BAB III

METODOLOGI PENELITIAN

3.1 Gambaran Umum Penelitian

Penelitian ini dilakukan untuk menganalisa kinerja *hybrid port knocking* yang diimplementasikan pada aplikasi *client-server*. *Hybrid port knocking* adalah metode yang menggabungkan 3 konsep yaitu *port knocking*, steganografi, serta *mutual authentication*. Dalam meningkatkan keamanannya teknik kriptografi dan steganografi digunakan untuk mengirim TCP *payload*, sedangkan *mutual authentication* diperlukan untuk mengautentikasi *client* dan *server* yang sedang berkomunikasi. Jika dalam proses *mutual authentication* salah satu pihak baik *server* maupun *client* tidak benar maka komunikasi akan terputus. Setelah melakukan implementasi *hybrid port knocking* pada jaringan, maka dilakukan analisa kinerja sistem. Analisa dilakukan untuk mengetahui berapa lama waktu yang diperlukan untuk memroses *hybrid port knocking script* dan berapa banyak penggunaan memori.

3.2 Metode Penelitian

Metode penelitian dalam penelitian ini dapat dijelaskan sebagai berikut:

a. Studi literatur

Melakukan studi literatur untuk mencari dan mengumpulkan beberapa informasi yang berkaitan dengan penelitian. Teori yang dikaji antara lain adalah pengenalan *port knocking*, implementasi *port knocking*, cara kerja *port knocking*, metode-metode pengembangan *port knocking* pada penelitian sebelumnya, dan teori pendukung lainnya. Referensi yang digunakan meliputi buku, jurnal, majalah *online*, forum, artikel, dan lain-lain.

b. Analisis dan perancangan aplikasi

Menganalisa metode implementasi *hybrid port knocking* pada aplikasi *client-server* sebagai acuan. Tahap perancangan meliputi perancangan konfigurasi pada *client* dan *server*.

c. Implementasi aplikasi

Mengimplementasikan *hybrid port knocking*sesuai dengan persyaratan yang ditetapkan.

d. Uji coba dan pembahasan

Melakukan uji coba terhadap aplikasi yang telah diimplementasikan serta melakukan evaluasi pada hasil dengan memberikan beberapa pembahasan.

3.3 Spesifikasi umum sistem

Sistem berupa aplikasi *client-server* yang menggunakan *hybrid port knocking*. Aplikasi menggunakan bahasa pemrograman *python* versi 2.7.3[14] dan diimplementasikan pada sistem operasi Linux Ubuntu 12.04. Prinsip aplikasi ini adalah memanfaatkan *firewall* untuk membatasi akses yang masuk dan keluar pada jaringan komputer. Secara *default, firewall* diatur untuk menutup akses *client* ke semua *port* yang berada di *server* kecuali *port* 80 (http). Saat *meng-execute hybrid port knocking*diperlukan beberapa program pendukung diantaranya adalah *scapy*[16], *python imaging library*[17] dan GnuPG[15]. *Scapy* digunakan untuk mengirimkan dan meng-*capture* paket melalui protokol TCP, *python imaging library* untuk memroses gambar yang digunakan dalam steganografi, GnuPG digunakan untuk melakukan kriptografi.

Aplikasi ini memerlukan konfigurasi di sisi *client* dan *server*. Konfigurasi di sisi *client* meliputi *port knock sequence*, direktori tempat penyimpanan gambar yang digunakan untuk steganografi, direktori tempat penyimpanan *public key* yang digunakan untuk kriptografi, *default user id* yang dipercaya oleh *server*.

Di sisi *server* konfigurasinya meliputi *knock sequence port* yang terdiri dari setidaknya 5 buah *port* yang unik, *port range* yang akan diperiksa (misalnya *port* 1-65535), pemeriksaan *port knock sequence*, jumlah *bytes* yang digunakan untuk menghasilkan *random* karakter pada proses kriptografi, direktori tempat penyimpanan *public key*, jumlah *thread* yang dihasilkan oleh proses, nama yang digunakan untuk pengaturan *firewall iptables*, serta perintah *firewall* iptables yang digunakan untuk membuka *port* atau menutup *port* .

Setelah semua program pendukung seperti scapy, python imaging library (PIL), python cryptoraphy, line profiler, serta memory profiler diinstall dan konfigurasi telah sesuai maka *client* dapat meng-*execute* aplikasi ini dengan mengetikkan perintah melalui terminal dengan format seperti nama *python script* yang akan di-*execute*, nama *domain*, *username*, perintah yang dinginkan, misalnya untuk membuka port 22 digunakan perintah *./hpkClient monicasabrina@yahoo.com 192.168.56.7 O 22*. Jika format *execute*-nya tidak sesuai maka akan muncul pesan kesalahan yang berisi pilihan untuk mengirimkan perintah (O: untuk *open port*, U: untuk *username*, C: untuk *close port*, E: untuk mengirmkan *echo message* dan restart layanan yang disediakan oleh server) serta formatnya yang benar sehingga *client* dapat memperbaikinya lagi.

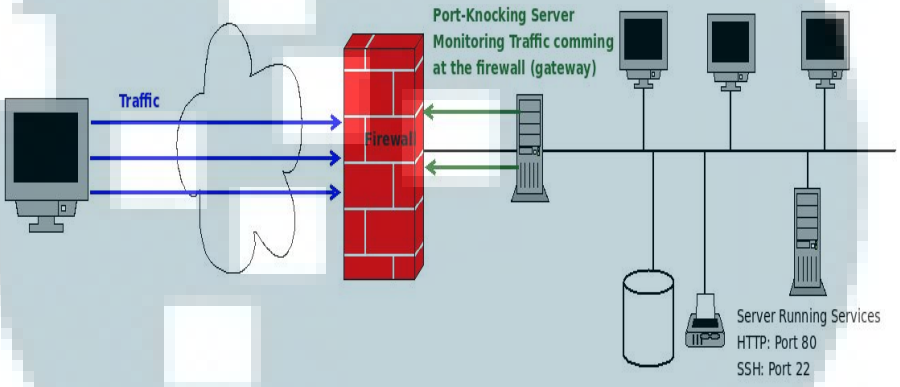
Aplikasi ini dapat menerima 4 perintah yaitu perintah untuk membuka port, mengirimkan echo message, me-restart layanan yang disediakan oleh server serta menutup kembali port yan telah dibuka.

Aplikasi yang sedang berjalan diamati untuk mengetahui kinerjanya. Pengujian meliputi beberapa aspek diantaranya adalah mengetahui berapa lama waktu yang diperlukan untuk memroses setiap baris *hybrid port knocking script* serta penggunaan memori.

3.4 *Flow chart*

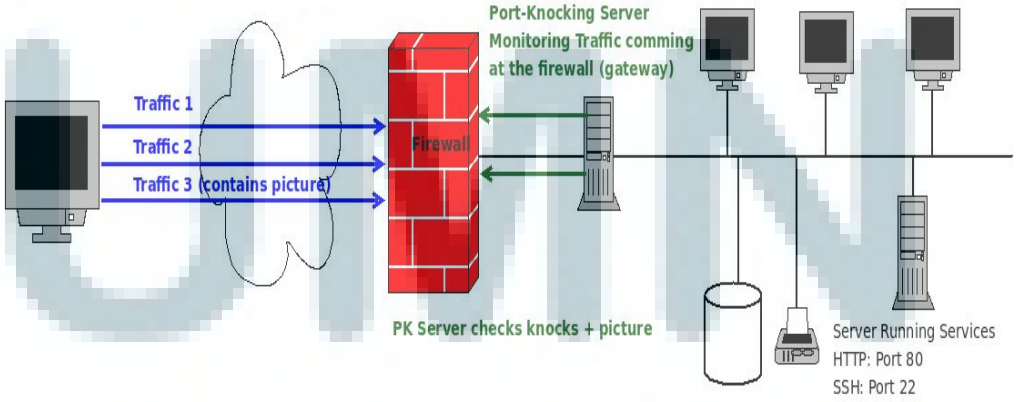
Penelitian ini bertujuan untuk membahas cara mengimplementasikan *hybrid port knocking* pada aplikasi *client-server* serta melakukan analisa kinerja. *Flow chart hybrid port knocking*[5] dapat dilihat pada gambar 3.1.

Berdasarkan gambar 3.1 dapat dilihat bahwa saat aplikasi dijalankan, proses awalnya adalah *port knocking server monitoring* setiap *traffic* yang membawa paket melalui *firewall* seperti terlihat pada gambar di bawah.



Gambar 3.2 *Server* melakukan *monitoring*[3]

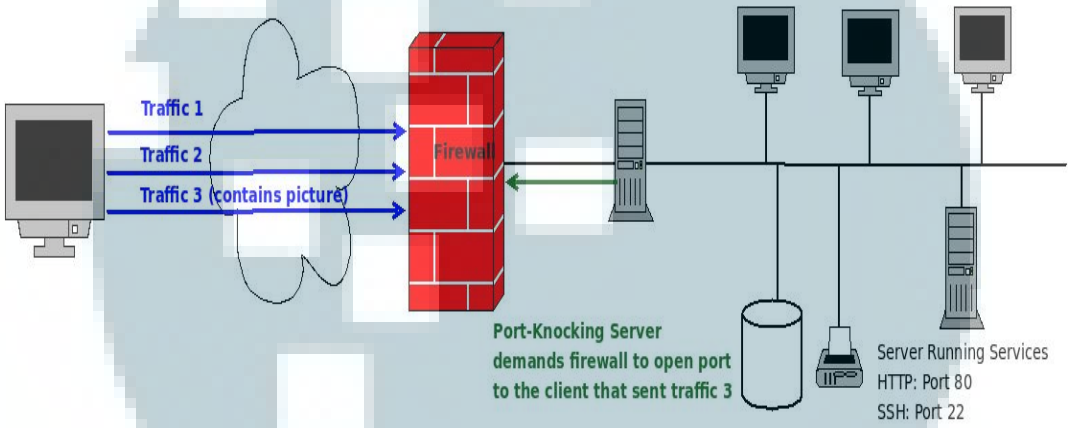
Payload yang berisi gambar akan di*capture*, namun jika tidak maka paket tidak akan diteruskan.



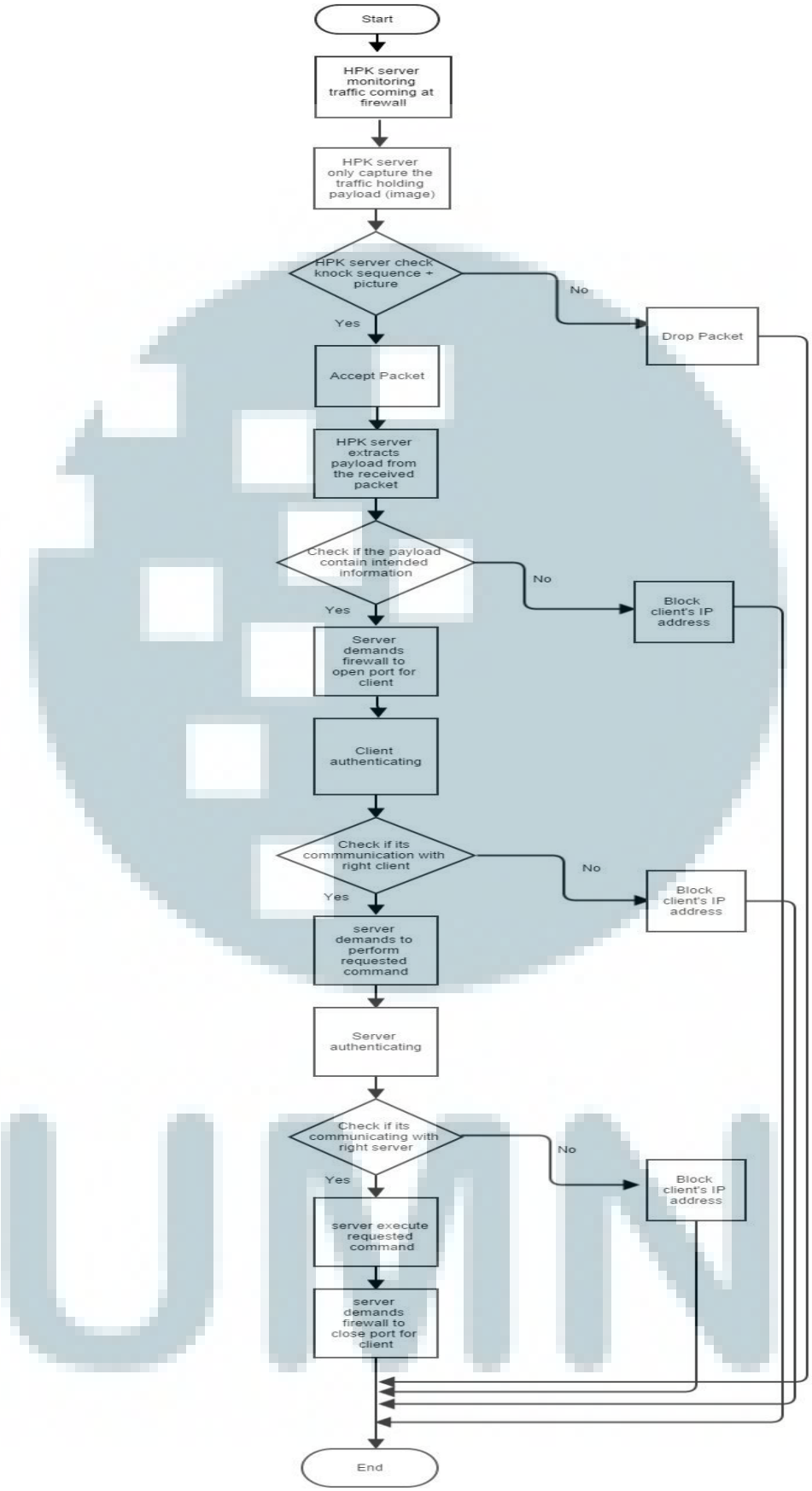
Gambar 3.3 *Server* meng-*capture payload* yang berisi gambar[3]

Server akan meng-*extract payload* (gambar) pada paket yang diterima. *Payload* harus berisi informasi untuk membuktikan identitas *client* dan melakukan *request* yang disembunyikan menggunakan steganografi.

Paket berisi informasi yang diinginkan yaitu *knock sequence* untuk meminta *firewall* membuka atau menutup *port* seperti ditunjukkan oleh gambar 3.4 atau *execute command* untuk melakukan *remote* access pada *server*



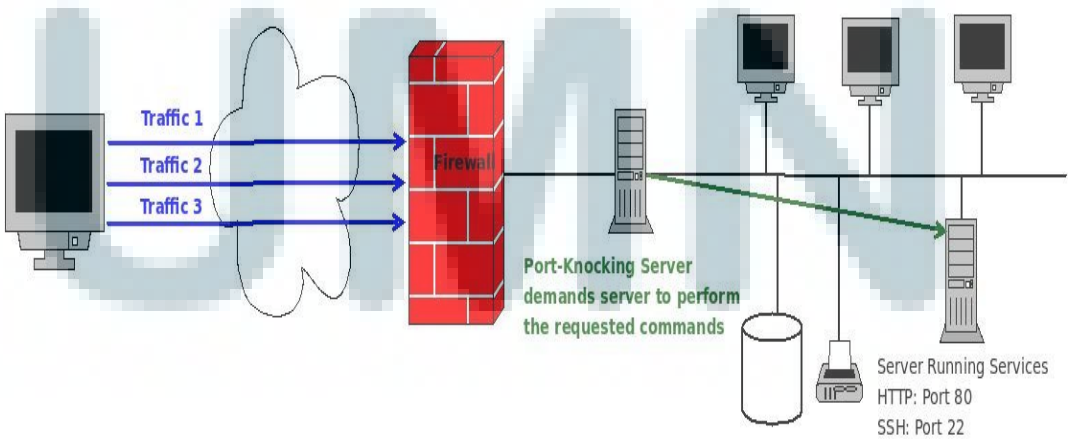
Gambar 3.4 *Server* meminta *firewall* untuk membuka *port* bagi *client*[3]



Gambar 3.1 *hybrid port knocking flow chart*

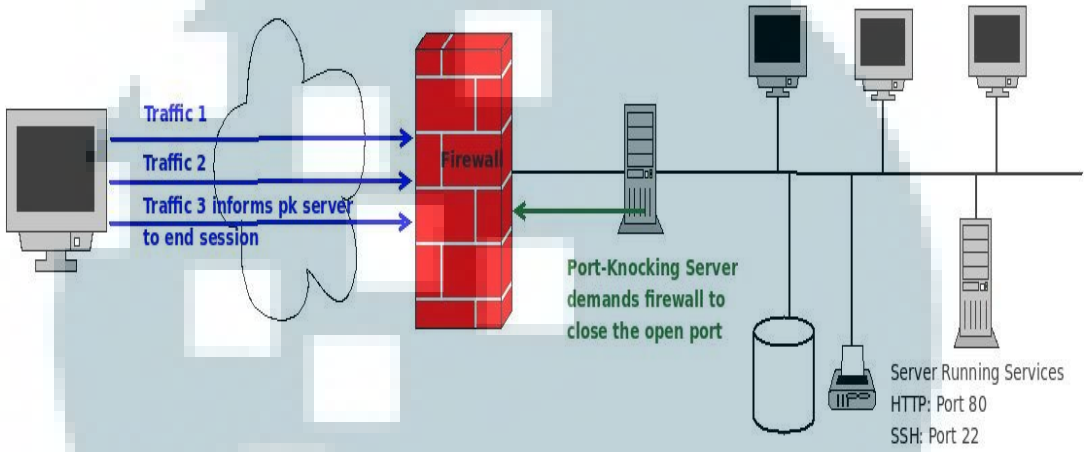
Sebaliknya jika paket tidak berisi informasi yang diinginkan, maka *server* melakukan *blocking* terhadap alamat IP tesebut. Pada tahap ini tidak ada *port* yang dibuka atau ditutup serta tidak ada *execute command* yang diproses, hanya melakukan pemeriksaan untuk proses selanjutnya. Setelah *server* telah memastikan bahwa paket membawa informasi yang diinginkan, maka *server* harus memastikan apakah *client* tersebut adalah *client* yang benar. *Server* melakukan *generate random number* dan mengenkripsinya menggunakan *client* GnuPG *public key* lalu mengirimkan paket berisi *payload* yang dienkripsi ke *client*. Saat paket diterima oleh *client*, paket di-*extract* dan didekripsi menggunakan *client* GnuPG *priavte key*.

Client kemudian memilih *random number* dan mengenkripsinya lalu mengirimkan kembali ke *server* untuk membuktikan bahwa *client* sedang berkomunikasi dengan *server* yang tepat. Status *server* pada tahap ini masih dalam keadaan *monitoring* dan menerima balasan dari *client* terhadap pemeriksaan *random number* yang dikirim oleh *server*. Lalu *server* meng-*extract* balasan tersebut dan memeriksa kembali apakah pesan berisi *random number* yang sama dengan yang di-*generate* oleh *server* sebelumnya. Jika iya, maka *server* akan membuka atau menutup *port* yang sesuai dengan *requestclient*, selain itu juga akan *execute remote command* yang diminta oleh *client*.



Gambar 3.5 *Server* memenuhi *client's request*[3]

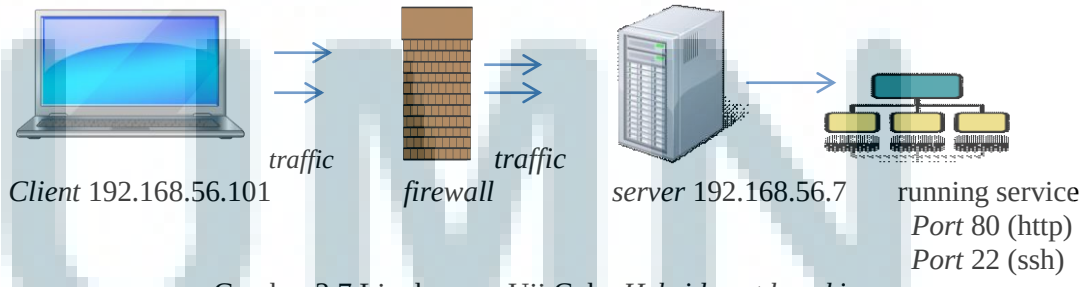
Langkah terakhir adalah proses penutupan *port*. Setelah *requestclient* telah terpenuhi maka *server* akan menutup kembali *port* yang terbuka. Penutupan *port* dapat dilakukan oleh *client* dengan mengirim *command* kepada *server* atau *server* dapat memutuskan untuk menutup *port* setelah beberapa waktu tertentu, seperti ditunjukkan oleh gambar 3.6



Gambar 3.6 Penutupan *port* [3]

3.5 Pembangunan Lingkungan Uji coba

Aplikasi *hybrid port knocking*[3] dijalankan pada *virtual machine* (*client* dan *server*) melalui virtualbox yang ditunjukkan oleh gambar 3.7.



Gambar 3.7 Lingkungan Uji Coba *Hybrid port knocking*

Pembangunan lingkungan uji coba mencakup pengaturan di sisi *server* maupun *client*. Pengaturan dilakukan dengan menginstall beberapa *software* diantaranya adalah *python* 2.7.3, versi *python* yang digunakan adalah *python* 2.6 ke atas atau dapat juga

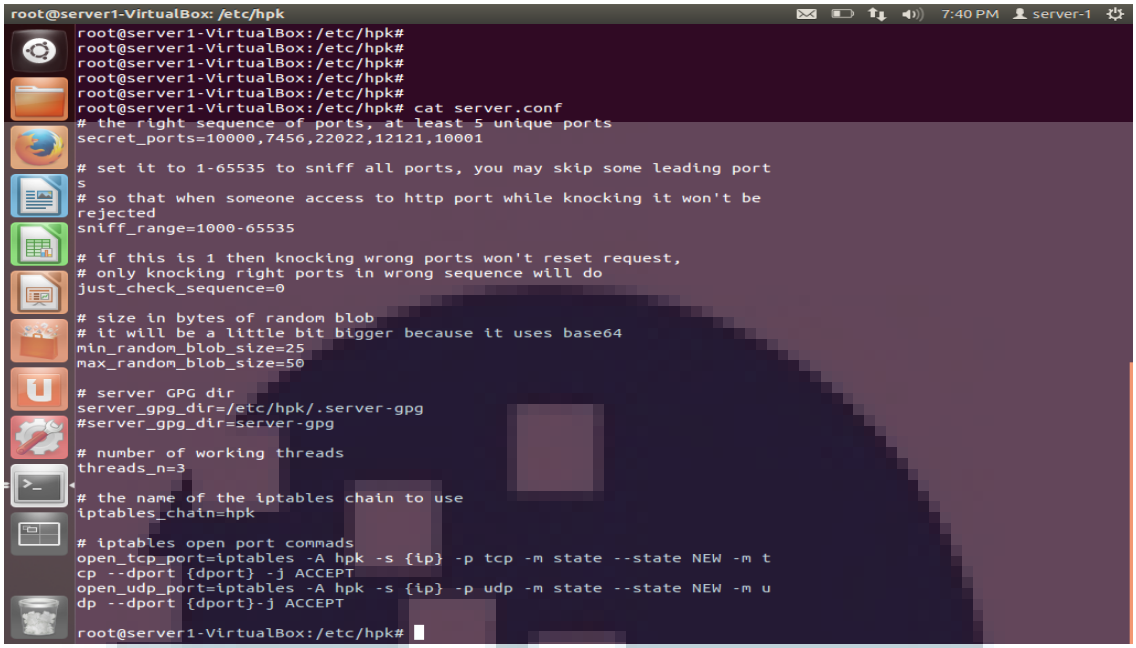
menggunakan *python* versi 2.6, *python imaging library* (PIL) untuk proses steganografi, GnuPG untuk enkripsi dan dekripsi *payload*, *Scapy* diperlukan untuk mengirimkan dan meng-*capture payload*, serta *Linux kernel* terbaru dengan *iptables* (misalnya: 2.6).

Setelah semua *software* yang diperlukan telah di-install, selanjutnya adalah membuat direktori untuk menyimpan *python script* untuk *server* dan *client*. Di dalam direktori tersebut tersimpan semua script yang digunakan untuk implementasi *hybrid port knocking*termasuk *folder* untuk menyimpan kunci yang digunakan dalam proses enkripsi serta konfigurasi sistemnya. *Folder* yang berisi kunci harus diatur *permission*-nya aar tidak dapat diakses oleh pihak yang tidak berkepentingan. Pengaturan *permission* menggunakan *chmod* 600 yang artinya hanya *administrator* yang dapat mengakses direktori ini.

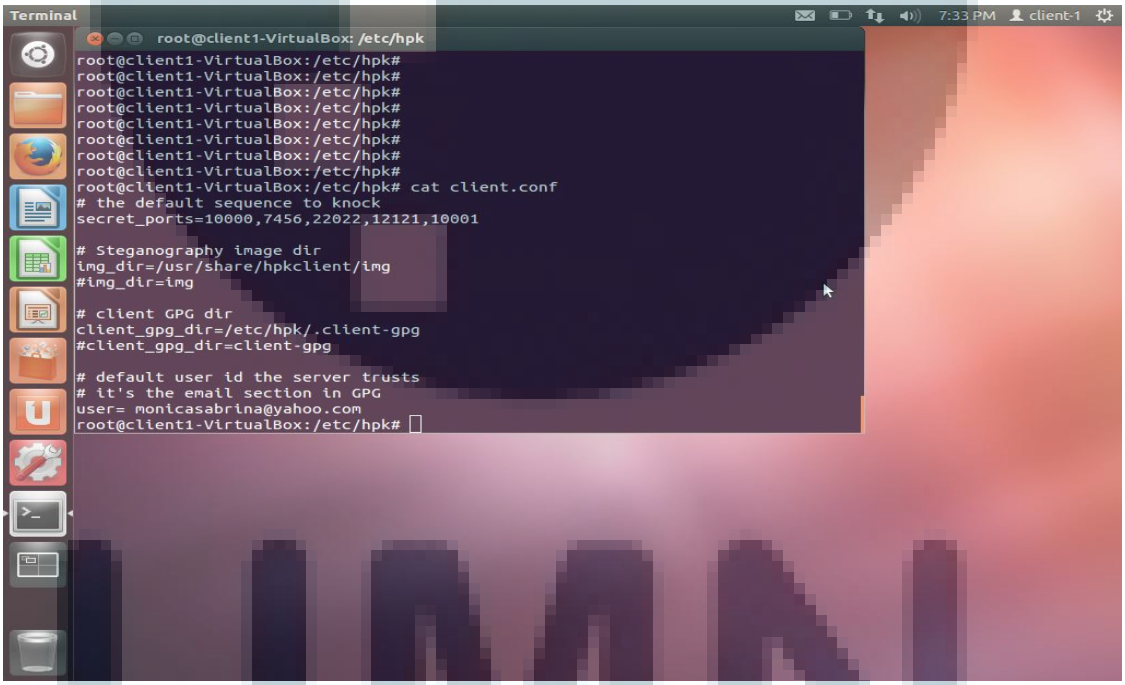
3.6 Struktur Navigasi Aplikasi

Aplikasi ini dijalakan melalui terminal yang disediakan secara *default* oleh sistem operasi Linux. Sebelum meng-*execute* aplikasi *python*, terlebih dahulu seorang *administrator* dan *client* harus *login* sebagai *root*. Selanjutnya baik *server* maupun *client* harus berada di direktori dimana aplikasi *python*-nya disimpan untuk dapat meng-*execute* aplikasi.

Pada penelitian ini *server script* disimpan dengan nama *HPKServer* dan ditempatkan pada direktori /etc/hpk, begitu juga *client script* disimpan dengan nama *HPKClient* dan ditempatkan di direktori /etc/hpk. Di dalam direktori /etc/hpk di *server* maupun *client* berisi sebuah direktori yang diberi nama *HPK*. Direktori ini berisi semua *script* yang berkaitan dengan *port knocking*, diantaranya adalah *gnupg script*, *steganography script, port knockingscript*(*client script* dan *server script*) , *utility script*, serta *initial script*. Konfigurasi *server* dan *client* disimpan di direktori /etc/hpk yang terdapat pada masing-masing *virtual machine*.



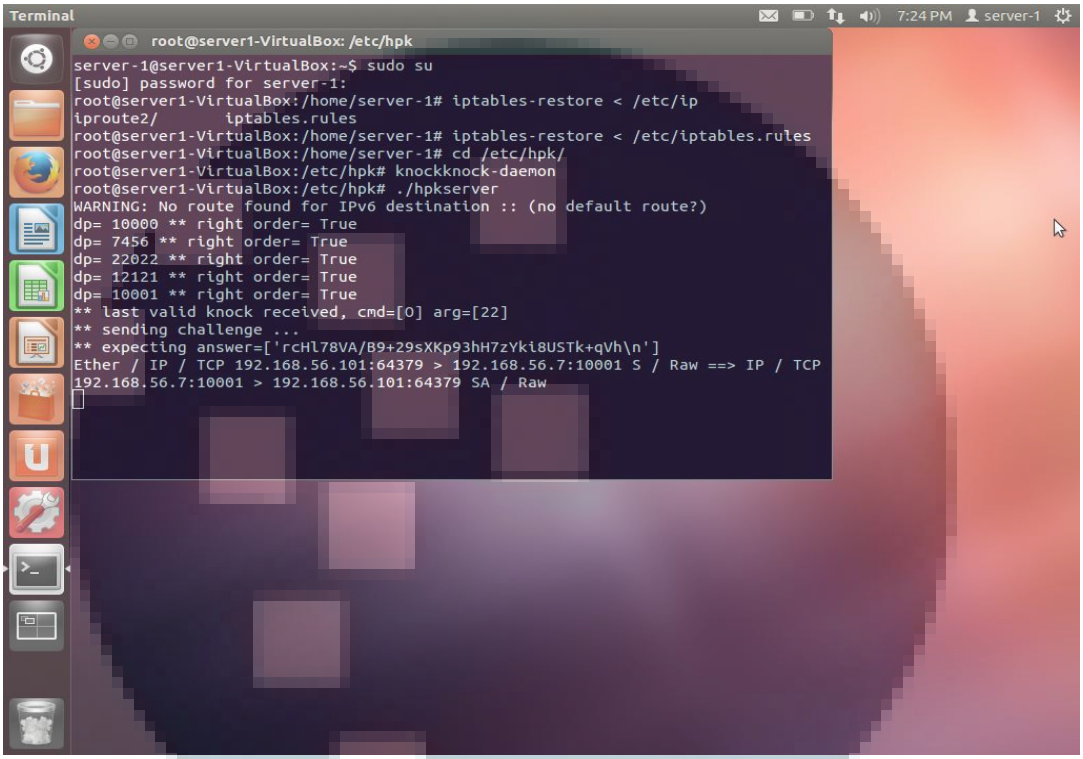
Gambar 3.8 *Server Configuration*



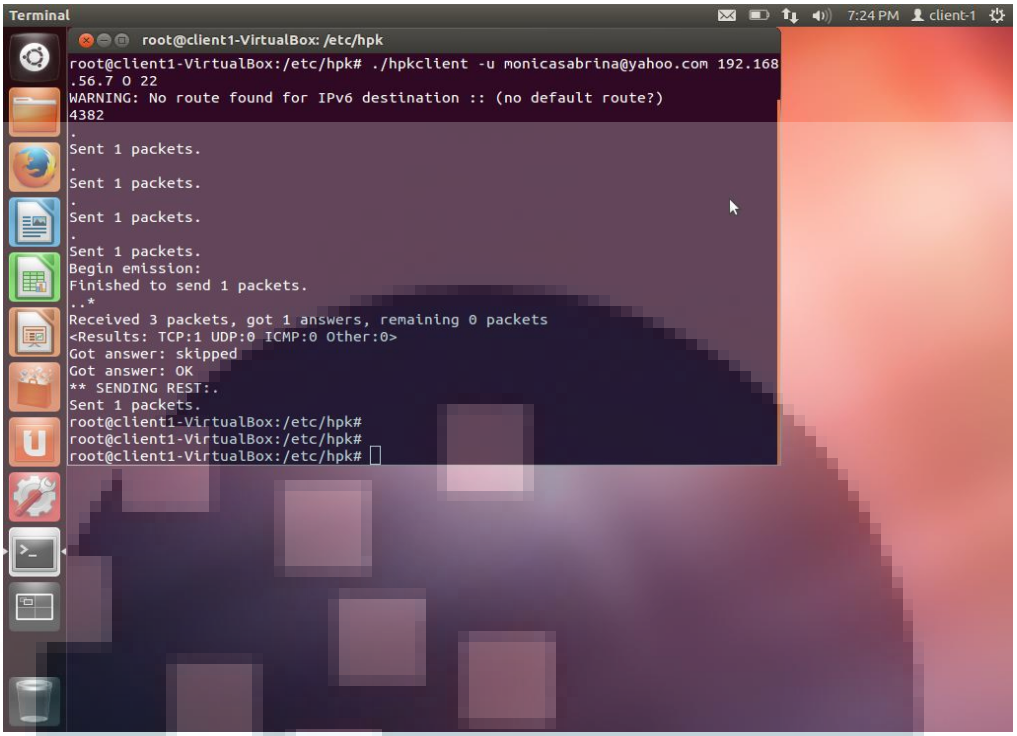
Gambar 3.9 *Client configuration*

Execute aplikasi *python HPKServer* dilakukan dengan mengetikkan command ./HPK*Server*, sedangkan untuk *client* adalah dengan mengetikkan ./HPK*Client* <command yang diinginkan misalnya –u untuk user id, -c untuk melihat konfigurasi> <nama domain *server*> <*client's request*>, contoh: *./hpkclient* –u

monicasabrina@yahoo.com 192.168.56.101 O 22, ini adalah command yang diketikkan *client* jika mengirim *request* untuk membuka *port* 22 ditunjukkan oleh gambar 3.10 dan 3.11

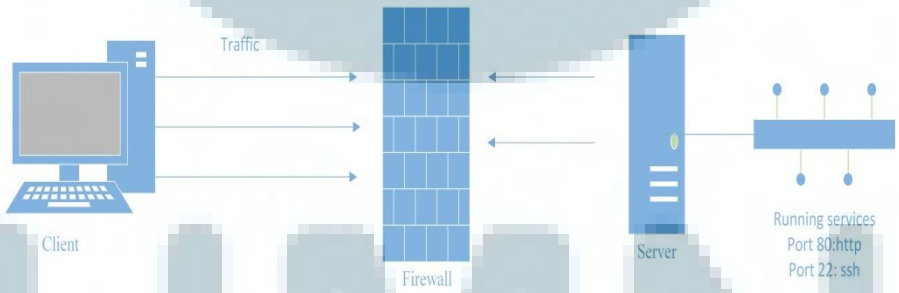


Gambar 3.10 command untuk *execute server script* melalui terminal



Gambar 3.11 *command execute client's request* untuk membuka *port* 22

3.7 Metode Pengujian



Gambar 3.12 Ilustrasi aplikasi *hybrid port knocking*

Semua *client* memiliki *python script* untuk menjalankan *hybrid port knocking* yang disimpan pada sebuah *folder* tertentu misalnya (/etc/hpk),begitu juga dengan *server*

Beberapa langkah yang harus dilakukan untuk melakukan pengujian, dijelaskan di bawah ini:

1. Komputer yang dilengkapi dengan aplikasi VirtualBox.
2. Dengan menggunakan *virtualbox,* dibuat 3 pasang *virtual machine* (*client-server* model) berbasis sistem operasi Linux 12.04.
3. Masing-masing *virtual machine* di-*install scapy, python imaging library, cython, line_profiler, memory_profiler.*
4. *Network Adapter* pada setiap *virtual machine* diatur dengan mode *host-only* dan menggunakan *static ip* agar berada dalam satu jaringan.
5. Setiap *virtual machine (*baik *client* maupun *server) meng-execute python script* yang diperlukan untuk menjalankan *hybrid port knocking*.
6. Pada komputer *client* juga *server* dicatat dan dianalisis kinerjanya
7. Uji coba dilakukan berdasarkan beberapa skenario yaitu *port knock* dengan *sequence* yang benar, *port knocking*dengan *sequence* yang salah, *port knock* yang benar namun *sequence*nya salah.