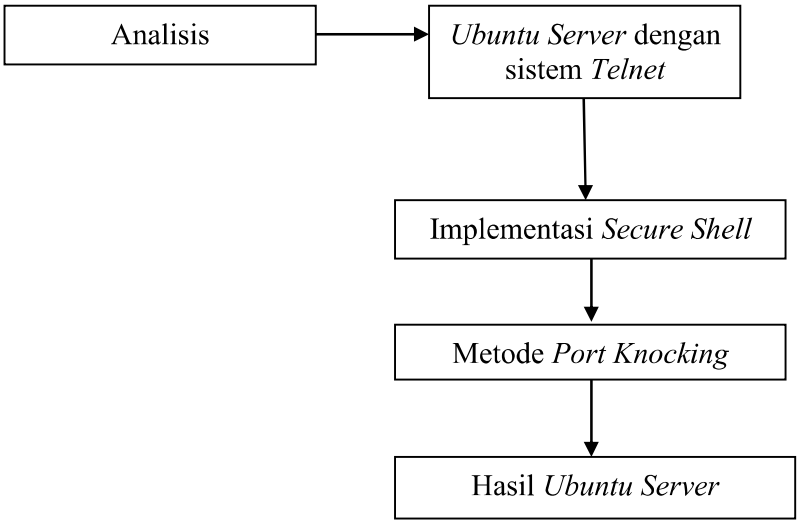
5. **Garimella.Kumar. 2015. Secure Shell Its Signification In Networking (SSH).**

Pembahasan Garimella dan Kumar Secure Shell adalah salah satu solusi untuk masalah keamanan yang ada melalui jaringan. Kedua, tidak hanya mengamankan transfer file, juga membantu dalam remote login, port forwarding dan mekanisme kontrol akses lainnya. Ini tidak hanya menghilangkan resiko keamanan tetapi juga memberi kita ruang untuk mengembangkan jaringan besar dengan fitur-fitur canggih. Meskipun Ssh memiliki kelemahan dan ancaman seperti *men-in-the-middle attacks*, kinerja, masalah port dan pembatasan lainnya, penelitian selanjutnya yang ditujukan untuk meningkatkan ini dan menambahkan fitur baru. Jumlah keprihatinan untuk data sensitif melalui jaringan yang menuju perluasan lebih lanjut dari keamanan jaringan.

2.5 Kerangka Pemikiran

Uma Sekaran dalam bukunya *Business Researh* (1992) mengemukakan bahwa, kerangka berfikir merupakan model konseptual tentang bagaimana teori berhubungan dengan berbagai faktor yang telah di identifikasi sebagai masalah

yang penting Kerangka berfikir yang baik akan menjelaskan secara teoritis pertautan antar variabel yang akan diteliti. Kerangka berfikir dalam suatu penelitian perlu dikemukakan apabila dalam penelitian tersebut berkenaan dua variabel atau lebih. Sugiyono (2012: 60) Berikut ini adalah kerangka pemikiran peneliti:

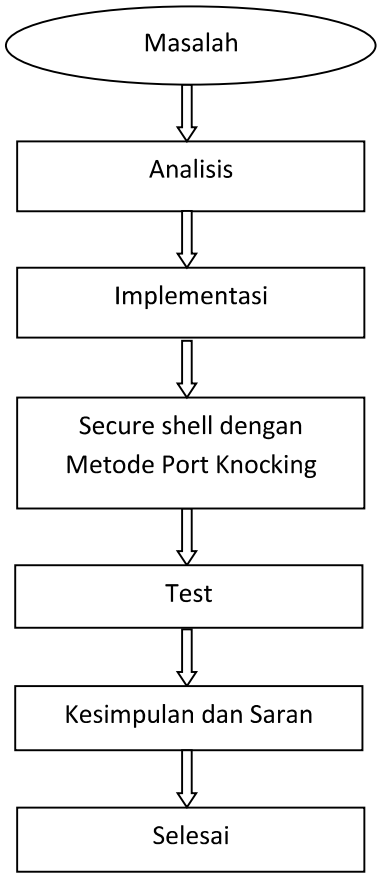


Gambar 2.4 Kerangka Pemikiran

BAB III
METODE PENELITIAN

4.1 Desain Penelitian

Menurut (Sudaryono 2015:3) desain penelitian berisi rumusan tentang langkah-langkah penelitian, dengan pendekatan, metode penelitian, teknik pengumpulan data, dan sumber data tertentu serta alasan-alasan mengapa menggunakan metode tersebut. Sedangkan menurut (Kuncoro 2009) desain penelitian menggambarkan apa yang akan dilakukan oleh peneliti dalam terminologi teknis. (Sudaryono, 2015: 157).



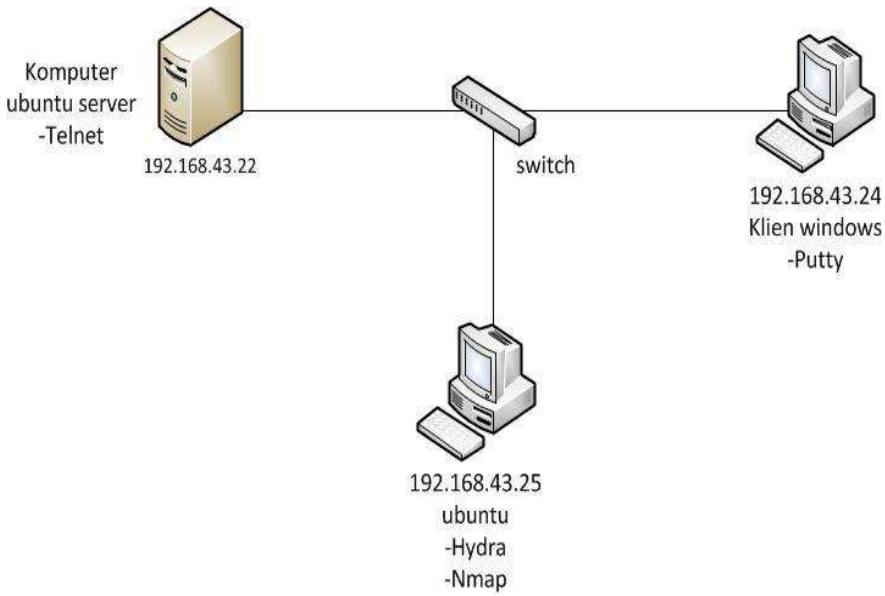
Gambar 3.1 Desain penelitian

4.2 Analisis *Network Security* Model Lama

1. Topologi Jaringan

Topologi jaringan yang dipakai menggunakan satu komputer sebagai klien, satu komputer sebagai *server*, dan satu komputer untuk mengetes proses keamanan jaringan saat melakukan *remote* akses ke *server*. Koneksi *server* pada klien dengan menggunakan *swtich*.

Pada *server* memiliki *ip address* 192.168.43.22 dan klien *windows* memiliki *ip address* 192.168.43.24, dan klien *ubuntu* 192.168.43.25. Berikut contoh gambar topologi jaringan lama:



Gambar 3.2 Topologi jaringan

2. *Software* yang sedang dipakai

Server memakai sistem operasi *ubuntu* 16.04.1 LTS. Pada *server ubuntu* diterapkan *telnet server* dengan aplikasi *telnetd* untuk *remote* akses ke *server*. Klien memakai sistem operasi *windows* dengan aplikasi *putty* untuk melakukan proses *remote* ke *server*, dan juga memakai aplikasi *Hydra dan Nmap* untuk menguji sistem keamanan pada proses terjadinya *remote* akses ke *server*.

4.3 Implementasi *Network Security* Model Baru

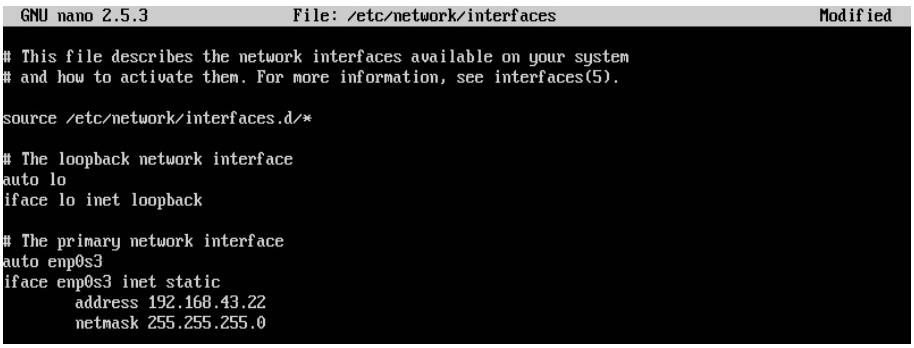
4.3.1 *Softwar*e Yang Dipakai

Pada sistem operasi masih menggunakan *ubuntu* 16.04.1 LTS. *Server* tersebut menerapkan SSH *server* dengan aplikasi *openssh*. Dan metode *port knocking* dengan aplikasi *knockd*. Untuk menguji sistem keamana saat *remote* ke *server* dengan aplikasi *hydra*, dan untuk *remote* akses masih menggunakan aplikasi *putty*.

4.3.2 Tahapan Rencana Implementasi

Peneliti menyiapkan tiga komputer, satu komputer sebagai *server*, satu komputer sebagai klien,dan satu komputer untuk menguji keamanan *remote* akses. Komputer *server* di *install* dengan sistem operasi *Ubuntu* 16.04.1 LTS. Pada *server* diimplementasikan *secure shell* dan *port knocking* sebagai berikut:

1. Konfigurasi *interfaces* memberi *ip address* 192.168.43.22 *netmask* 255.255.255.0 dan *save* ctrl x pilih *yes* lalu ketik *service networking restart*.

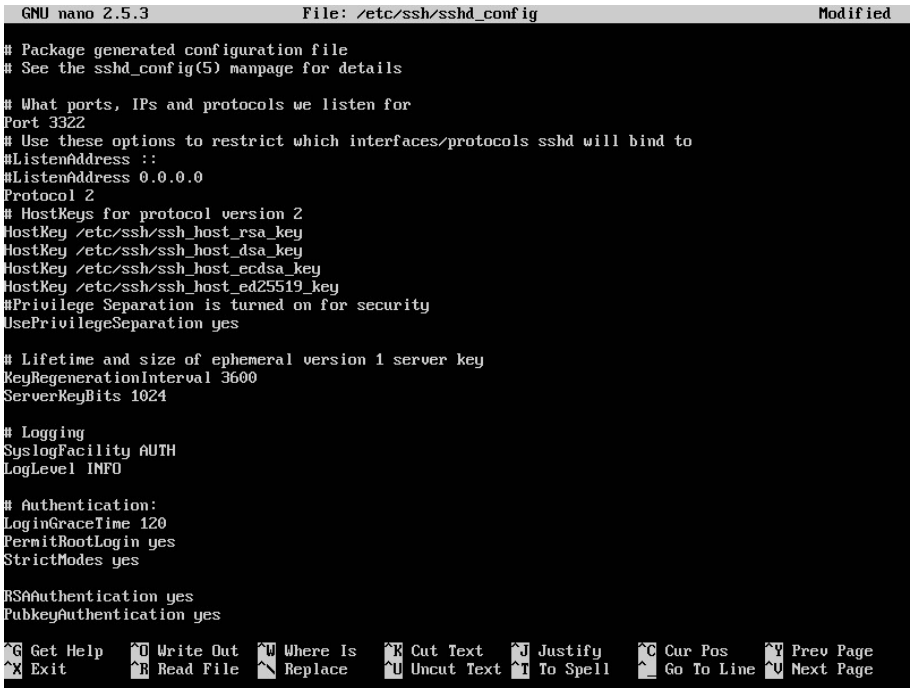


Gambar 3.3 *Nano* /etc/*network/interfaces*

2. *Install openssh.*

Gambar 3.4 Apt-*get install openssh-server*

3. Konfigurasi ssh ganti *port* menjadi 3322 dan *permitrootlogin* ganti *yes* lalu ctrl x pilih *yes*.



Gambar 3.5 *Nano* /etc/ssh/*sshd_config*

4. *Restart* ssh

```
root@ubuntu:~# service ssh restart
root@ubuntu:~#
```

Gambar 3.6 *Service ssh restart*

5. *Install knockd*

```
root@ubuntu:~# apt-get install knockd
Reading package lists... Done
Building dependency tree
Reading state information... Done
knockd is already the newest version (0.5-3ubuntu1).
0 upgraded, 0 newly installed, 0 to remove and 110 not upgraded.
root@ubuntu:~#
```

Gambar 3.7 Apt-*get install knockd*

6. *Install iptables*

```
root@ubuntu:~# apt-get install iptables
Reading package lists... Done
Building dependency tree
Reading state information... Done
iptables is already the newest version (1.6.0-2ubuntu3).
0 upgraded, 0 newly installed, 0 to remove and 110 not upgraded.
```

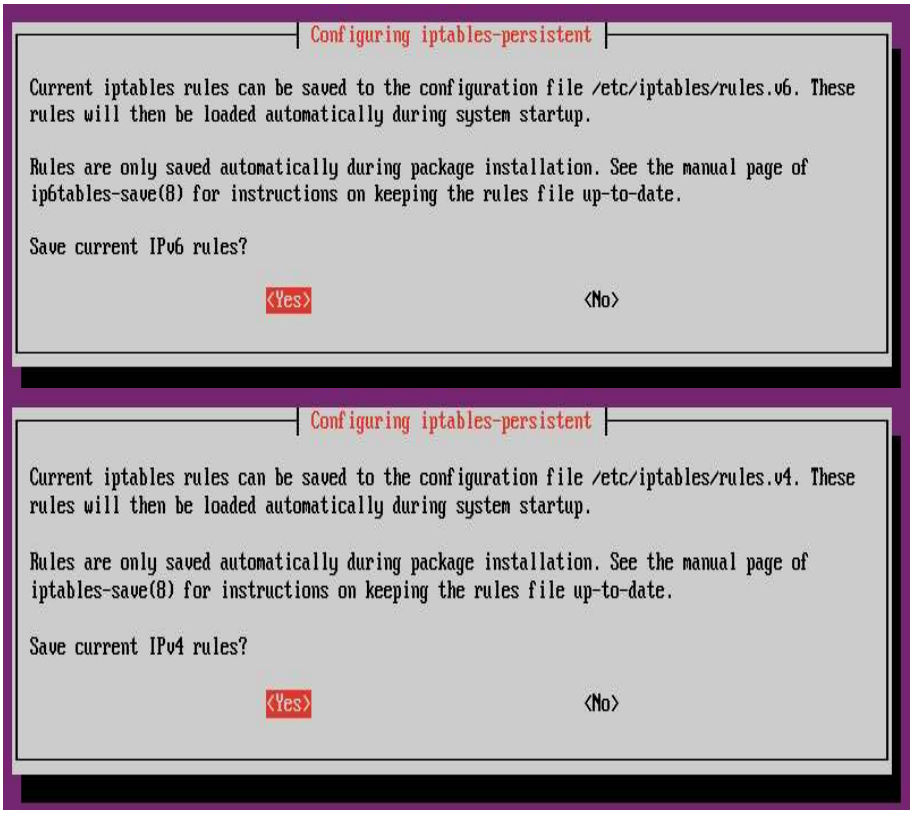
Gambar 3.8 Apt-*get install iptables*

7. Menambah aturan *iptables*

```
root@ubuntu:~# iptables --flush
root@ubuntu:~# iptables -t nat --flush
root@ubuntu:~# iptables -t mangle --flush
root@ubuntu:~# iptables --policy OUTPUT ACCEPT
root@ubuntu:~# iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
root@ubuntu:~# iptables -A INPUT -p tcp --destination-port 3322 -j DROP
```

Gambar 3.9 Aturan *iptables*

8. *Install iptables persistent* pilih *yes*



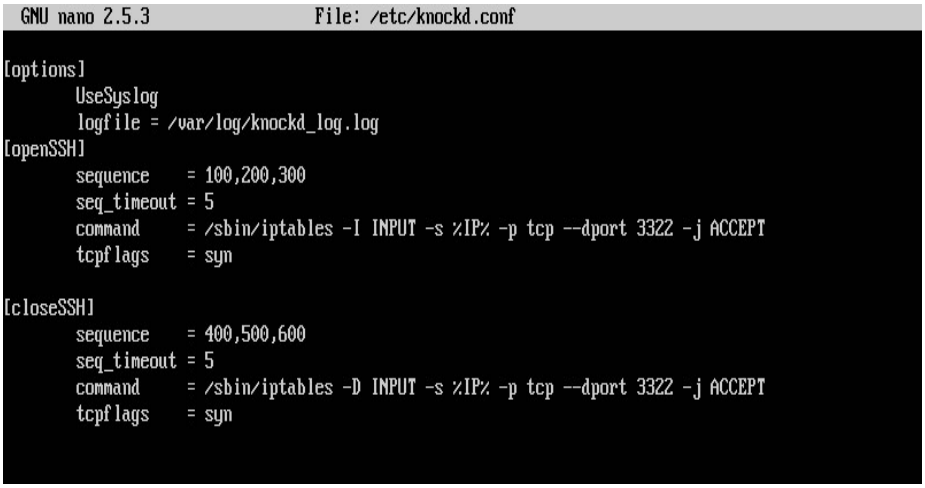
Gambar 3.10 Apt-*get install iptables-persistent*

9. *Iptables-save* menyimpan *rule iptables* yang telah diinput.

```
root@ubuntu:~# iptables-save
# Generated by iptables-save v1.6.0 on Thu Jan  5 19:16:43 2017
*nat
:PREROUTING ACCEPT [4:864]
:INPUT ACCEPT [4:864]
:OUTPUT ACCEPT [11:813]
:POSTROUTING ACCEPT [11:813]
COMMIT
# Completed on Thu Jan  5 19:16:43 2017
# Generated by iptables-save v1.6.0 on Thu Jan  5 19:16:43 2017
*filter
:INPUT ACCEPT [1:229]
:FORWARD ACCEPT [0:0]
:OUTPUT ACCEPT [18:1403]
-A INPUT -m conntrack --ctstate RELATED,ESTABLISHED -j ACCEPT
-A INPUT -p tcp -m tcp --dport 3322 -j DROP
COMMIT
# Completed on Thu Jan  5 19:16:43 2017
```

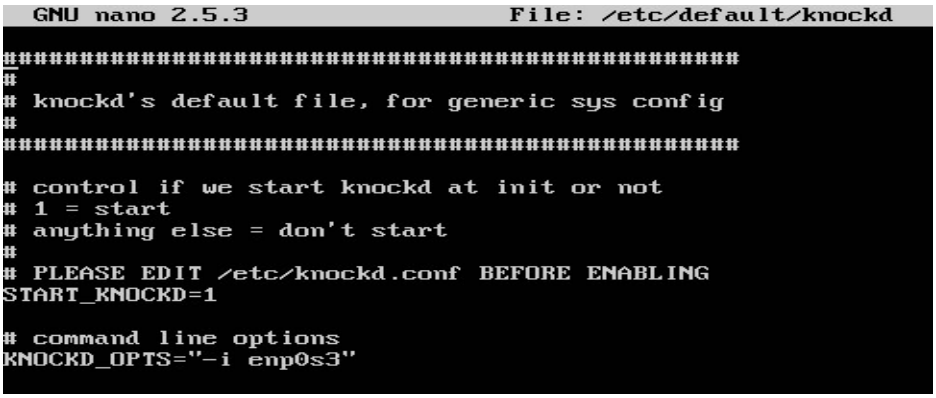
Gambar 3.11 *Iptables-save*

10. Atur ketukkan *port knocking* pada *sequence* 100 200 300 dan *seq_timeout* 5.



Gambar 3.12 *Nano* /etc/*knockd.conf*

11. Konfigurasi *default knockd* pada *start_knockd* diganti 1 dan *knockd_opts* diganti enp0s3.

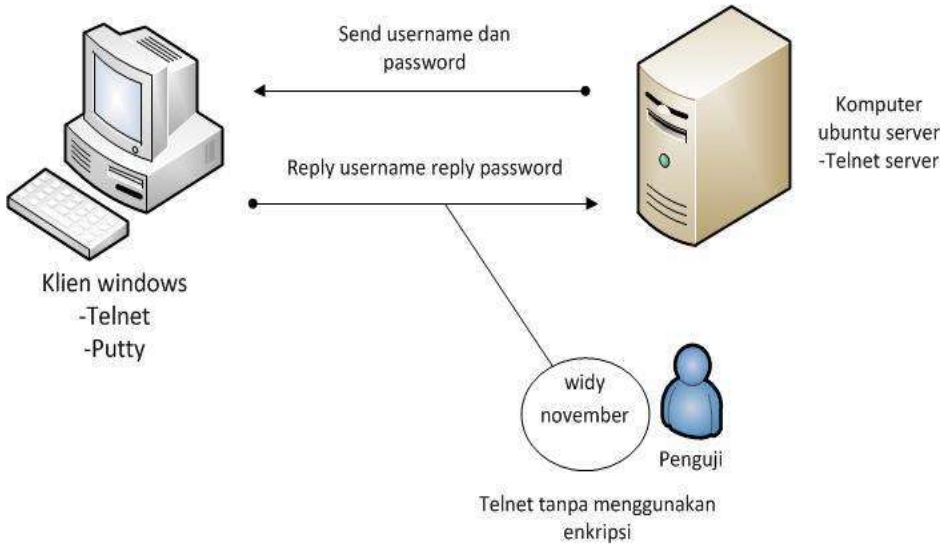


Gambar 3.13 *Nano* /etc/*default*/*knockd*

Sesudah konfigurasi *knockd* maka ketik *service knockd restart*, untuk menjalankan *knockd* ketik *service knockd start*. Pada komputer *ubuntu* untuk menguji keamanan *remote* akses ke *server* di install aplikasi *hydra* dan *nmap* dengan perintah *apt-get install hydra*, *apt-get install nmap*.

4.3.3 Perbedaan *Network Security* Model Lama dengan Model Baru

1. ***Network security* model lama saat terjadi *remote* akses antara klien dan *server* menggunakan *telnet* tidak terenkripsi berikut contoh gambar:**

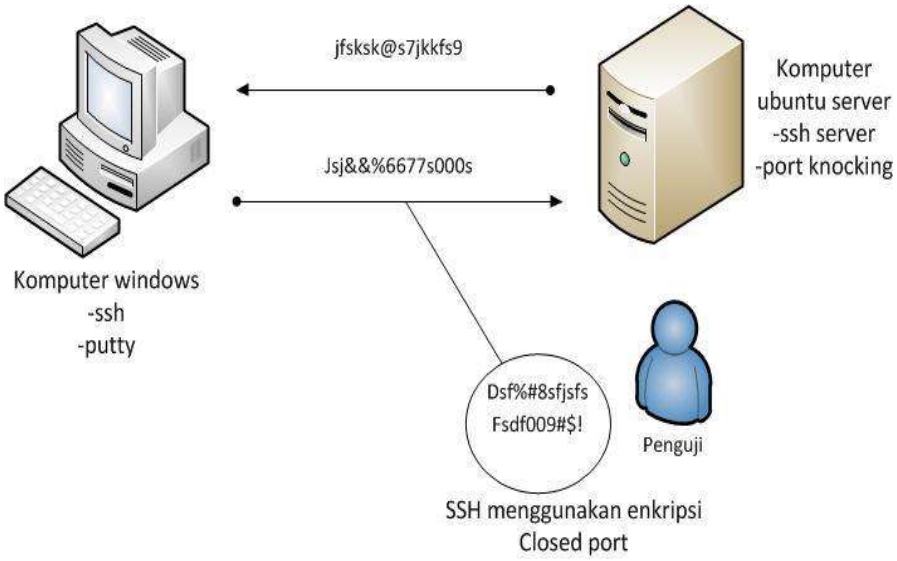


Gambar 3.14 *Telnet*

(Sumber: Burande, *et al*., 2014: 4)

Pada gambar diatas dalam melakukan *telnet*, klien *windows* memakai aplikasi *putty* untuk melakukan *remote* akses ke *server* dengan *port* 23. Dalam aplikasi klien *windows* akan mengirimkan permintaan ke *server telnet*. *Server* tersebut akan membalas meminta nama pengguna dan kata sandi, jika di terima sama *server* maka *telnet* klien akan tersambung ke *telnet server*. Dalam proses *remote* akses ke *server* menggunakan *telnet* tidak menggunakan enkripsi.

2. ***Network security*** **model baru pada proses** ***remote*** **akses antara klien dan** ***server*** **terenkripsi dan juga harus melalui** ***port knocking*** **yang telah di atur berikut contoh gambar:**



Gambar 3.15 Ssh dengan metode *port knocking*

(Sumber: Burande, *et al*., 2014: 4)

Pada gambar diatas dalam melakukan proses *remote* akses ke *server*, komputer *windows* memakai aplikasi *putty* dan melalui *port* 3322. *Port* tersebut di tutup dengan metode *port knocking* dan diatur berapa ketukan sesuai *port* berapa yang diinginkan. Ssh memiliki kunci *public* dan *private* dalam identitasnya. Klien *windows* melakukan otentikasi ke *server* dengan kunci *public* dan *private* yang sama dengan *server*, juga menggunakan *username* dan *password*.

4.4 Lokasi dan Jadwal Penelitian

4.4.1 Lokasi Penelitian

Lokasi penelitian dilakukan pada komplek Batam Executive Centre Blok E No.15.

4.4.2 Jadwal Penelitian

Penelitan akan dilakukan selama 5 bulan dimulai dari September 2016 sampai dengan Januari 2017 dapat dilihat pada tabel berikut ini:

Tabel 3.1 Jadwal Penelitian

No	Kegiatan	Sep 2016				Okt 2016				Nov 2016				Des 2016				Jan 2017			
		1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
1	Pengajuan judul																				
2	Pengajuan surat ijin penelitian																				
3	Analisis Masalah																				
4	Implementasi SSH dan Port knocking																				
5	Pengujian																				
6	Laporan																				