IMPLEMENTASI IPS DAN IDS MENGGUNAKAN APLIKASI BRO-IDS (INTRUSION DETECTION SYSTEM) YANG TERINTEGRASI DENGAN SMS GATEWAY

IMPLEMENTATIONS IPS AND IDS USING BRO-IDS (INTRUSION DETECTION SYSTEM) APPLICATION INTEGRATED WITH SMS GATEWAY

PROYEK AKHIR

Wahyu Febriyan Ramadhan
6302134017

PROGRAM STUDI D3 TEKNIK KOMPUTER
FAKULTAS ILMU TERAPAN
UNIVERSITAS TELKOM
BANDUNG, 2016

Atas izin Allah yang Maha Pengasih Lagi Maha Penyayang

Karya ini saya persembahkan untuk :

Kedua orangtua ku ayahanda Sugiarno dan Ibunda Zahroni Yusuf yang selalu memberikan dukungan do'a, semangat, moril dan materil.

Adik – adikku tersayang :

Wahyu Dwi Galuh sejati, Wahyu Tri Yulian Prabowo dan Wahyu Catur Prayoga terimakasih untuk dukungan dan do'anya

Seluruh keluarga besar terimakasih atas do'a dan dukungannya

LEMBAR PENGESAHAN PROYEK AKHIR

IMPLEMENTASI IPS DAN IDS MENGGUNAKAN APLIKASI BRO-IDS (INTRUSION DETECTION SYSTEM) YANG TERINTEGRASI DENGAN SMS GATEWAY

Penulis
Wahyu Febriyan Ramadhan
NIM 6302134017

_____

Pembimbing I
Tedi Gunawan, S.T., M.Kom.
NIP 14771574

_____

Pembimbing II
Setia Juli Irza Ismail, S.T., M.T.
NIP 15781712-1

_____

Ketua Program Studi
Hendri Rossi Andrian,ST.,M.T
NIP 09820562-1

_____

Tanggal Pengesahan: 2016

PERNYATAAN

Dengan ini saya menyatakan bahwa:

1. Proyek Akhir ini adalah asli dan belum pernah diajukan untuk mendapatkan gelar akademik (Ahli Madya, Sarjana, Magister dan Doktor), baik di Fakultas Ilmu Terapan Universitas Telkom maupun di perguruan tinggi lainnya;
2. karya tulis ini murni gagasan, rumusan, dan penelitian saya sendiri, tanpa bantuan pihak lain, kecuali arahan tim pembimbing atau tim promotor atau penguji;
3. dalam karya tulis ini tidak terdapat cuplikan karya atau pendapat yang telah ditulis atau dipublikasikan orang lain, kecuali secara tertulis dengan jelas dicantumkan sebagai acuan dalam naskah dengan menyebutkan nama pengarang dan dicantumkan dalam daftar pustaka;
4. saya mengijinkan karya tulis ini dipublikasikan oleh Fakultas Ilmu Terapan Universitas Telkom, dengan tetap mencantumkan saya sebagai penulis; dan

Pernyataan ini saya buat dengan sesungguhnya dan apabila pada kemudian hari terdapat penyimpangan dan ketidakbenaran dalam pernyataan ini maka saya bersedia menerima sanksi akademik berupa pencabutan gelar yang telah diperoleh karena karya tulis ini, serta sanksi lainnya sesuai norma yang berlaku di Fakultas Ilmu Terapan Universitas Telkom.

Bandung, 29 Agustus 2016

Pembuat pernyataan,

Wahyu Febriyan Ramadhan

KATA PENGANTAR

Puji syukur penulis haturkan kepada Allah SWT karena atas berkat rahmat dan hidayah-Nya sehingga penulis dapat menyelesaikan proyek akhir ini dengan baik. Proyek akhir ini bertujuan untuk memenuhi salah satu syarat kelulusan Program Diploma Tiga (D3) Teknik Komputer Fakultas Ilmu Terapan, Universitas Telkom.

Proyek akhir ini merancang sistem keamanan berbasis *IPS (Intrusion Prevention System)* dan *IDS* (*Intrusion Detection System)* yang berfungsi untuk melindungi *server* atau jaringan dari serangan penyusup atau ancaman dari orang yang tidak bertanggung jawab lainnya. Sistem keamanan ini menggunakan aplikasi *BRO Network Monitor Security* yang merupakan salah satu *tools* keamanan yang bersifat *open source*.

Penulis mengucapkan terima kasih kepada pihak – pihak yang membantu dalam menyelesaikan proyek akhir, khususnya kepada :

1. Orang tua yang telah memberikan dukungan, bantuan, dan perhatian untuk kelancaran proyek akhir ini;
2. Bapak Tedy Gunawan selaku dosen pembimbing I dan Bapak Setia Juli Irza Ismail selaku pembimbing II dalam pengerjaan proyek akhir;
3. Teman – teman baik yang selalu dengan sabar direpotkan dengan keluhan yang penulis punya seperti Adhy Pradana, Nurul Maryam, anggota kontrakan pink dan masih banyak yang lainnya dan tidak cukup lembar untuk disebutkan di sini.

Penulis mengharapkan proyek akhir ini dapat bermanfaat bagi pihak yang membutuhkan.

Bandung, 2016

Penulis

ABSTRAK

Banyaknya layanan publik yang memberikan akses jaringan kepada penggunanya tidak hanya memberikan rasa nyaman tetapi juga dapat menimbulkan tindak kejahatan jika tidak dikelola dengan baik, seperti pencurian data, pembajakan sistem, atau manipulasi sistem. *Bro Network Security Monitor* merupakan aplikasi *linux* yang berbasis *open source* . *Bro* berperan penting dalam keamanan jaringan karena memiliki fungsi *Intrusion Detection System* untuk mendeteksi serangan dan melaporkannya kepada *admin*, sehingga *admin* dapat bertindak cepat untuk mengatasi serangan tersebut. Hasil serangan yang terdeteksi oleh *Bro* akan tercatat di dalam *log file*, dan akan diteruskan dengan notifikasi ke *handphone admin* yang telah di konfigurasi pada *IFTTT.*

Kata Kunci: *Bro Network Security Monitor*, *Intrusion Detection System*, Keamanan Jaringan,IFTTT

ABSTRACT

Many public service which provided acces to the user network not only give a sense of comfortable but can aslo crime act if not managed well, such as data stealing, hijacking system, or manipulated system. Bro Network Security Monitor is linux application based of open source. Bro play an important role in network security for having function Intrusion Detection System to detect attack and report to admin, so admin can act quickly to overcome the attack. The attack attacked by bro will be recorded in logs file , and will be continued with notification admin to handphone which has a directive on ifttt

Keywords : Bro Network Security, Instrusion Detection System, Network Security, IFTTT

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

DAFTAR LAMPIRAN

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi informasi dalam berbagai bidang dewasa ini sangatlah signifikan, baik dalam bidang pendidikan, bisnis, keamanan, dan lain sebagainya. Salah satu teknologi yang berkembang saat ini adalah teknologi *Wifi* (*Wireless Fidelity)*. Dalam kehidupan modernisasi saat ini internet merupakan salah satu kebutuhan pokok manusia, baik untuk pekerjaan, pendidikan maupun untuk hiburan. Untuk itu tak sedikit layanan publik yang dipasang teknologi *wireless* seperti taman, *café*, terminal, bandara, stasiun, perpustakaan dan lain sebagainya, akan tetapi sebagian dari tempat tersebut tidak mempedulikan keamanan layanan yang diberikan. Sehingga layanan yang mereka berikan biasanya mudah *down* atau diserang oleh pengunjung yang jahil dan hasilnya pun tidak hanya merugikan pengunjung lain tapi juga merugikan pengelola tersebut. Untuk itu dibutuhkan sebuah sistem yang dapat memberikan informasi mengenai layanan yang sedang dikelola sehingga kerusakan jaringan dapat diminimalisir.

Fitur *IPS* dan *IDS* ini dapat meningkatkan keamanan dalam *server* sehingga dapat mempersulit serangan yang dilakukan terhadap *server*. *IPS* dapat mencegah serangan yang berlangsung, dan *IDS* dapat memberikan notifikasi kepada *admin* dan mencatat informasi si penyerang dan memasukkan ke dalam *log files.*

Salah satu aplikasi keamanan yang mendukung *IPS* dan *IDS* ini adalah *BRO-IDS* (*Intrusion Detection System*). Aplikasi ini merupakan aplikasi *open source* sehingga dapat dikembangkan dengan bebas oleh para penggunanya. Kelebihan dari aplikasi ini, dapat me*monitoring* jaringan secara *real-time* dan memberikan notifikasi kepada *admin* pengelola jaringan tersebut.

1.2 Rumusan Masalah

Berdasarkan latar belakang masalah di atas maka rumusan masalahnya adalah :

1. Bagaimana cara menerapkan *IPS* dan *IDS* pada jaringan atau *server* dengan aplikasi *Bro* ?
2. Bagaimana cara mengintegrasikan *IPS* dan *IDS* menggunakan aplikasi *Bro* dengan *sms gateway* ?
3. Bagaimana memberikan notifikasi ke *admin* mengenai jaringan yang dikelola?

1.3 Tujuan

Tujuan dari penelitian ini adalah :

1. Mengkonfigurasi *IPS* dan *IDS* pada *server* menggunakan aplikasi *Bro* pada *linux CentOS 7*
2. Mengintegrasikan *sms gateway* pada *IPS* dan *IDS* yang telah dikonfigurasi dengan aplikasi *Bro*
3. Memberikan notifikasi kepada *admin* mengenai jaringan yang dikelola saat terjadi serangan maupun saat tidak terjadi serangan.

1.4 Batasan Masalah

Permasalahan yang akan diteliti dibatasi agar penelitian terfokus pada tujuan yang ingin dicapai. Batasan permasalahan dalam penelitian ini adalah :

1. Perangkat lunak yang digunakan dalam membuat sistem ini adalah *Bro*, sedangkan untuk *sms gateway*-nya menggunakan aplikasi *third-party* atau *sms gateway open source* lainnya
2. Sistem Operasi yang digunakan *server Linux CentOS 7 64 bit*

3. Jumlah perangkat yang digunakan sebanyak 3 *PC* (1 sebagai *server* dan 2 sebagai *client*)
4. *Access point* menggunakan *handphone admin*
5. Menggunakan *IP dynamic*
6. Menggunakan aplikasi *Bro-ids* 2.4.1
7. Fitur *Bro-ids* yang digunakan mendeteksi serangan yang berlangsung
8. Fitur *Bro-ids* yang digunakan me*monitoring* jaringan
9. Fitur *Bro-ids* yang digunakan mengirim notifikasi serangan ke *email* lalu diteruskan ke *hanpdhone admin*
10. Pada proyek akhir ini lebih fokus terhadap fungsi *Bro-ids* sebagai *IDS*
11. Menggunakan serangan *DOS* (*Distribute of Service*)
12. Menggunakan serangan *Port Scanner*
13. Menggunakan serangan *FTP Brute-force*
14. Mengirim notifikasi ke *email admin*
15. Mengirim notifikasi ke *handphone admin* saat terjadi serangan
16. Waktu masuknya notifikasi ke *Handphone admin* tergantung dari jaringan yang digunakan.

1.5 Definisi Operasional

Bro Network Monitor Security adalah sebuah aplikasi *open source* yang dikembangkan untuk melindungi *server* ataupun jaringan berbasis *IDS*, aplikasi ini tidak hanya mampu melindungi, tetapi dapat juga menganalisis trafik jaringan. Karena aplikasi ini bersifat *open source*, maka pengguna secara bebas dapat mengembangkan aplikasi ini.

Aplikasi *third-party* adalah : sebuah program perangkat lunak atau perangkat keras yang dikembangkan oleh pihak ketiga yang mengacu pada sistem operasi atau program tertentu untuk mendukung kinerja dari sitem operasi atau program tersebut. Biasanya aplikasi *third party* ini adalah *plugin* atau perangkat lunak untuk aplikasi tertentu.[6]

IFTTT (IF THIS THEN THAT) adalah sebuah aplikasi yang memungkinkan kita untuk menghubungkan 2 aplikasi *web* menjadi satu. Secara sederhana, setiap aktivitas yang kita lakukan secara *online* dapat diatur untuk memiliki reaksi otomatis pada sebuah aplikasi *web* tertentu. *IFTTT* adalah rantai yang dapat menggabungkan aplikasi tersebut dan dapat saling mendukung satu sama lainnya. Cara kerja *IFTTT* ini pada dasarnya membutuhkan 2 hal yaitu pemicu **(*trigger)*** dan reaksi ***(resulting action).*** [7]

SMS Gateway adalah sebuah sistem aplikasi yang digunakan untuk mengirim atau menerima pesan dari pihak tertentu, aplikasi ini biasanya digunakan untuk melakukan *broadcast* atau promosi kepada kalangan terkait. Untuk *sms gateway* ini dapat berupa jaringan *GSM* atau *CDMA* tergantung dari pemilik sistem tersebut.

1.6 Metode Pengerjaan

Metodologi yang dilakukan untuk menyelesaikan implementasi *IPS* dan *IDS* menggunakan *BRO-IDS* yang terintegrasi dengan *sms gateway* adalah :

a. Studi Pustaka

   Studi pustaka dilakukan dengan cara mencari dan pegumpulan data – data, teori dan informasi yang diambil dari buku – buku yang ada hubungannya dengan masalah yang akan dibahas dalam pengerjaan.

b. Perancangan Sistem

   Pada tahap ini dilakukan perancangan dan pemodelan pada sistem yang akan diuji serta perangkat keras (*hardware*) dan perangkat lunak (*software*) dan kemungkinannya untuk diimplementasikan.

c. Implementasi

Implementasi sistem yang dilakukan sesuai dengan hasil analisa dan perancangan desain sistem. Mengumpulkan data – data parameter yang telah ditentukan dari pengujian implementasi.

d. Pengujian

Melakukan pengujian terhadap sistem yang dibuat apakah sistem sudah berjalan seperti yang diinginkan pada topologi yang telah direncanakan.

e. Analisis pengujian dan penarikan kesimpulan

Melakukan analisis pengujian yang telah didapatkan dari hasil pengujian

f. Penyusunan Laporan

Mendokumentasikan secara keseluruhan atas kegiatan yang telah dilakukan dalam pengerjaan proyek akhir.

1.7 Jadwal Pengerjaan

Tabel 1- 1 Jadwal pengerjaan

Kegiatan	2016															
	April				Mei				Juli				Juli			
	1	2	3	4	1	2	3	4	1	2	3	4	1	2	3	4
Studi Pustaka																
Perancangan Sistem																
Implementasi																
Pengujian																
Analisis Pengujian dan Penarikan Kesimpulan																
Penyusunan Laporan																

BAB 2
TINJAUAN PUSTAKA

2.1 Definisi BRO

Bro adalah sebuah aplikasi *open source* yang berbasis *network* analisis yang digunakan untuk keamanan jaringan atau keamanan data. Aplikasi ini dapat melakukan *monitoring* secara *real-time* dan memiliki perfomansi yang tinggi, selain itu dapat melakukan analisis terhadap *protokol – protokol* pada jaringan, sehingga aplikasi ini banyak digunakan untuk *intrusion detection system* (*IDS*) dan *network analiysis framework* pada keamanan jaringan. [1]

Prinsip kerja dari aplikasi *Bro* adalah dapat berbasis *signature* atau berbasis *anomaly* tergantung dari konfigurasi yang dilakukan. Perbedaan antara *signature* dengan *anomaly* adalah jika *signature* merupakan *IDS* yang dibuat sesuai dengan cara penyerangan atau penyusupan sedangkan *anomaly* merupakan *IDS* yang mengikuti pola kebiasaan pada sebuah jaringan, yaitu *IDS* akan mencocokan keadaan normal sebuah jaringan pada umumnya, jika pada jaringan tidak sesuai dengan keadaan normal pada umumnya maka *IDS* akan menjalankan fungsinya. Dalam hal ini metode *anomaly* dapat dikatakan sebagai metode *true and false*.

2.2 Kelebihan dan Kekurangan BRO

Kelebihan *BRO* :

a. Dapat melakukan identifikasi secara detail terhadap protokol dan memberikan informasi secara spesifik terhadap permasalahan yang terjadi pada sebuah sistem, sehingga dapat membantu *admin* atau pengguna untuk segera bertindak

b. Memiliki kemampuan yang tinggi dalam analisis setiap protokol dan memberikan informasi secara *real-time* sehingga sangat baik untuk *IDS.*

Kekurangan *BRO* :

a. Tidak disediakannya tampilan *GUI* dalam mengkonfigurasi *Bro*, untuk menambahkan tampilan *GUI* harus memasang *framework* tertentu terlebih dahulu

b. Memerlukan pengetahuan tentang bahasa pemprograman untuk mengkonfigurasi *Bro*, ditambah lagi *Bro* memiliki bahasanya sendiri yaitu *BroScript.*

2.3 Definisi IPS (Intrusion Prevention System)

Intrusion Prevention System (*IPS*) adalah sebuah sistem yang bekerja untuk me-*monitoring traffic* jaringan, mendeteksi aktivitas mencurigakan, dan melakukan pencegahan dini terhadap *intrusi* atau ancaman yang dapat membuat jaringan berjalan tidak sebagaimana mestinya. *IPS* ini dapat mem-*block packet* pada jaringan saat terjadi serangan [2]. *Intrusion Prevention System* dianggap sebagai pengembangan dari *Intrusion Detection System*, keduanya bekerja untuk memonitor lalu lintas jaringan dari aktivitas yang membahayakan jaringan. Tidak seperti *IDS*, karena *IPS* mampu mencegah serangan yang datang dengan bantuan *admin*istrator secara minimal atau bahkan tidak sama sekali.

Untuk *IPS* sendiri memiliki 4 jenis yakni :

1. *Network-based Intrusion Prevention System*, memiliki fungsi yaitu untuk memantau seluruh jaringan dari lalu lintas yang mencurigakan dengan menganalisa aktivitas protokol.
2. *Wireless Intrusion Preventions System*, memiliki fungsi yaitu untuk memonitor jaringan nirkabel dari lalu lintas yang mencurigakan dengan menganalisa protokol dari jaringan nirbakel.
3. *Network Behavior Analysis* (NBA), memeriksa lalu lintas jaringan untuk mengidentifikasi ancaman yang menghasilkan arus lalu lintas yang tidak biasa, seperti *Distributed Denial of Service attacks*, beberapa bentuk *malware*, dan pelanggaran lainnya.

4. *Host-based Intrusion Preventions System*, yaitu perangkat lunak yang di pasang untuk memonitor sebuah *host* untuk kegiatan yang mencurigakan dengan menganalisis peristiwa yang terjadi di dalam *host* tersebut. [3]

2.4 Definisi IDS (Intrusion Detection System)

Intrusion Detection System adalah sistem yang melakukan pengawasan terhadap traffik jaringan dan pengawasan terhadap kegiatan – kegiatan yang mencurigakan di dalam sebuah jaringan. Jika ditemukan kegiatan – kegiatan yang mencurigakan yang berhubungan dengan traffik jaringan maka *IDS* akan memberikan peringatan kepada sistem atau *admin*istrator jaringan. *IDS* dapat melakukan analisis dan mencari bukti percobaan penyusupan atau peretasan. *IDS* menggunakan teknik *anomaly-based* atau menggunakan *signature* untuk melakukan pengamanan, tapi bisa juga dengan kedua teknik tersebut disatukan.

Untuk *IDS* memiliki 2 jenis yakni :

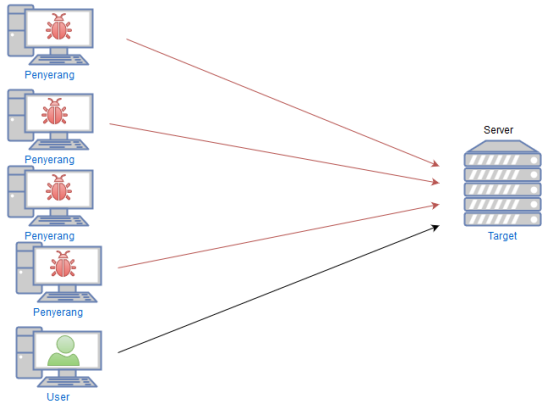
1. *Network-based Intrusion Detection System* (*NIDS*), yaitu semua lalu lintas yang mengalir ke sebuah jaringan akan dianalisis untuk mencari apakah ada percobaan serangan atau penyusupan ke dalam sistem jaringan. Kelemahan dari *NIDS* ini, bahwa *NIDS* agak rumit diimplementasikan dalam sebuah jaringan yang menggunakan *switch ethernet*, meskipun beberapa vendor *switch ethernet* telah menerapkan fungsi *IDS* di dalam *switch* buatannya untuk memonitor *port* atau koneksi.
2. *Host-based Intrusion Detections System* (*HIDS*), aktivitas sebuah *host* jaringan individual akan dipantau apakah terjadi sebuah percobaan serangan atau penyusupan ke dalamnya atau tidak.

2.5 Definisi SMS Gateway

SMS Gateway adalah sebuah sistem aplikasi 2 arah yang dapat menerima dan mengirim pesan baik berupa tulisan ataupun gambar ke *client* atau pihak tertentu, pesan yang dikirimkan bisa pesan pribadi (*private*) atau umum (*broadcast*). Aplikasi yang banyak digunakan untuk membuat *sms gateway* yang terkenal adalah *gammu*. *Gammu* merupakan *tools open source* yang dapat di-*install* ke dalam

sistem komputer baik *OS Linux* ataupun *Windows* dan dapat dikonfigurasikan secara mudah oleh pengguna. Dalam membangun *SMS gateway* diperlukan sebuah *database* yang digunakan untuk menampung pesan maupun nomor tujuan.

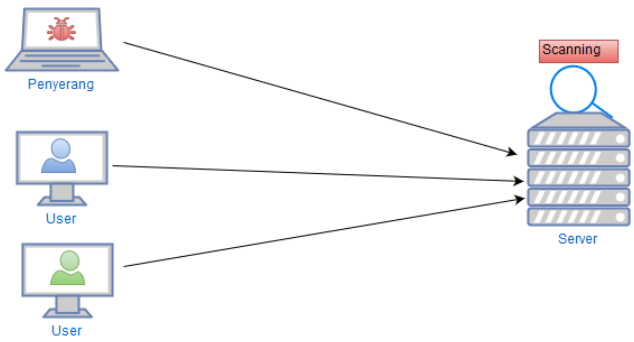
2.6 Denial of Service (DOS)



Gambar 2- 1 DOS (Denial of Service)

DOS (Denial of Service) merupakan sebuah serangan yang mengirimkan paket secara terus menerus pada sebuah *server* atau jaringan sehingga membuat *server* atau jaringan tersebut menjadi *down* atau *crash*. Serangan *DOS (Denial of Service)* merupakan serangan paling sederhana dari jenis serangan lainnya karena hanya mengirimkan *request* kepada *server*, jenis paket yang dikirimkan dapat ditentukan oleh penyerang.

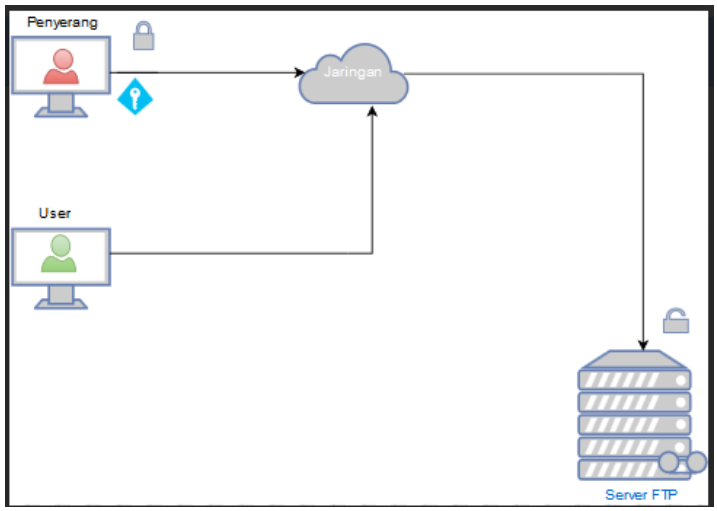
2.7 Port Scanner



Gambar 2- 2 Serangan Port Scanner

Port scanner adalah sebuah aktivitas yang memeriksa *port* yang terbuka pada jaringan *TCP* dan *UDP* yang dilakukan dengan tujuan mengetahui *port* yang terbuka. Tujuan dari *port scanning* ini bagi penyerang yaitu untuk menyusup ke dalam sebuah *server* atau jaringan guna mencuri atau melumpuhkan *server*, sedangkan bagi *admin* berguna untuk memeriksa kembali jaringan yang terbuka atau kelemahan dari *server* yang ditangani. [4]

2.8 *FTP Brute-force*



Gambar 2- 3 *FTP Brute-force*

Brute-force adalah sebuah teknik serangan terhadap sebuah sistem keamanan komputer yang menggunakan percobaan terhadap semua kemungkinan yang menjadi sebuah kunci masuk untuk mengakses sebuah sistem atau mengusai sebuah akun tertentu. Pada umumnya serangan *brute-force* biasanya menyerang protokol *telnet, http, https, SMB, POP3, IMAP,* dan lain sebagainya. [5]

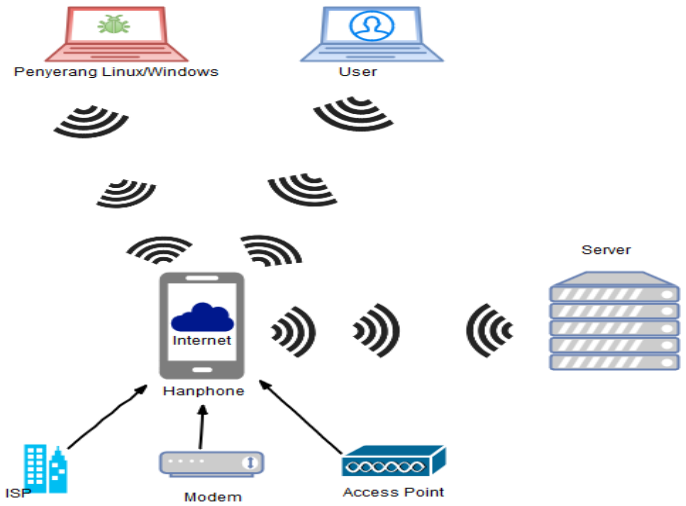
FTP (File Transfer Protokol) adalah sebuah protokol yang berfungsi sebagai tempat pertukaran data atau *file* tertentu pada sebuah jaringan, untuk mengakses *protokol* ini setiap pengguna memiliki *username* dan *password* masing – masing. Protokol ini banyak digunakan oleh perusahaan besar untuk saling bertukar data antar karyawan atau pemimpin dengan karyawan.

BAB 3

ANALISIS DAN PERANCANGAN

3.1 Topologi Jaringan Sebelum Implementasi Bro

Penerapan topologi sebelum implementasi *IPS* dan *IDS* yang terintegrasi dengan *sms gateway*, dalam hal ini biasanya pengelola hanya mendesain topologi tanpa mementingkan keamanan layanan tersebut, sehingga pada saat penyerangan dilakukan *admin* atau pengguna terlambat melakukan tindakan sehingga kerusakan menjadi lebih parah.



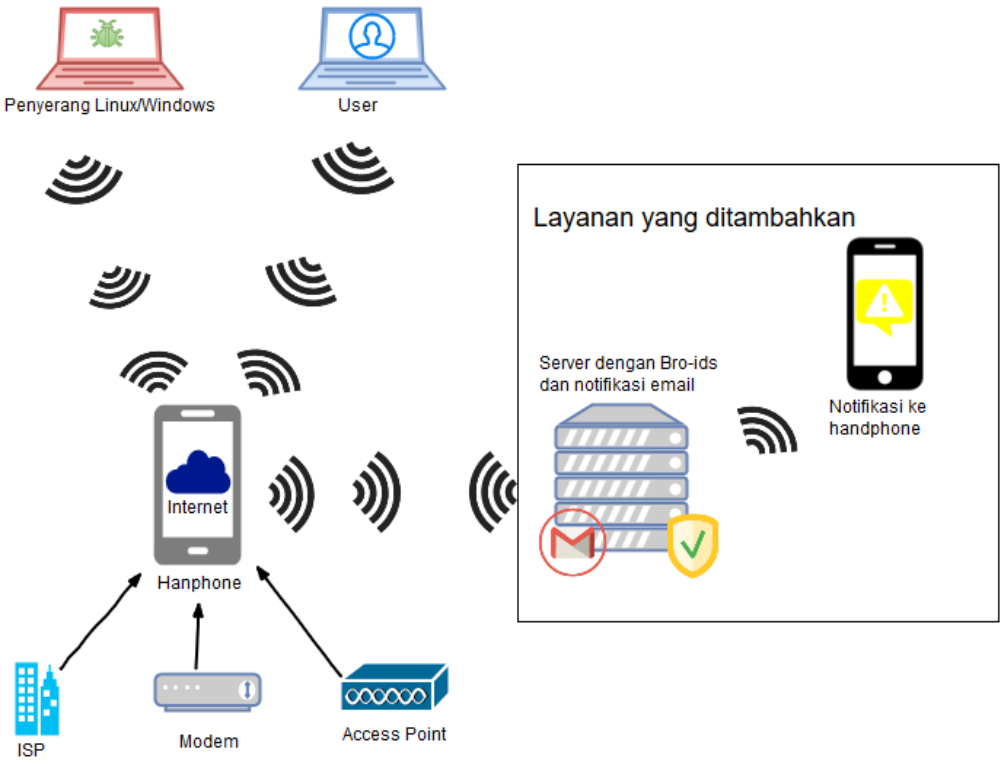
Gambar 3- 1 Topologi sebelum implementasi Bro

Pada gambar 3-1 menunjukkan bahwa tidak adanya keamanan pada jaringan, sehingga bisa terjadi penyusupan dan pencurian *file* pada *server*.

3.2 Topologi Jaringan Setelah diberikan Implementasi Bro

Penerapan *sms gateway* di dalam *IPS* dan *IDS* ini sangat penting untuk meningkatkan keamanan pada jaringan yang dikelola, untuk itu diperlukan aplikasi atau sistem yang dapat melakukan hal tersebut sehingga keamanan data dan jaringan semakin meningkat, dan menghambat penyusupan ke dalam jaringan yang

dikelola. Sehingga jika ada ancaman, maka *admin* atau pengguna dapat bertindak lebih cepat untuk menghindari hal – hal yang tidak diinginkan.



Gambar 3- 2 Topologi setelah implementasi Bro

3.3 Analisis Kebutuan Sistem

Kebutuan produk yang dibutuhkan dalam masalah ini adalah :

a. Dibutuhkan aplikasi *IDS* yang dapat bekerja secara *real-time*

b. Aplikasi *IDS* tersebut mendukung terhadap sistem yang sedang dikelola oleh *admin*

c. Aplikasi dapat dibuat dengan biaya yang rendah.

Untuk itu kebutuhan *hardware* dalam membangun sistem keamanan ini ialah sebagai berikut :

1. *Server*

Berikut merupakan spesifikasi minimum dan spesifikasi yang digunakan untuk membangun *server* menggunakan *CentOS 7 64-bit*.

Tabel 3- 1 Spesifikasi minimum *server*

NO	Perangkat Keras	Spesifikasi
1.	Processor	Pentium III or higher 500MHz or higher
2.	RAM	*3 GB on 32 bit system, 200 MB of disk space minimum*
3.	Harddisk	*2 GB minimum for large environment*

Tabel 3- 2 Spesifikasi yang digunakan

NO	Perangkat Keras	Spesifikasi
1.	Processor	Intel® Core™ i3-2370M CPU @2.40GHz 2.40 GHz
2.	RAM	*6 GB*
3.	Harddisk	*250 GB*

2. Penyerang

Berikut merupakan spesifikasi minimum penyerang dan spesifikasi yang digunakan untuk menjalankan aplikasi serangan pada sistem operasi penyerang.

Tabel 3- 3 Spesifikasi minimum penyerang

NO	Perangkat Keras	Spesifikasi
1.	Processor	*32-bit or 64-bit processor*
2.	RAM	*1 GB*
3.	Harddisk	*40 GB*

Tabel 3- 4 Spesifikasi yang digunakan

NO	Perangkat Keras	Spesifikasi
1.	Processor	Intel® Core™ i3-2370M CPU @2.40GHz 2.40 GHz
2.	RAM	*4 GB*
3.	Harddisk	*500 GB*

3. Pengguna

Berikut merupakan spesifikasi minimum dan yang digunakan oleh pengguna untuk mengakses *server* atau untuk saling bertukar data.

Tabel 3- 5 Spesifikasi minimum pengguna

NO	Perangkat Keras	Spesifikasi
1.	Processor	*32-bit or 64-bit*
2.	RAM	*256 MB*
3.	Harddisk	*50 GB*

Tabel 3- 6 Spesifikasi yang digunakan

NO	Perangkat Keras	Spesifikasi
1.	Processor	*Intel® Core™ i3-2370M CPU @2.40GHz 2.40 GHz*
2.	RAM	*2 GB*
3.	Harddisk	*500 GB*

4. Handphone Admin

Berikut merupakan spesifikasi minimum dan juga spesifikasi *handphone* dari penulis yang dijadikan sebagai tempat notifikasi saat terjadi serangan.

Tabel 3- 7 Spesifikasi minimum

NO	Perangkat Keras	Spesifikasi
1.	Processor	*Android 4.0 (Ice Cream) IOS 7.0 for Iphone*
2.	RAM	*512 MB*
3.	Memory Card	*25 MB free space*
4.	Sim Card	*All sim card support internet*

Tabel 3- 8 Spesifikasi yang digunakan

NO	Perangkat Keras	Spesifikasi
1.	Processor	*Android 5.0 (Lolipop)*
2.	RAM	*1 GB*
3.	Memory Card	*4 GB*

5. Bro-IDS

Berikut merupakan spesifikasi minimum dan yang digunakan untuk menjalankan aplikasi Bro-IDS pada *server*.

Tabel 3- 9 Spesifikasi minimum Bro-IDS

NO	Perangkat Keras	Spesifikasi
1.	Processor	1 GHz CPU
2.	Sistem Operasi	FreeBSD 5.x
3.	RAM	*512 MB*
4.	Harddisk	*10 GB for small network*

Tabel 3- 10 Spesifikasi yang digunakan

NO	Perangkat Keras	Spesifikasi
1.	Processor	*Intel® Core™ i3-2370M CPU @2.40GHz 2.40 GHz*
2.	Sistem Operasi	*Linux CentOS 7 64 bit*
3.	RAM	*6 GB*
4.	Harddisk	*500 GB*

Kebutuhan perangkat lunak dalam pengerjaan proyek akhir ini baik pada sisi *server*, pengguna, maupun penyerang sebagai berikut :

Tabel 3- 11 Kebutuan perangkat lunak

NO	Jenis Perangkat	Spesifikasi
1.	Sistem Operasi	*Linux CentOs 7 64 bit*
2.	Sistem Operasi	*Windows 10*
3.	Sistem Operasi	*Ubuntu 14.04 LTS*
4.	Perangkat Lunak	*IFTTT*
5.	Perangkat Lunak	*Win32DiskImager-0.9.5*
6.	Aplikasi Penyerang	*Nmap*
7.	Aplikasi Penyerang	*Hydra*
8.	Aplikasi Penyerang	*Hping3/LOIC*
9.	Aplikasi Email	*Sendmail*
10.	Aplikasi FTP	*VSFTPD*

Kebutuhan perangkat keras dalam pengerjaan proyek akhir ini baik dari sisi *server*, pengguna, maupun penyerang adalah sebagai berikut :

Tabel 3- 12 Kebutuhan perangkat keras

Processor	RAM	Sistem Operasi	Jumlah
Intel® Core™ i3-2370M CPU @2.40GHz 2.40 GHz	6 GB	Linux CentOS 7 64-bit	1
Intel® Core™ i3-2370M CPU @2.40GHz 2.40 GHz	4 GB	Ubuntu 14.04 LTS	1
Intel® Core™ i3-2370M CPU @2.40GHz 2.40 GHz	2 GB	Windows 10 64-bit	1
Intel Atom Z2520 Dual-core 1.2 GHz (Asus Zenfone 4)	1 GB	Android 5.0 (Lolipop)	1

3.4 Langkah – Langkah Pengerjaan

Langkah – langkah yang dilakukan untuk mengerjakan proyek akhir ini sebagai berikut :

3.4.1 Langkah Pengerjaan dari Sisi *Server*

a. Melakukan pemasangan sistem operasi *Linux CentOS 7 64 bit* pada perangkat laptop yang akan dijadikan sebagai *server*

b. Melakukan konfigurasi *FTP* pada *server* agar dapat diakses oleh *user*

c. Melakukan konfigurasi *IPS* dan *IDS* pada perangkat yang ter-*install linux CentOS*

d. Melakukan konfigurasi menggunakan aplikasi *sendmail* yang kemudian akan diintegrasikan dengan *gmail* sebagai tempat pengolah pesan

e. Membuat *user* untuk *client* agar bisa mengakses *FTP*.

3.4.2 Langkah Pengerjaan dari Sisi *Admin*

a. Meng-*install* aplikasi *IFTTT* pada *handphone admin*

b. Membuat *recipe* untuk mengirim notifikasi dari *gmail* jika terjadi pemberitahuan dari *Bro*

c. Meingntegrasikan *gmail* dengan *aplikasi IFTTT* Sebagai *sms gateway.*

3.4.3 Langkah Pengerjaan dari Sisi User

a. Melakukan konektivitas terhadap *server*

b. Mengakses *FTP* pada *server*

3.4.4 Langkah Pengerjaan dari Sisi Penyerang

a. Melakukan instalasi aplikasi untuk menyerang seperti *nmap*, *hping3* atau *LOIC*, dan *hydra*

b. Melakukan serangan *port scanning* menggunakan aplikasi *nmap*, yang berfungsi untuk mengetahui celah keamanan yang terbuka seperti *port* yang terbuka, *operating system* yang digunakan serta hal lainnya

c. Melakukan serangan *FTP Brute-force* menggunakan aplikasi *hydra*, yang berfungsi untuk mendapatkan *password* serta *username user* agar dapat mengakses *FTP* pada *server*

d. Melakukan serangan *DOS* guna melumpuhkan *server* menggunakan aplikasi *hping3* atau *LOIC*.

3.5 Rencana Pengujian

Berikut merupakan rencana – rencana pengujian yang dilakukan pada proyek akhir ini yakni :

3.5.1 Rencana Pengujian dari Sisi *Server*

Rencana pengujian pada sisi *server* yaitu melakukan instalasi dan konfigurasi layanan yang dibutuhkan agar dapat diakses oleh *user*, berikut instalasi dan konfigurasi tersebut :

a. Memasang sistem operasi *Linux CentOS 7 64-bit* pada perangkat laptop sebagai *server*

b. Memasang dan menerapkan *IPS* dan *IDS* menggunakan aplikasi *Bro* pada laptop *server*

c. Memasang *FTP server* pada *server*

d. Mengkonfigurasi *FTP server* agar dapat diakses oleh *user*

e. Mengkonfigurasikan *sendmail* pada *server* untuk diintegrasikan ke *email*

f. Mengirim pesan notifikasi jika terjadi serangan ke *handphone admin*

g. Memantau trafik pada jaringan saat terjadi serangan.

h. Menggunakan *rule scan.bro, synflood.bro*, dan *ftp-bruteforce.bro*. Untuk *rule*-nya dapat dilihat pada lampiran 6, lampiran 7 dan lampiran 8

3.5.2 Rencana Pengujian dari Sisi *Admin*

Rencana pengujian dari sisi *admin* yaitu untuk mengintegrasikan antara *server* dengan perangkat yang dimiliki oleh *admin*, agar notifikasi dari *server* dapat diterima oleh *admin*, berikut konfigurasi tersebut :

a. Melakukan *registrasi* ke *website IFTTT*

b. Membuat *recipe* untuk mengintegrasikan *email* pada aplikasi *IFTTT* untuk mengirim pesan notifikasi ke *handphone admin*

c. Meng-*install* aplikasi *IFTTT* pada *handphone admin*

d. Mengintegrasikan aplikasi *IFTTT* pada *handphone admin* dengan *IFTTT server* agar notifikasi dapat terkirim ke *handphone admin*

3.5.3 Rencana Pengujian dari Sisi User

Rencana pengujian dari sisi *user* yaitu *user* dapat mengakses layanan yang diberikan oleh *server*.

a. Mengakses fitur *FTP* pada *server*

b. Melakukan *download* dan *upload file* pada *FTP server*

3.5.4 Rencana Pengujian dari Sisi Penyerang

Rencana pengujian dari sisi penyerang yaitu untuk melakukan serangan ke *server* dan mendapatkan informasi – informasi penting mengenai *server* tersebut.

a. Melakukan Instalasi aplikasi penyerangan seperti *nmap*, *hydra*, dan *hping3* atau *LOIC*

b. Melakukan serangan menggunakan metode *port scanning*

c. Melakukan serangan dengan metode *FTP Brute-force.*

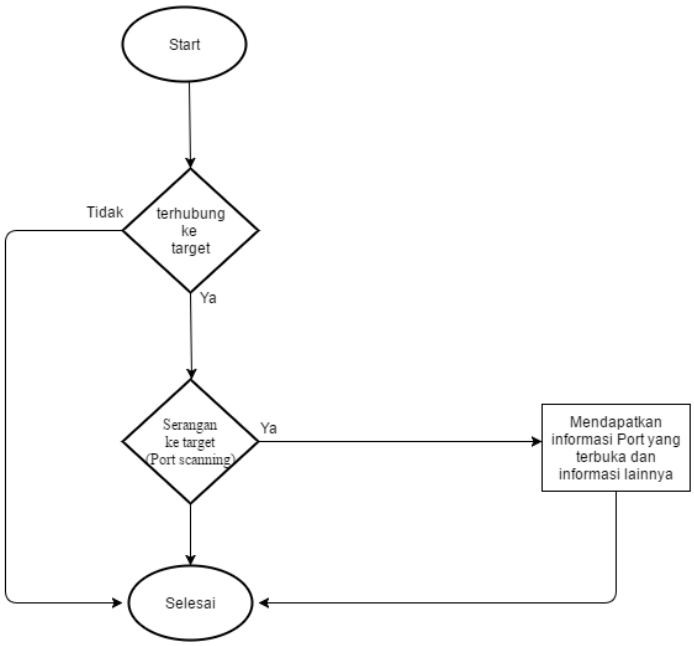
3.6 Skenario Pengujian

Pada bagian ini akan ada 2 skenario pengujian yang akan dilakukan pada sisi penyerang yaitu skenario penyerangan tanpa menggunakan *Bro* dan skenario penyerangan menggunakan *Bro*.

3.6.1 Berikut adalah Scenario Penyerangan Tanpa Menggunakan Bro

Berikut adalah skenario pengujian tanpa menggunakan *Bro* :

1. *Port Scanning*

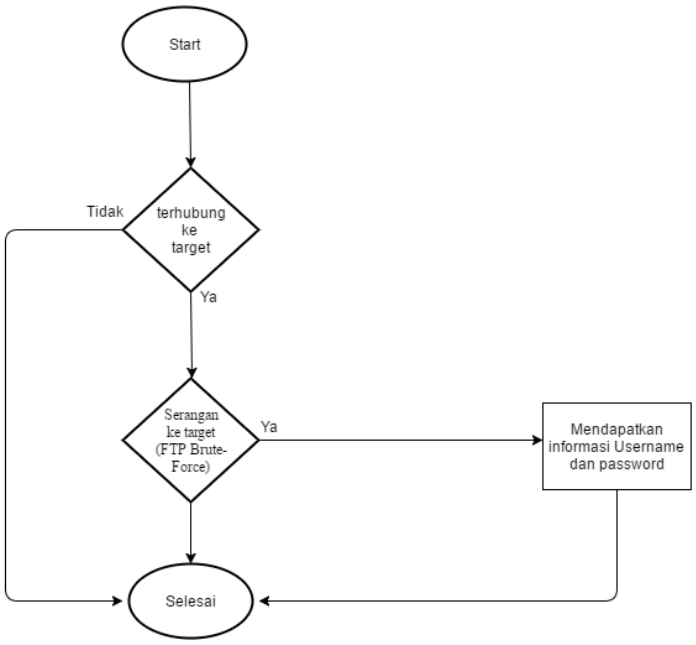


Gambar 3- 3 Flowchart serangan port scanning tanpa Bro

Keterangan :

1. Penyerang mencoba terkoneksi dengan *server*, dengan melakukan *ping* terhadap *server* apakah penyerang terkoneksi atau tidak dengan *server*
2. Setelah itu penyerang melakukan serangan *port scanning* menggunakan aplikasi *nmap* kepada *server*
3. Setelah serangan dilakukan penyerang mendapatkan informasi *port* yang terbuka dan informasi lainnya seperti *OS* yang digunakan, dan layanan lainnya.

2. *FTP Brute-force*

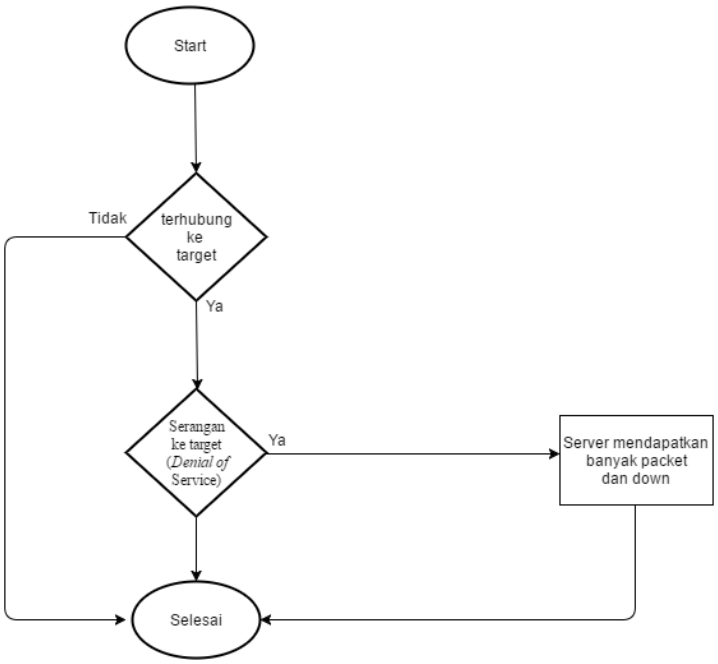


Gambar 3- 4 Flowchart serangan *FTP Brute-force* tanpa menggunakan Bro

Keterangan :

1. Penyerang mencoba terkoneksi dengan *server*, dengan melakukan *ping* terhadap *server* apakah penyerang terkoneksi atau tidak dengan *server*
2. Setelah itu penyerang melakukan serangan *FTP Brute-force* menggunakan aplikasi *hydra* kepada *server*
3. Setelah itu penyerang mendapatkan *username* dan *password* yang akan digunakan untuk mengakses *FTP server*.

3. *Denial of Service (DOS)*



Gambar 3- 5 Flowchart serangan Denial of Service (DOS) tanpa menggunakan Bro

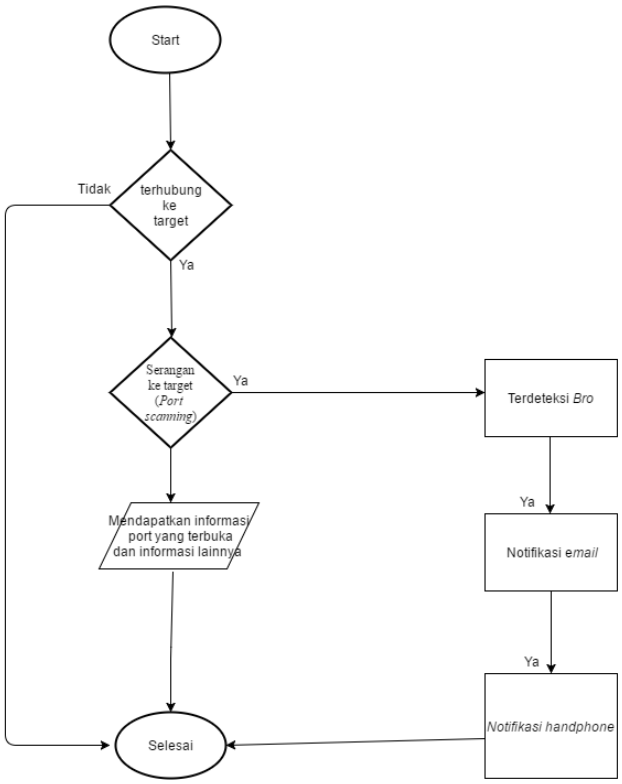
Keterangan :

1. Penyerang mencoba terkoneksi dengan *server*, dengan melakukan *ping* terhadap *server* apakah penyerang terkoneksi atau tidak dengan *server*
2. Setelah itu penyerang akan melakukan serangan *DOS* menggunakan aplikasi *hping3* kepada *server*
3. Setelah itu *server* akan menerima banyak *packet request* dan peningkatan penggunaan jumlah *memory* pada *server*.

3.6.2 Berikut adalah Scenario Penyerangan Menggunakan Bro

Berikut adalah skenario pengujian menggunakan *Bro* :

1. *Port Scanning*



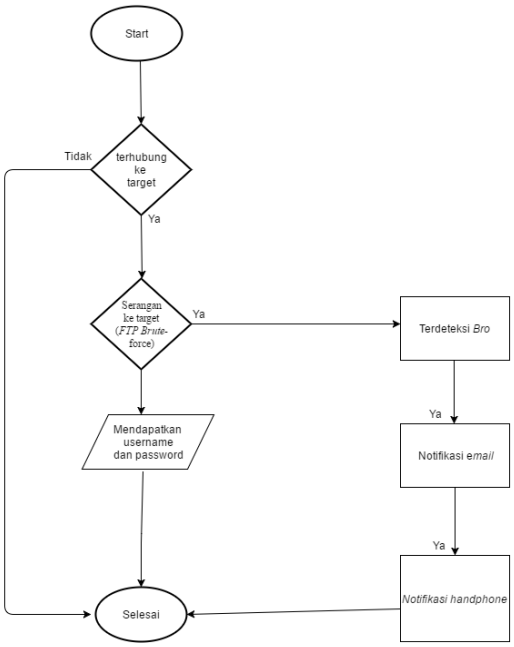
Gambar 3- 6 Flowchart serangan port scanning menggunakan Bro

Keterangan :

1. Penyerang mencoba melakukan koneksi kepada *server*, dengan cara mengirim *ping* terhadap *server*
2. Setelah itu penyerang melakukan *port scanning* dengan menggunakan *nmap*
3. Penyerang mendapatkan informasi mengenai *port* yang terbuka dan informasi lainnya
4. Serangan yang dilakukan penyerang terdeteksi oleh *bro* dan dicatat di *log*

5. Kemudian bro akan mengirim notifikasi *email* ke *admin*
6. Setelah notifikasi *email* terkirim maka aplikasi *IFTTT* akan mengirim notifikasi ke *handphone admin*

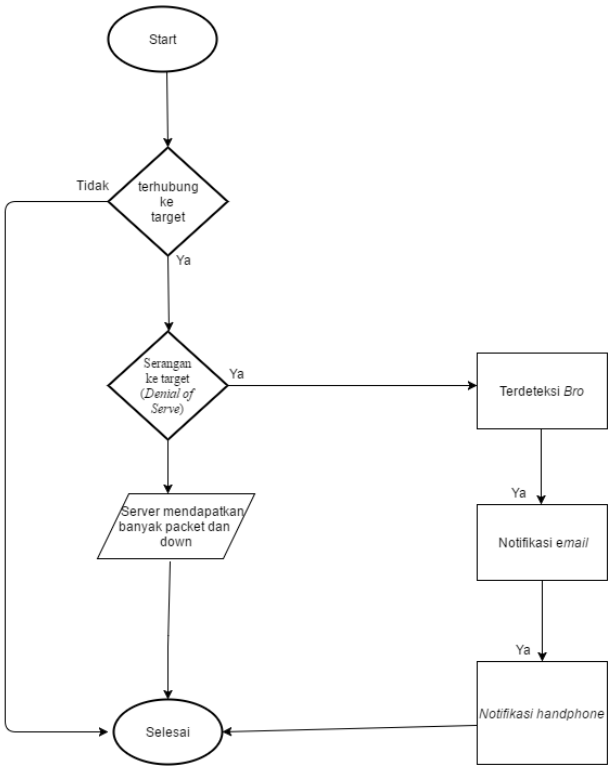
2. *FTP Brute-force*



Gambar 3- 7 Flowchart serangan *FTP Brute-force* menggunakan Bro

1. Penyerang mencoba melakukan koneksi kepada *server*, dengan cara mengirim *ping* terhadap *server*
2. Setelah itu penyerang melakukan *FTP Brute-force* kepada *server*
3. Kemudian penyerang mendapatkan *username* dan *password* yang digunakan untuk mengakses *FTP server*
4. Serangan yang dilakukan penyerang terdeteksi oleh *Bro* dan tercatat di *log*
5. Kemudian *Bro* akan mengirim notifikasi ke *email admin*
6. Setelah notifikasi *email* terkirim maka aplikasi *IFTTT* akan mengirim notifikasi ke *handphone admin*.

3. *DOS (Denial of Service)*



Gambar 3- 8 Flowchart serangan Denial of Service (DOS) menggunakan Bro

Keterangan :

1. Penyerang mencoba melakukan koneksi kepada *server*, dengan cara mengirim *ping* terhadap *server*
2. Setelah itu penyerang melakukan serangan *DOS* (*Denial of Service*) menggunakan aplikasi *hping3* kepada *server*
3. *Server* akan menerima banyak *packet* yang terkirim
4. Serangan yang dilakukan penyerang terdeteksi oleh *Bro* dan tercatat di *log*
5. Kemudian *Bro* akan mengirim notifikasi *email* ke *admin*
6. Setelah notifikasi *email* terkirim maka aplikasi *IFTTT* akan mengirim notifikasi ke *handphone admin*.

BAB 4
IMPLEMENTASI DAN PENGUJIAN

4.1 Implementasi

Pada tahap ini dijelaskan implementasi – implementasi apa saja yang dilakukan pada sisi *server*, *user*, *admin* dan penyerang.

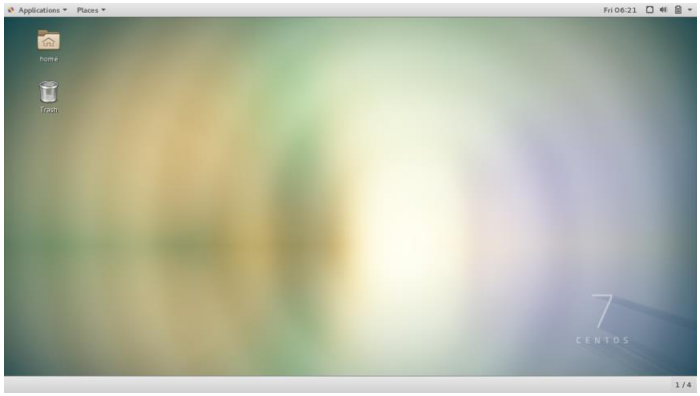
4.1.1 Implementasi dari Sisi *Server*

Pada tahap ini akan dijelaskan langkah – langkah instalasi *CentOS* pada *server*, Instalasi *vsftpd* dan konfigurasi *sendmail*, instalasi dan konfigurasi *Bro-ids* serta pembuatan *recipe* untuk notifikasi *client* via *handphone* berikut langkah – langkah tersebut :

4.1.1.1 Instalasi Linux CentOS

1. Sambungkan *USB Installer* yang berisi *ISO Linux CentOS 7 64 bit* ke laptop yang di-*install*, di sini penulis menggunakan aplikasi *Win32Diskimager-0.9.5*. Kemudian *restart* laptop dan konfigurasi pada *BIOS* untuk membaca *USB Installer* saat *booting*
2. Pada tampilan menu *boot CentOS 7* pilih *Install CentOS 7*, lalu tekan *enter*
3. Setelah itu pilih bahasa yang akan digunakan, lalu klik *continue*
4. Kemudian klik *INSTALLATION DESTINATION*
5. Lalu pilih *harddisk* yang akan digunakan, lalu klik *done*
6. Kemudian klik *SOFTWARE SELECTION*
7. Lalu pilih *Server with GUI* dan centang *service* yang dibutuhkan lalu klik *done*
8. Kemudian klik *Begin Installation*
9. Setelah itu klik *ROOT PASSWORD* untuk membuat *password* untuk *root*

10. Kemudian masukkan *password* di kolom *Root Password* dan di kolom *Confirm*, lalu klik *Done*
11. Setelah itu klik *USER CREATION* untuk membuat *user* baru
12. Setelah itu isikan pada kolom *Full name* dan kolom *User name* untuk *user name* yang akan digunakan oleh pengguna yang baru. Kemudian isikan kolom *Password* dan Confirm *password* untuk *password* yang akan digunakan oleh *user* baru
13. Setelah Instalasi selesai klik *Reboot*
14. Setelah itu tekan angka 1 kemudian tekan huruf c lalu tekan *enter*
15. Setelah itu tekan angka 2 kemudian tekan *enter*
16. Setelah itu tekan huruf c kemudian tekan *enter*
17. Setelah itu maka *Linux CentOS 7 64 bit* telah selesai ter-*install*



Gambar 4- 1 Tampilan setelah login di CentOS 7

Untuk lebih lengkap gambar ada di lampiran 1

4.1.1.2 Instalasi FTP pada Server

Berikut merupakan langkah – langkah instalasi service *FTP* pada *Server*.

1. Meng-*Install packet vsftpd* dengan perintah *yum install vsftpd*

```
[root@localhost ~]# yum -y install vsftpd
```

Gambar 4- 2 Instalasi packet vs*ftp*d

2. Setelah itu konfigurasi *file* pada *nano /etc/vsftpd/vsftpd.conf*

```
[root@localhost ~]# nano /etc/vsftpd/vsftpd.conf
```

Gambar 4- 3 *File* konfigurasi vs*ftp*d.conf

3. Lakukan konfigurasi seperti gambar 4-4

```
  GNU nano 2.3.1       File: /etc/vsftpd/vsftpd.conf

# Example config file /etc/vsftpd/vsftpd.conf
#
# The default compiled in settings are fairly paranoid. This sample file
# loosens things up a bit, to make the ftp daemon more usable.
# Please see vsftpd.conf.5 for all compiled in defaults.
#
# READ THIS: This example file is NOT an exhaustive list of vsftpd option$
# Please read the vsftpd.conf.5 manual page to get a full idea of vsftpd's
# capabilities.
#
# Allow anonymous FTP? (Beware - allowed by default if you comment this o$
anonymous_enable=NO
#
# Uncomment this to allow local users to log in.
# When SELinux is enforcing check for SE bool ftp_home_dir
local_enable=YES
#
# Uncomment this to enable any form of FTP write command.
write_enable=YES
#
```

Gambar 4- 4 Konfigurasi vs*ftp*d.conf

Keterangan :

Anonymous_enable=NO (tidak memberikan akses terhadap anonymous, atau pihak luar yang tidak terdaftar pada *server*)

Local_enable=YES (memberikan hak akses local *direktori* terhadap user)

Write_enable=YES (memberikan hak akses user untuk dapat meng-*edit* atau membaca *file*).

4. Setelah itu jalankan *file vsftpd* dengan perintah *systemctl start vsftpd* dan *systemctl enable vsftpd*

```
[root@localhost ~]# systemctl start vsftpd
[root@localhost ~]# systemctl enable vsftpd
Created symlink from /etc/systemd/system/multi-user.target.wants/vsftpd.service
to /usr/lib/systemd/system/vsftpd.service.
```

Gambar 4- 5 Perintah mengaktifkan vs*ftp*d

5. Buatlah *user* untuk mengakses *FTP* tersebut seperti gambar 4-6.

```
[root@DESKTOP-C3E1G4P ~]# adduser user1
[root@DESKTOP-C3E1G4P ~]# passwd user1
Changing password for user user1.
New password:
BAD PASSWORD: The password is shorter than 7 characters
Retype new password:
Sorry, passwords do not match.
New password:
Retype new password:
passwd: all authentication tokens updated successfully.
```

Gambar 4- 6 Membuat user untuk *FTP*

4.1.1.3 Instalasi Bro-IDS pada Server

Berikut langkah – langkah instalasi *Bro* pada *server Linux CentOS 7 64-bit* :

1. Meng-*install packet* – *packet* yang diperlukan agar aplikasi *Bro* dapat berjalan dengan baik pada *server Linux CentOS*
2. Kemudian *Install packet GeoIP-devel*
3. Setelah itu *download* aplikasi *GeoLiteCity*
4. Pindahkan *GeoLiteCity.dat* menuju *folder /usr/local/share/GeoIP/GeoIP.dat*
5. Kemudian *install* aplikasi *gawk* dengan perintah *yum install gawk*
6. Kemudian *install gperftools* dengan perintah *yum install gperftools*
7. Kemudian *download* aplikasi *ipsumdump*
8. Setelah itu masuk ke *folder ipsumdump* yang sudah di-*download* tadi lalu ketikkan perintah *./configure –prefix=/usr/*
9. Kemudian ketikkan perintah *make* di dalam *folder* aplikasi tersebut

10. Setelah itu ketikkan kembali perintah *make install* untuk mulai meng-*install* aplikasi tersebut

11. Setelah itu *download* aplikasi *Bro* pada *website* www.bro.org/release/bro-2.4.1.tar.gz.

12. Kemudian masuk ke *folder Bro* setelah itu masukkan perintah *./configure*

13. Kemudian masukkan perintah *make*

14. Setelah itu masukkan kembali perintah *make install*, maka aplikasi *Bro* akan ter-*install* pada *server Linux CentOS*.

Untuk lebih lengkapnya gambar ada pada lampiran 2

4.1.1.4 Konfigurasi *Rule* pada Bro

Langkah – langkah konfigurasi *rule* yang digunakan pada Bro :

1. Untuk *rule* yang ada pada Bro terdapat pada *direktori*nya sendiri, alamat *direktori*nya seperti pada gambar 4-7

```
[root@DESKTOP-C3E1G4P ~]# cd /usr/local/bro/share/bro/policy/protocols
[root@DESKTOP-C3E1G4P protocols]# ls
conn  dhcp  dns  ftp  http  modbus  mysql  rdp  smtp  ssh  ssl
```

Gambar 4- 7 Lokasi *rule* pada Bro

2. Untuk mengaktifkan *rule FTP Brute-force* masuk ke folder *FTP* dengan perinah *cd ftp* seperti gambar 4-8

```
[root@DESKTOP-C3E1G4P protocols]# cd ftp/
```

Gambar 4- 8 Masuk ke *direktori ftp*

3. Setelah pada folder *ftp* akan ada *rule* dengan nama *detect-bruteforcing.bro* seperti gambar 4-9

```
[root@DESKTOP-C3E1G4P ftp]# ls
detect.bro  detect-bruteforcing.bro  ftp_brute.bro  software.bro
```

Gambar 4- 9 Tampilan *direktori ftp*

4. Buka *file* detect-bruteforcing.bro menggunakan perintah *nano* seperti gambar 4-10

```
[root@DESKTOP-C3E1G4P ftp]# nano detect-bruteforcing.bro
```

Gambar 4- 10 Membuka *file* detect-bruteforcing.bro

5. Lalu tambahkan *script* seperti gambar di bawah pada *script detect-bruteforcing.bro* lalu *save*

```
hook Notice::policy(n: Notice::Info)
        {
                add n$actions[Notice::ACTION_EMAIL];
        }
```

Gambar 4- 11 Script untuk mengirim *email*

6. Setelah itu untuk mengaktifkannya *rule*-nya harus di tulis di *local.bro* yang berada pada *direktori cd /usr/local/bro/share/bro/site* seperti gambar 4-12

```
[root@DESKTOP-C3E1G4P ftp]# cd /usr/local/bro/share/bro/site
[root@DESKTOP-C3E1G4P site]# ls
conn-add-geodata.bro         intel-extend.bro  local.bro          local-proxy.bro   README    scan.bro          scan_udp.bro  smtp-url.bro
http-exe-bad-attributes.bro  ipblocker.bro     local-manager.bro  local-worker.bro  roam.bro  scan.cluster.bro  sidejack.bro  synflood.bro
```

Gambar 4- 12 Tampilan isi direktori site

7. Setelah masuk ke *direktori* tersebut, buka *file* local.bro menggunakan perintah *nano* seperti gambar 4-13

```
[root@DESKTOP-C3E1G4P site]# nano local.bro
```

Gambar 4- 13 Membuka file local.bro

8. Lalu ketikkan *@load* alamat *rule* yang akan dimasukkan, di sini alamat *rule* yang kita gunakan yaitu *policy/protokols/ftp/detect-bruteforcing*

```
#########FTP Brute-Force###########
@load policy/protocols/ftp/detect-bruteforcing
################+++++#########################
```

Gambar 4- 14 Mengaktifkan detect-bruteforcing

9. Setelah itu kita akan memasukkan *rule* scan.*bro* yang ada pada *folder site* sebelumnya. Buka *file* scan.*bro* menggunakan perintah *nano* seperti gambar 4-15

```
[root@DESKTOP-C3E1G4P site]# nano scan.bro
```

Gambar 4- 15 Membuka file scan.bro

10. Setelah itu masukkan kembali *script* pada *scan.bro* seperti gambar4-16, lalu *save*

```
hook Notice::policy(n: Notice::Info)
        {
                add n$actions[Notice::ACTION_EMAIL];
        }
```

Gambar 4- 16 Script untuk mengirim *email*

11. Kemudian buka kembali *local.bro*

```
[root@DESKTOP-C3E1G4P site]# nano local.bro
```

Gambar 4- 17 Membuka file local.bro

12. Setelah itu masukkan *script* yang akan dijalankan seperti gambar 4-18, lalu *save*

```
##########SCAN#############
@load scan
########+++############
```

Gambar 4- 18 Mengaktifkan file scan.bro

13. Setiap kita memasukkan *script* baru pada *Bro*, agar dapat terbaca saat *bro* dijalankan, maka kita harus menjalankan ulang aplikasi *Bro* tersebut.

4.1.1.5 Konfigurasi *Sendmail* pada *Server*

Langkah – langkah instalasi dan konfigurasi *sendmail* agar terintegrasi dengan akun *gmail* :

1. Lakukan instalasi *packet – packet sendmail* dengan perintah *yum install sendmail mailutils sendmail-bin*
2. Setelah itu buat *folder authinfo* sebagai tempat autentikasi *email* yang akan digunakan
3. Setelah itu masuk ke *folder* yang telah dibuat tadi
4. Kemudian buat sebuah *file* dengan nama *gmail-auth*
5. Pada *file* yang telah dibuat tadi isikan alamat *email* serta *password* yang digunakan setelah itu simpan *file* tersebut
6. Setelah itu buat *database gmail-auth* dengan perintah *makemap hash*

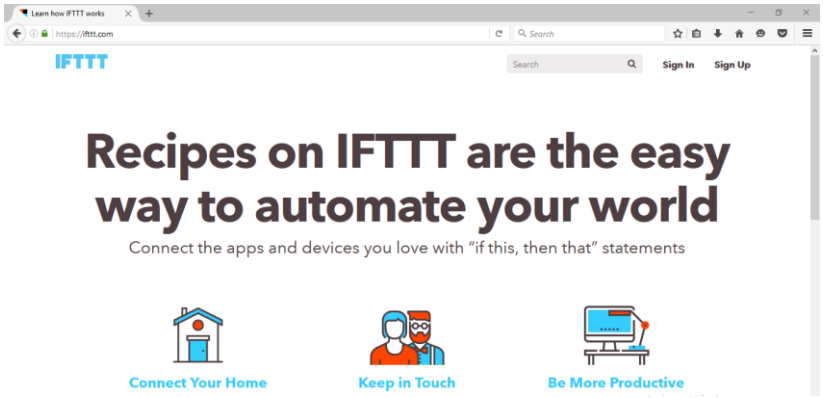
7. Kemudian *edit file sendmail.mc*
8. Lalu isikan seperti gambar dibawah
9. Setelah itu masukkan perintah *make –C*
10. Kemudian *restart sendmail.*

Untuk lebih lanjut gambar ada di lampiran 3

4.1.1.6 Cara Registrasi di *IFTTT*

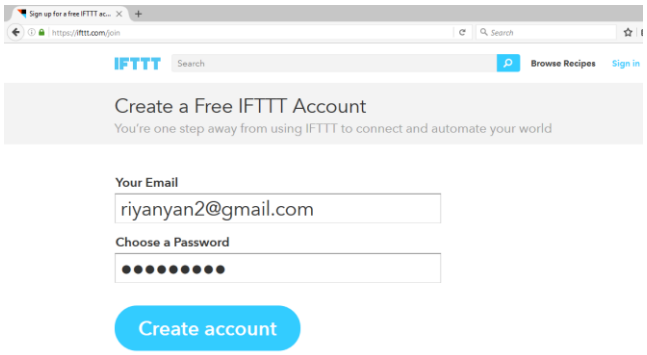
Berikut langkah – langkah registrasi di *web IFTTT*

1. Buka *web IFTTT* di alamat *www.IFTTT.com*, setelah itu klik pada *Sign Up* pada pojok atas sebelah kanan



Gambar 4.7 Tampilan form registrasi website *IFTTT*

2. Setelah itu isikan alamat *email* yang akan digunakan berserta *password* yang akan digunakan. Kemudian klik *Create account*.



Gambar 4- 19 Tampilan form registrasi *website IFTTT*

4.1.1.7 Cara Membuat Recipe pada *IFTTT*

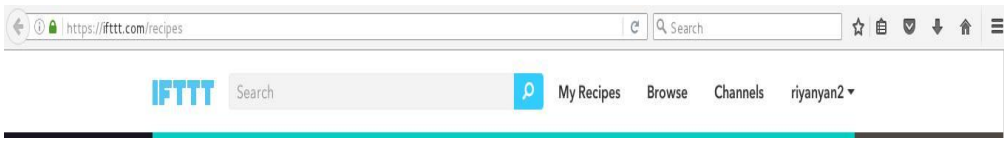
Langkah – langkah membaut *recipe* pada *IFTTT*

1. Klik nama *user* yang telah digunakan lalu pilih *create*
2. Setelah itu klik tulisan *this* yang digaris bawahi berwarna biru
3. Pada kolom *Choose Trigger Channel* ketikkan *Gmail*, lalu klik *Gmail* tersebut
4. Setelah itu klik *New email in inbox from*
5. Kemudian isikan alamat *email* yang akan dijadikan sebagai acuan untuk mengirim notifikasi, setelah itu klik *Create Trigger*
6. Kemudian klik tulisan *that* dengan garis bawah dan berwarna biru
7. Setelah itu pada kolom *choose Action Channel* ketikkan *notif*, lalu pilih *IF Notificaions*
8. Kemudian klik *send a notification*
9. Setelah itu isikan pesan dengan menambahkan *{{FromAddress}}, {{Subject}} dan {{BodyPlain}}*, kemudian klik *create action*
10. Kemudian klik *Create Recipe*
11. Setelah itu untuk mengaktifkannya, ubah *Turn off* menjadi *Turn on* dengan mengkliknya.

Untuk lebih lanjut gambar ada pada lampiran 4

4.1.1.8 Membuat Recipe Notifikasi untuk Client

1. Login pada *website IFTTT*, setelah itu klik *menu my recipes*

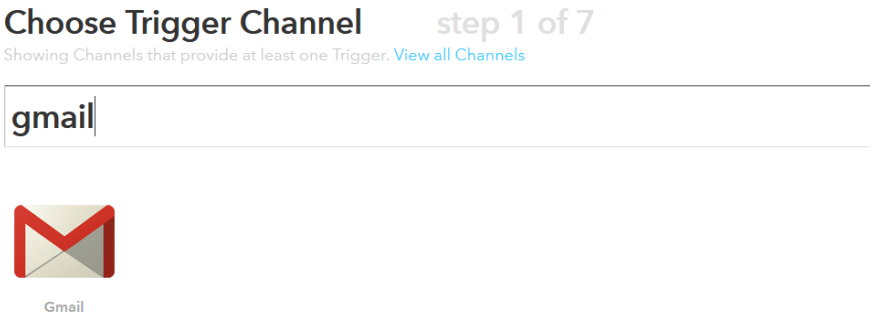


Gambar 4- 20 Tampilan Login pada website IFTTT

2. Setelah itu klik tulisan *this* yang berwarna biru

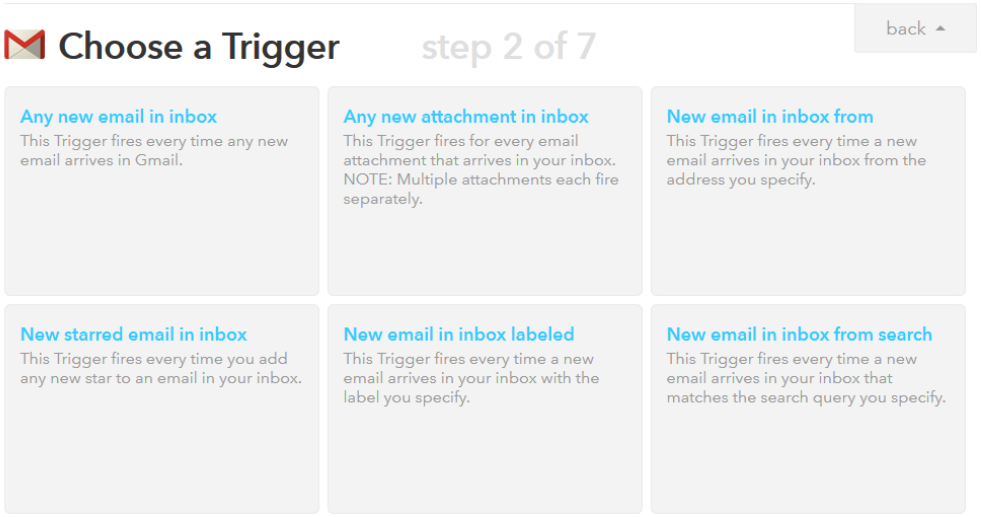
Gambar 4- 21 Tulisan this yang bergaris bawah berwarna biru

3. Setelah itu pada *Choose Trigger Channel*, ketikkan *gmail*, lalu klik gmail tersebut



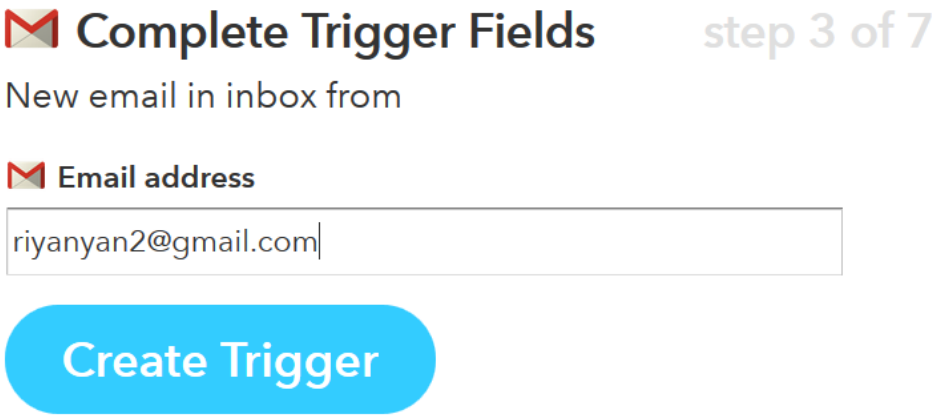
Gambar 4- 22 Aplikasi Gmail yang akan jadi pemicu sms

4. Pada *Choose a Trigger,* klik *New email in inbox from* seperti gambar 4-23



Gambar 4- 23 New email in inbox from

5. Pada *Email Address* tuliskan alamat *email* yang akan jadi pemicunya

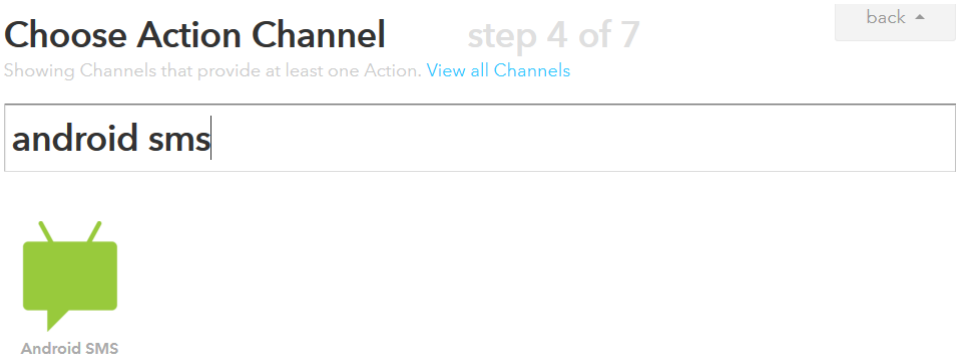


Gambar 4- 24 Membuat Trigger

6. Kemudian klik tulisan *that*

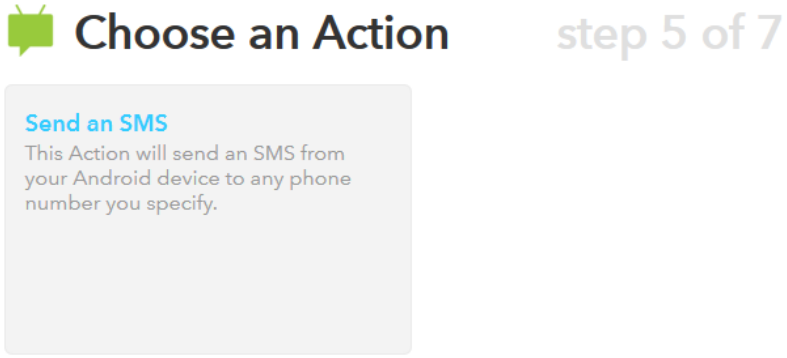
Gambar 4- 25 Tulisan that yang bergari bawah dan berwarna biru

7. Pada *Choose Action Channel* ketikkan *Android SMS*, untuk mengirim *sms* jika ada *email* dari *email* yang dituliskan tadi, seperti gambar 4-26



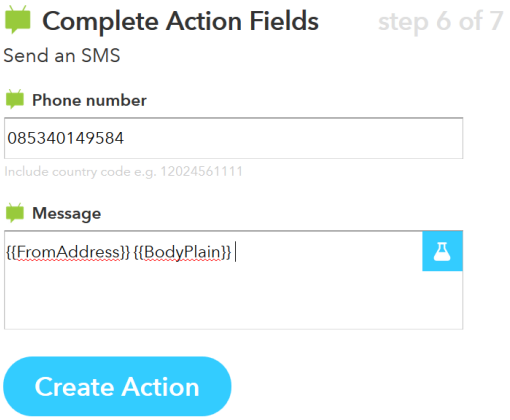
Gambar 4- 26 Android SMS untuk mengirim pesan

8. Pada *Choose an Action,* pilih *Send an SMS*



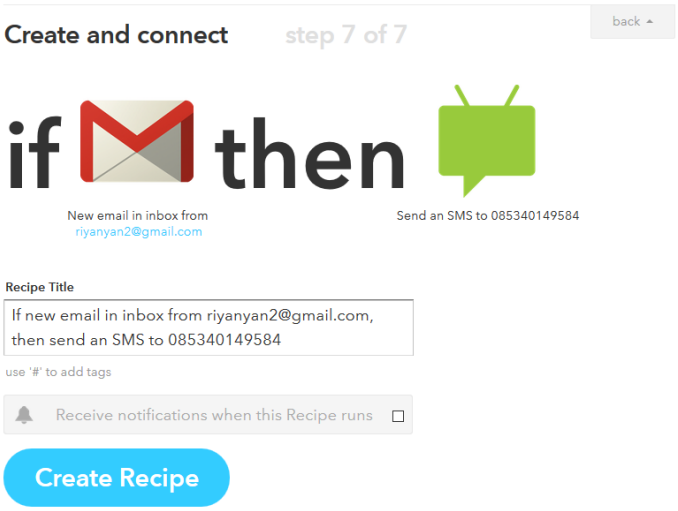
Gambar 4- 27 Send an SMS

9. Kemudian masukkan nomor telepon yang akan dikirimi pesan, dan juga pesan yang akan ditampilkan pada kolom *Messages,* setelah itu klik *Create Action*



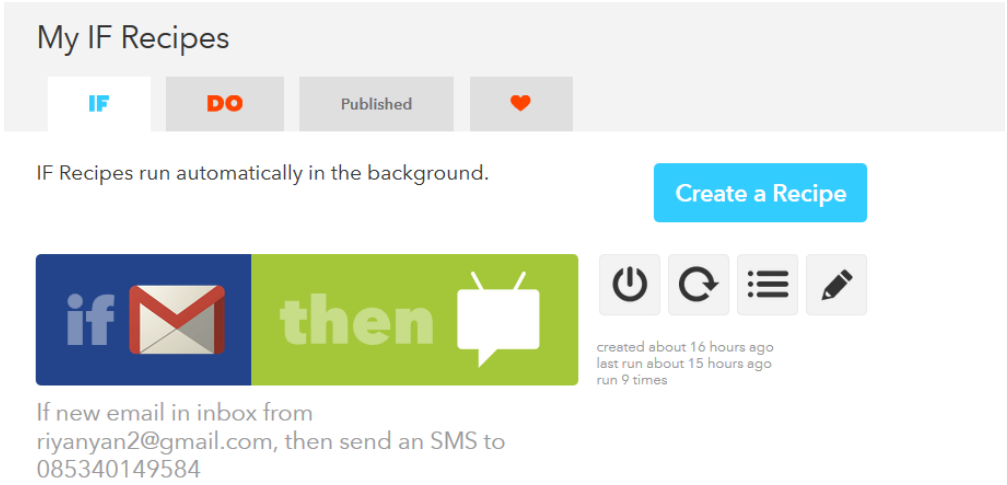
Gambar 4- 28 Masukkan nomor dan tampilan pesan yang ingin dikirim

10. Setelah itu klik *Create Recipe* seperti gambar 4-29



Gambar 4- 29 Create Recipe

11. Tampilan Recipe yang telah berhasil dibuat



Gambar 4- 30 Recipe yang telah berhasil dibuat

4.2 Implementasi dari Sisi *Admin*

Pada tahap ini akan dilakukan langkah – langkah cara meng-*install* aplikasi *IFTTT* dan konfigurasi *IFTTT* pada *handphone admin*, berikut langkah – langkah tersebut :

4.2.1.1 Konfigurasi *IFTTT* pada *Handphone Admin*

Berikut langkah – langkah konfigurasi dan instalasi *IFTTT* pada *handphone*

1. Buka *playstore* dan ketikkan di kotak pencarian *IFTTT*

2. Setelah itu klik PASANG untuk melakukan instalasi pada *handphone*

3. Kemudian klik TERIMA untuk melanjutkan instalasi
4. Setelah itu selesai ter-*install* klik BUKA, untuk menjalankan aplikasi tersebut
5. Kemudian masukkan *email/username* dan *password* yang sudah kita daftarkan sebelumnya pada *web IFTTT*
6. Setelah itu klik *icon* pada pojok kanan atas
7. Lalu pilih *recipe* yang akan digunakan di sini untuk mengaktifkan *recipe* yang telah dibat tinggal mengklik *recipe* tersebut.

Untuk lebih lengkapnya gambar bisa di lihat pada lampiran 5

Untuk implementasi pada sisi pengguna dan pada sisi penyerang tidak dilakukan karena untuk sisi pengguna hanya menerima layanan dari *server* dan untuk sisi penyerang hanya melakukan serangan terhadap *server* tanpa ada konfigurasi.

4.3 Pengujian

Pada tahap ini dilakukan pengujian yaitu pengujian tanpa menggunakan *Bro* dan pengujian menggunakan *Bro*, serta untuk menguji apakah konfigurasi sudah berjalan dengan baik atau tidak akan dilalukan pengujian, dari sisi *server*, pengguna, *admin*, dan penyerang. Pengujiannya adalah sebagai berikut :

a. Pengujian dari sisi *Server*
    1. Pengujian koneksi antara *server* dengan *client*. *Server* dengan penyerang, dan *client* dengan penyerang
    2. Pengujian aplikasi *Bro* apakah sudah berjalan dengan baik pada *server*
    3. Pengujian *sendmail* pada *server*
    4. Pengujian *FTP server* pada *server* yang telah dibuat

b. Pengujian dari sisi User

1. Pengujian *user* untuk mengakses *FTP* meng-*upload* dan men-*download file*

c. Pengujian dari sisi Penyerang

1. Pengujian dengan serangan *port scanning* untuk mengetahui *port* yang terbuka pada *server*, saat tidak menggunakan *Bro* dan saat menggunakan *Bro*
2. Pengujian dengan serangan *FTP Brute-force* untuk mengetahui *username* dan *password*, saat tidak menggunakan *Bro* dan saat menggunakan *Bro*
3. Pengujian dengan serangan *DOS (Denial of Service)* untuk melumpuhkan *server*, pada saat menggunakan *Bro* dan saat tidak menggunakan *Bro*.

Pengujian dari sisi *admin* tidak dilakukan karena saat pengujian *sendmail* dari sisi *server* yaitu dengan mencoba mengirim *email* ke alamat *admin*, maka aplikasi *IFTTT* akan memberi notifikasi ke *handphone admin* saat *email* tersebut telah sampai ke tujuan.

4.3.1 Pengujian dari Sisi *Server*

Pada tahap ini akan dilakukan dilakukan pengujian aplikasi dan konfigurasi yang sudah ter-*install* sebelumnya pada *server*, pengujiannya adalah sebagai berikut :

4.3.1.1 Pengujian Konektivitas pada Server dengan Client dan Penyerang

Pada tahap ini akan dilakukan pengujian koneksi antara *server* dengan *client*, *server* dengan penyerang dan penyerang dengan *client*. Tujuannya adalah untuk mengetahui apakah antara *server*, *client* dan penyerang dapat terhubung dengan baik. Pengujian dialkukan dengan mengirim paket *PING* kepada masing – masing *node*. Berikut hasil pengujiannya dapat dilihat pada tabel berikut :

1. *Server*

Hasil pengujian konektivitas antara *server* dengan *user* dan *server* dengan penyerang.

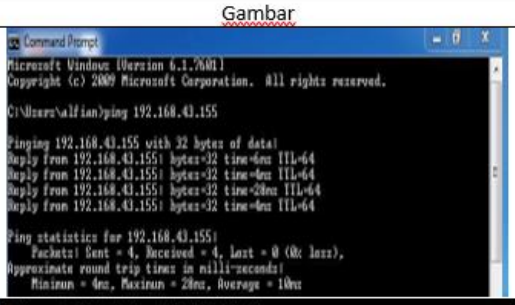
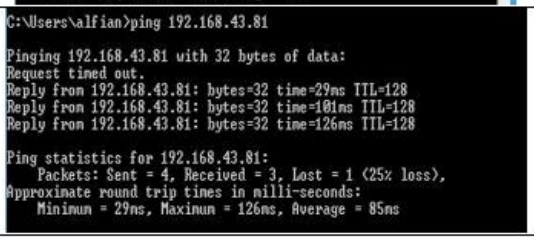
Tabel 4- 1 Konektivitas server

NO	Perangkat	Gambar	Hasil
1.	User	[root@DESKTOP-C3E1G4P bin]# ping 192.168.43.178 PING 192.168.43.178 (192.168.43.178) 56(84) bytes of data. 64 bytes from 192.168.43.178: icmp_seq=1 ttl=128 time=10.4 ms 64 bytes from 192.168.43.178: icmp_seq=2 ttl=128 time=235 ms 64 bytes from 192.168.43.178: icmp_seq=3 ttl=128 time=54.9 ms 64 bytes from 192.168.43.178: icmp_seq=4 ttl=128 time=77.4 ms 64 bytes from 192.168.43.178: icmp_seq=5 ttl=128 time=101 ms 64 bytes from 192.168.43.178: icmp_seq=6 ttl=128 time=1.99 ms 64 bytes from 192.168.43.178: icmp_seq=7 ttl=128 time=44.1 ms 64 bytes from 192.168.43.178: icmp_seq=8 ttl=128 time=67.5 ms	Terhubung
2.	Penyerang	[root@DESKTOP-C3E1G4P bin]# ping 192.168.43.81 PING 192.168.43.81 (192.168.43.81) 56(84) bytes of data. 64 bytes from 192.168.43.81: icmp_seq=1 ttl=128 time=92.7 ms 64 bytes from 192.168.43.81: icmp_seq=2 ttl=128 time=115 ms 64 bytes from 192.168.43.81: icmp_seq=3 ttl=128 time=2.28 ms 64 bytes from 192.168.43.81: icmp_seq=4 ttl=128 time=162 ms 64 bytes from 192.168.43.81: icmp_seq=5 ttl=128 time=185 ms 64 bytes from 192.168.43.81: icmp_seq=6 ttl=128 time=1.94 ms 64 bytes from 192.168.43.81: icmp_seq=7 ttl=128 time=27.6 ms 64 bytes from 192.168.43.81: icmp_seq=8 ttl=128 time=51.1 ms 64 bytes from 192.168.43.81: icmp_seq=9 ttl=128 time=73.0 ms	Terhubung

2. *Client*

Hasil pengujian konektivitas antara *client* dengan *server*, *client* dengan penyerang.

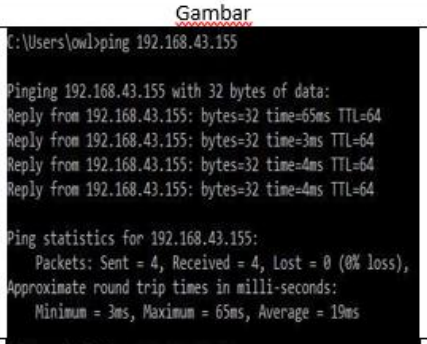
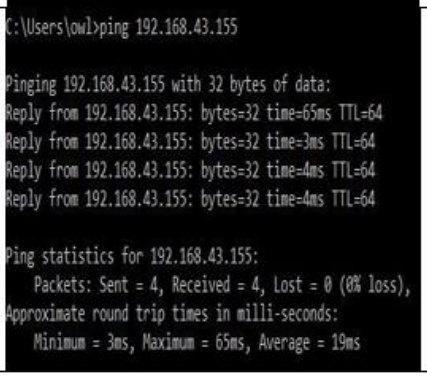
Tabel 4- 2 Konektivitas user

NO	Perangkat	Gambar	Hasil
1.	Server		Terhubung
2.	Penyerang		Terhubung

3. Penyerang

Hasil pengujian konektivitas antara penyerang dengan *server*, dan penyerang dengan *user.*

Tabel 4- 3 Konektivitas penyerang

NO	Perangkat	Gambar	Hasil
1.	Server		Terhubung
2.	User		Terhubung

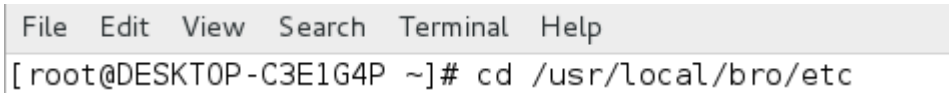
4.3.1.2 Pengujian Bro pada Linux CentOS

Pada tahap ini akan dilakukan pengujian aplikasi *Bro* yang telah di-*install* sebelumnya pada *server*.

1. Buka *terminal* pada *linux CentOS*

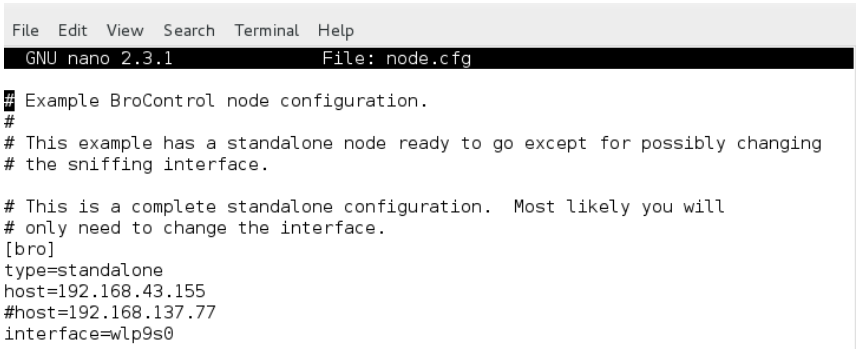
Gambar 4- 31 Terminal pada CentOS

2. Kemudian ketikkan pada *terminal cd /usr/local/bro/etc* untuk masuk ke *folder* konfigurasi *bro*.



Gambar 4- 32 Perintah untuk ke folder Bro

3. Setelah itu ketikkan *nano node.cfg*, fungsinya adalah untuk mengkonfigurasikan *IP address Bro* tersebut. Masukkan *IP address server* ke dalam *node.cfg* seperti pada gambar4-22.



Gambar 4- 33 Memberi IP address server pada Bro

4. Setelah itu ketikkan *nano broctl.cfg* lalu masukkan *email* yang akan dijadikan acuan saat terjadi serangan, seperti gambar 4-23.

```
  GNU nano 2.3.1                File: broctl.cfg

## Global BroControl configuration file.

###############################################
# Mail Options

# Recipient address for all emails sent out by Bro and BroControl.
MailTo = riyanyan2@gmail.com
```

Gambar 4- 34 Menulis *email* admin pada bro

5. Setelah itu ketikkan *cd /usr/local/bro/bin* untuk masuk ke *folder Bro*, kemudian ketikkan *./broctl* untuk menjalankan aplikasi *Bro*.

```
File  Edit  View  Search  Terminal  Help
[root@DESKTOP-C3E1G4P ~]# cd /usr/local/bro/bin
[root@DESKTOP-C3E1G4P bin]# ls
bro               broctl   bro_ipblocker_block  trace-summary
broccoli-config   bro-cut  capstats
[root@DESKTOP-C3E1G4P bin]# ./broctl

Welcome to BroControl 1.4

Type "help" for help.

[BroControl] >
```

Gambar 4- 35 Tampilan saat menjalankan bro

6. Kemudian ketikkan *deploy* pada tampilan *broctl* tersebut. Pada saat ini *Bro* akan mengecek apakah konfigurasi yang kita lakukan telah berhasil ataukah ada kesalahan

```
[BroControl] > deploy
checking configurations ...
installing ...
removing old policies in /usr/local/bro/spool/installed-scripts-do-not-touch/site ...
removing old policies in /usr/local/bro/spool/installed-scripts-do-not-touch/auto ...
creating policy directories ...
installing site policies ...
generating standalone-layout.bro ...
generating local-networks.bro ...
generating broctl-config.bro ...
generating broctl-config.sh ...
updating nodes ...
stopping ...
stopping bro ...
starting ...
starting bro ...
[BroControl] >
```

Gambar 4- 36 Menjalankan perintah Menjalankan perintah *deploy*

7. Untuk melihat *Bro* berjalan atau tidak pada *server* ketikkan *status* atau *top* pada *broctl*.

```
[BroControl] > status
Getting process status ...
Getting peer status ...
Name         Type       Host          Status    Pid    Peers  Started
bro          standalone 192.168.43.155 running   11394  0      10 Aug 18:04:54
[BroControl] >
```

Gambar 4- 37 Menjalankan perintah status

```
Name         Type       Host          Pid     Proc    VSize  Rss  Cpu  Cmd
bro          standalone 192.168.43.155 11394   parent  195M    53M  13%  bro
bro          standalone 192.168.43.155 11396   child   122M    45M   0%  bro
```

Gambar 4- 38 Menjalankan perintah top

4.3.1.3 Pengujian Sendmail pada Server

Pada tahap ini akan dilakukan pengujian terhadap aplikasi *sendmail* apakah telah berjalan dengan baik atau tidak dengan cara mengirim *email* ke *email* penulis.

1. Buka *terminal* lalu ketikkan perintah seperti di bawah pada terminal

   Echo “test gmail yanyan” | mail –s “sendmail gmail relay” riyanyan2@gmail.com

2. Setelah itu buka *gmail* untuk melihat apakah pesan terkirim ke alamat yang dituju atau tidak.

Gambar 4- 39 Tampilan *email* yang masuk dari pengujian sendmail

4.3.1.4 Pengujian FTP Server

Pada tahap ini akan dilakukan pengujian terhadap *FTP* yang telah dipasang pada *server* sebelumnya, apakah *client* dapat terhubung ke *server* atau tidak.

1. Pertama buat *file* terlebih dahulu seperti gambar 4-29

```
[root@ns1 ~]# nano /home/yanyan2.txt
```

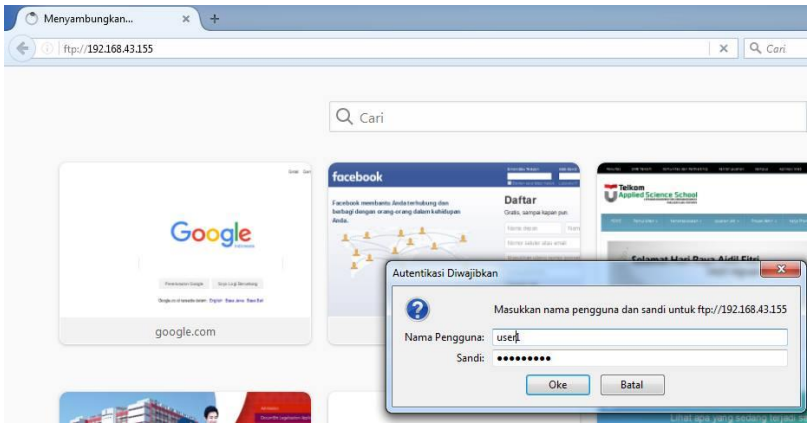
Gambar 4- 40 Membuat file untuk *FTP* server

2. Setelah itu cek apakah *file* telah berhasi dibuat atau tidak

```
[root@ns1 home]# ls
demo  riyan  test  user  user1  yanyan2.txt
```

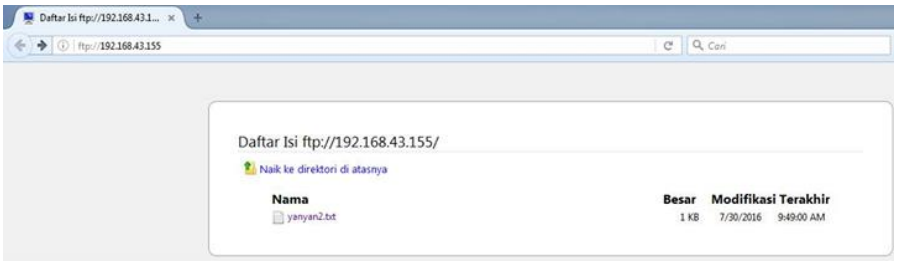
Gambar 4- 41 File yang telah dibuat

3. Kemudian pada *client* buka *browser*, lalu masukkan *ip server*, dan masukkan *username* dan *password* yang sudah dibuat sebelumnya di *server*



Gambar 4- 42 Browser untuk mencoba *FTP*

4. Setelah itu jika berhasil maka akan terlihat *file* yang sebelumnya sudah dibuat pada *server*.



Gambar 4- 43 File pada *ftp* server

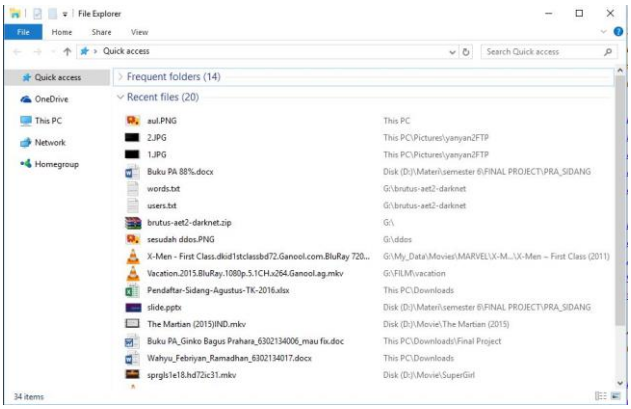
4.3.2 Pengujian dari Sisi User

Pada tahap ini akan dilakukan pengujian *FTP server* dari sisi *user* untuk meng-*upload* dan men-*download file* dari *server*.

4.3.2.1 Proses Meng-upload dan men-download *file*

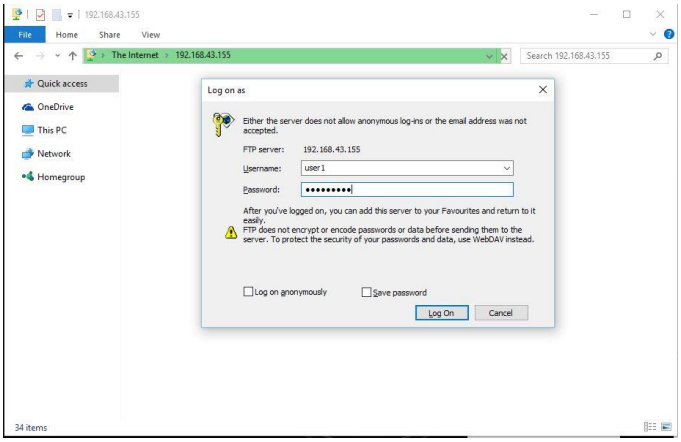
Pada tahap ini akan dilakukan pengujian *upload file* dan *download file* oleh *user*

1. Buka *file* explorer pada *windows*



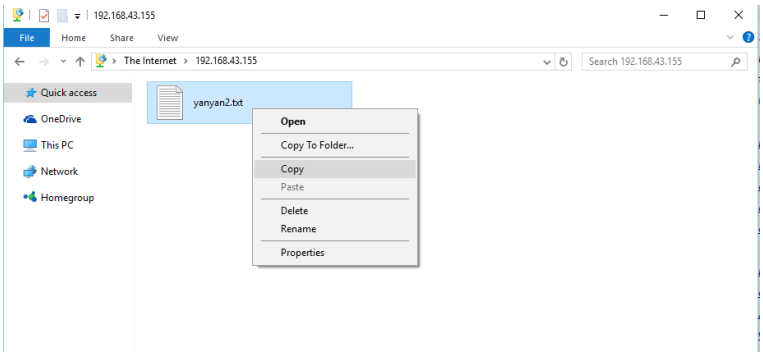
Gambar 4- 44 Tampilan file explorer pada windows

2. Setelah itu ketikkan *ftp://Alamat_IP_server*. Di sini *ftp://192.168.43.155*, lalu masukkan *username* dan *password* yang telah dibuat sebelumnya seperti gambar 4-34



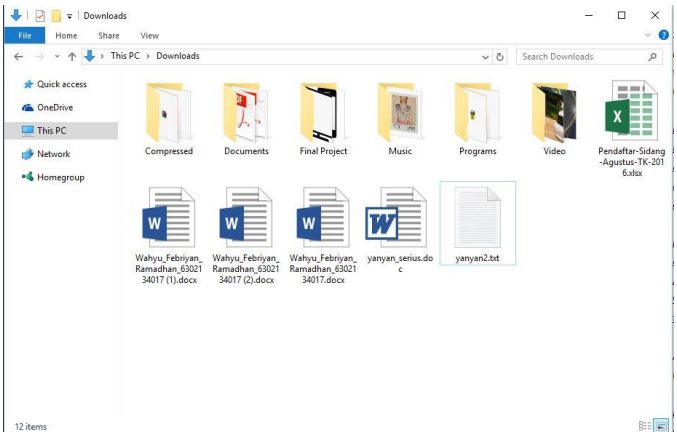
Gambar 4- 45 Memasukkan username dan password *FTP*

3. Setelah masuk ke *ftp server*, maka akan terlihat *file* yang telah dibuat sebelumnya yaitu yanyan2.txt, untuk mencoba mengambilnya digunakan perintah *copy* seperti gambar 4-35



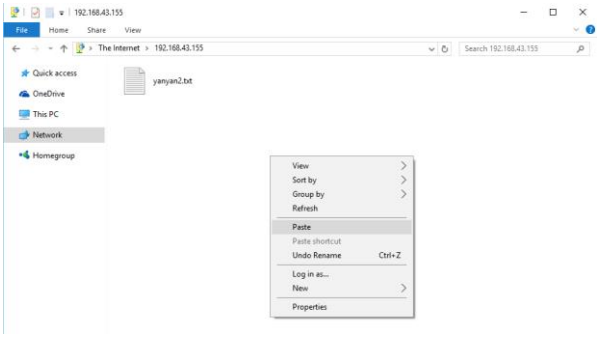
Gambar 4- 46 Mengambil file yanyan2.txt

4. Setelah itu kita *paste*-kan di tempat yang diinginkan



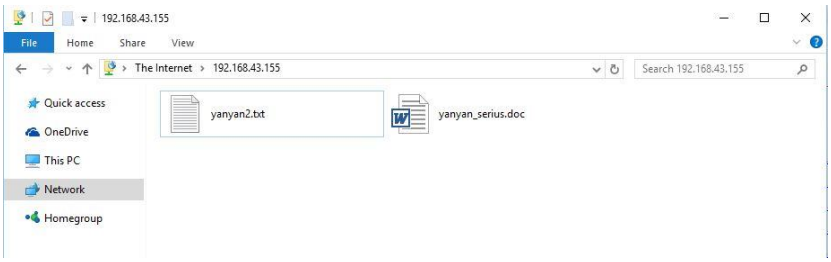
Gambar 4- 47 Menaruh file yanyan2.txt

5. Untuk proses meng-*upload file* ke dalam *server ftp*, copy *file* yang akan di upload ke *server ftp*, di sini penulis mencoba untuk meng-*upload file* yanyan_serius.doc setelah itu *paste*-kan pada *server ftp* seperti gambar 4-37



Gambar 4- 48 Proses menaruh file pada server *ftp*

6. *File* yanyan_serius.doc berhasil di upload pada *server ftp*, terlihat *file* tersebut berada pada *ftp server* seperti gambar 4-38.



Gambar 4- 49 File yanyan_serius.doc ditaruh di *ftp* server

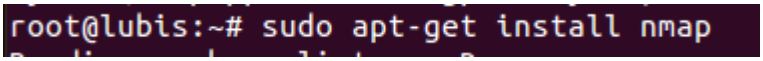
4.3.3 Pengujian dari Sisi Penyerang Tanpa menggunakan *Bro*

Pada tahap ini dilakukan pengujian serangan tanpa menggunakan *Bro* pada *server* Berikut pengujian dan hasil pengujiannya :

4.3.3.1 Pengujian Menggunakan Serangan Port Scanning

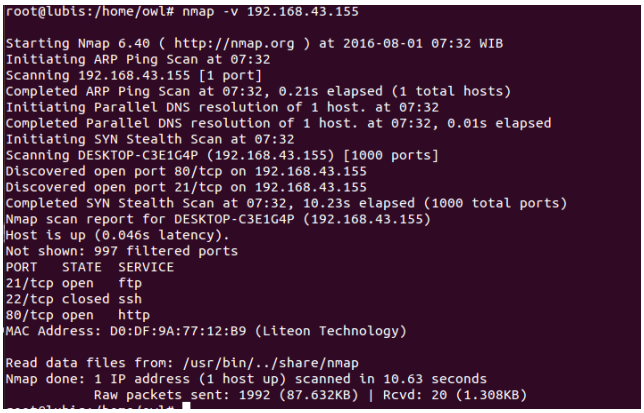
Pada tahap ini kita akan melakukan serangan *port scanning* terhadap *server* untuk melihat *port* yang terbuka atau mencari informasi lainnya.

1. Petama buka *terminal* pada *linux*, lalu install terlebih dahulu aplikasi *nmap* menggunakan perintah *sudo apt-get install nmap*



Gambar 4- 50 Proses installasi nmap

2. Setelah itu masukkan perintah *nmap –v IP-address* yang diserang, setelah itu maka *nmap* akan melakukan *scanning* terhadap target dan informasi akan muncul di layar *terminal*.



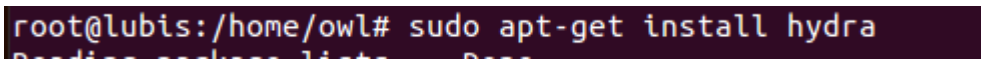
Gambar 4- 51 Informasi yang didapatkan dari port scanning

3. Penyerangan berhasil tanpa adanya notifikasi, hal ini sangat berbahaya karena akan dijadikan sebagai petunjuk untuk memulai serangan.

4.3.3.2 Pengujian Menggunakan Serangan FTP Brute-force

Pada tahap ini akan dilakukan pengujian menggunakan serangan *FTP Brute-force* dengan aplikasi *hydra*.

1. Pertama buka *terminal* dan instal aplikasi *hydra* menggunakan perintah *sudo apt-get install hydra*

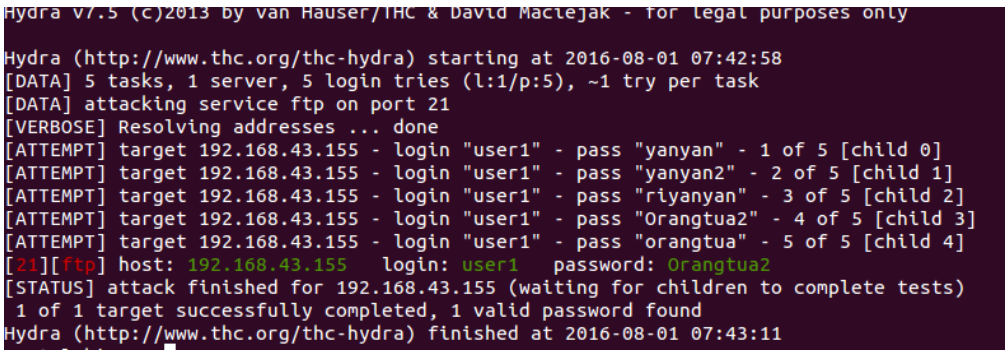


Gambar 4- 52 Proses Install hydra

2. Setelah itu masukkan perintah *hydra –l user1 –P /root/password.lst –vV ftp://192.168.43.155*

Gambar 4- 53 perintah serangan hydra

3. Setelah itu akan muncul *username* dan *password* target yang telah diserang



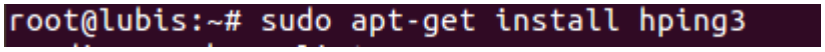
Gambar 4- 54 Hasil serangan hydra

4. Tidak ada notifikasi terhadap serangan ini dan sangat berbahaya karena dapat mengambil akun pengguna dan menggunakannya untuk hal – hal yang merugikan.

4.3.3.3 Pengujian Menggunakan Serangan DDOS

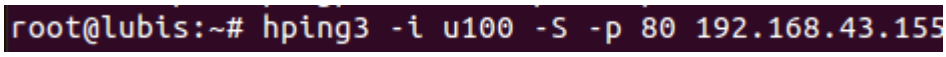
Pada tahap ini akan dilakukan pengujian menggunakan serangan *ddos* untuk melumpuhkan *server* menggunakan aplikasi *hping3*.

1. Buka *terminal* terlebih dahulu lalu instal aplikasi *hping3* dengan perintah *sudo apt-get install hping3*



Gambar 4- 55 Proses install hping3

2. Setelah itu ketikkan perintah *hping3 –i u100 –S 80 192.168.43.155*



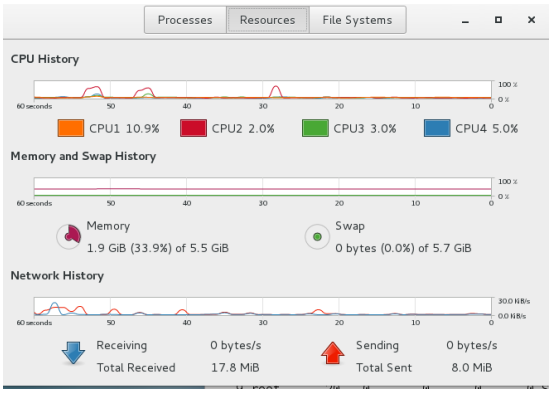
Gambar 4- 56 Perintah serangan hping3

3. *Hping3* akan melakukan serangan terhadap target, seperti gambar 4-46

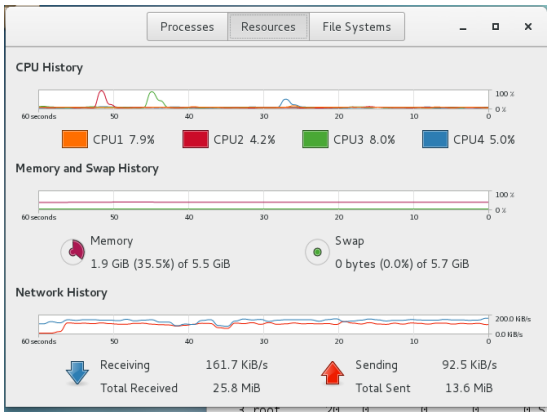
```
len=44 ip=192.168.43.155 ttl=64 DF id=0 sport=80 flags=SA seq=155283 win=29200 rtt=239.3 ms
len=44 ip=192.168.43.155 ttl=64 DF id=0 sport=80 flags=SA seq=155284 win=29200 rtt=239.2 ms
len=44 ip=192.168.43.155 ttl=64 DF id=0 sport=80 flags=SA seq=155285 win=29200 rtt=239.1 ms
len=44 ip=192.168.43.155 ttl=64 DF id=0 sport=80 flags=SA seq=155286 win=29200 rtt=239.0 ms
len=44 ip=192.168.43.155 ttl=64 DF id=0 sport=80 flags=SA seq=155287 win=29200 rtt=238.9 ms
len=44 ip=192.168.43.155 ttl=64 DF id=0 sport=80 flags=SA seq=155288 win=29200 rtt=238.8 ms
len=44 ip=192.168.43.155 ttl=64 DF id=0 sport=80 flags=SA seq=155289 win=29200 rtt=238.6 ms
len=44 ip=192.168.43.155 ttl=64 DF id=0 sport=80 flags=SA seq=155290 win=29200 rtt=238.5 ms
len=44 ip=192.168.43.155 ttl=64 DF id=0 sport=80 flags=SA seq=155291 win=29200 rtt=238.4 ms
len=44 ip=192.168.43.155 ttl=64 DF id=0 sport=80 flags=SA seq=155292 win=29200 rtt=238.3 ms
len=44 ip=192.168.43.155 ttl=64 DF id=0 sport=80 flags=SA seq=155293 win=29200 rtt=238.2 ms
len=44 ip=192.168.43.155 ttl=64 DF id=0 sport=80 flags=SA seq=155294 win=29200 rtt=238.1 ms
len=44 ip=192.168.43.155 ttl=64 DF id=0 sport=80 flags=SA seq=155295 win=29200 rtt=238.0 ms
len=44 ip=192.168.43.155 ttl=64 DF id=0 sport=80 flags=SA seq=155296 win=29200 rtt=237.9 ms
len=44 ip=192.168.43.155 ttl=64 DF id=0 sport=80 flags=SA seq=155297 win=29200 rtt=237.8 ms
len=44 ip=192.168.43.155 ttl=64 DF id=0 sport=80 flags=SA seq=155298 win=29200 rtt=237.7 ms
len=44 ip=192.168.43.155 ttl=64 DF id=0 sport=80 flags=SA seq=155299 win=29200 rtt=237.6 ms
len=44 ip=192.168.43.155 ttl=64 DF id=0 sport=80 flags=SA seq=155300 win=29200 rtt=237.5 ms
len=44 ip=192.168.43.155 ttl=64 DF id=0 sport=80 flags=SA seq=155301 win=29200 rtt=237.3 ms
len=44 ip=192.168.43.155 ttl=64 DF id=0 sport=80 flags=SA seq=155302 win=29200 rtt=237.2 ms
len=44 ip=192.168.43.155 ttl=64 DF id=0 sport=80 flags=SA seq=155303 win=29200 rtt=237.1 ms
len=44 ip=192.168.43.155 ttl=64 DF id=0 sport=80 flags=SA seq=155304 win=29200 rtt=237.0 ms
len=44 ip=192.168.43.155 ttl=64 DF id=0 sport=80 flags=SA seq=155305 win=29200 rtt=236.9 ms
len=44 ip=192.168.43.155 ttl=64 DF id=0 sport=80 flags=SA seq=155307 win=29200 rtt=236.7 ms
len=44 ip=192.168.43.155 ttl=64 DF id=0 sport=80 flags=SA seq=155308 win=29200 rtt=236.6 ms
len=44 ip=192.168.43.155 ttl=64 DF id=0 sport=80 flags=SA seq=155309 win=29200 rtt=236.5 ms
len=44 ip=192.168.43.155 ttl=64 DF id=0 sport=80 flags=SA seq=155310 win=29200 rtt=236.4 ms
len=44 ip=192.168.43.155 ttl=64 DF id=0 sport=80 flags=SA seq=155311 win=29200 rtt=236.3 ms
len=44 ip=192.168.43.155 ttl=64 DF id=0 sport=80 flags=SA seq=155312 win=29200 rtt=236.2 ms
len=44 ip=192.168.43.155 ttl=64 DF id=0 sport=80 flags=SA seq=155327 win=29200 rtt=235.5 ms
len=44 ip=192.168.43.155 ttl=64 DF id=0 sport=80 flags=SA seq=155328 win=29200 rtt=235.4 ms
len=44 ip=192.168.43.155 ttl=64 DF id=0 sport=80 flags=SA seq=155329 win=29200 rtt=235.3 ms
len=44 ip=192.168.43.155 ttl=64 DF id=0 sport=80 flags=SA seq=155330 win=29200 rtt=235.2 ms
len=44 ip=192.168.43.155 ttl=64 DF id=0 sport=80 flags=SA seq=155331 win=29200 rtt=238.9 ms
len=44 ip=192.168.43.155 ttl=64 DF id=0 sport=80 flags=SA seq=155332 win=29200 rtt=238.8 ms
```

Gambar 4- 57 Serangan hping3

4. Hasil penggunaan *resource server* akibat serangan ddos



Gambar 4- 58 Serangan sebelum hping3



Gambar 4- 59 Serangan sesudah hping3

5. Serangan berhasil tanpa adanya notifikasi, dan ini berbahaya karena akan berpengaruh terhadap performa *server*.

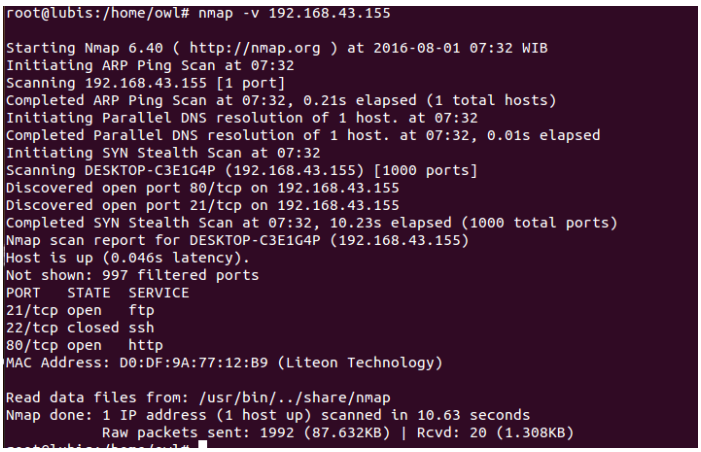
4.3.4 Pengujian dari Sisi Penyerang Menggunakan Bro

Pada tahap ini akan dilakukan serangan pada *server* yang sudah ter-*install Bro* di dalamnya, berikut pengujian dan hasil pengujiannya :

4.3.4.1 Pengujian Menggunakan Port Scanning

Pada tahap ini akan dilakukan penyerangan menggunakan metode *Port Scanning* terhadap *server* yang telah di-*Install Bro* sebelumnya, berikut pengujian dan hasil pengujiannya.

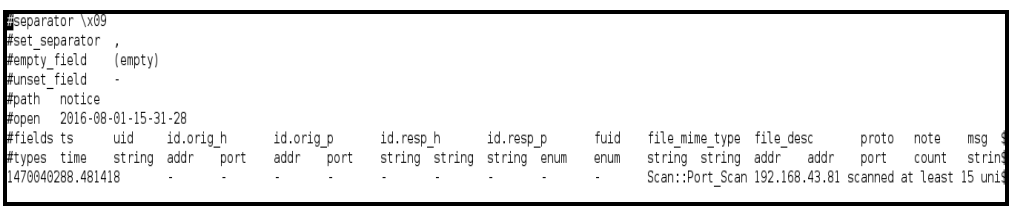
1. Buka *terminal*, lalu masukkan perintah *nmap –v IP* tujuan atau *web* yang dituju



Gambar 4- 60 Hasil serangan nmap

2. Setelah berhasil, maka akan muncul informasi mengenai target yang telah di-*scan* tersebut, seperti gambar 4-49

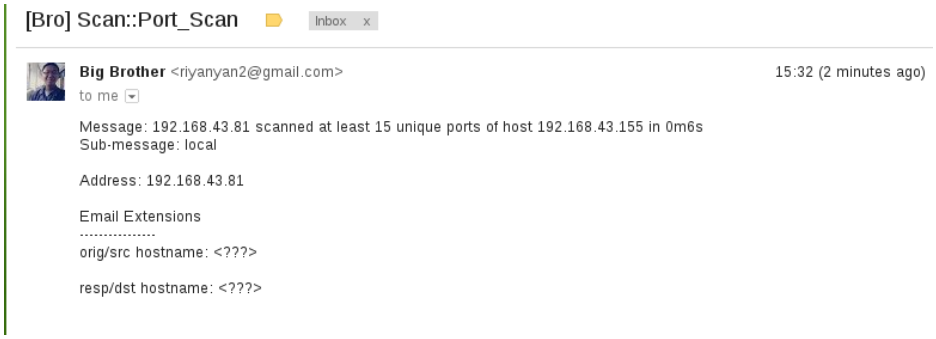
3. Tampilan log Bro saat terjadi serangan dan terdeteksi



Gambar 4- 61 Tampilan di log bro

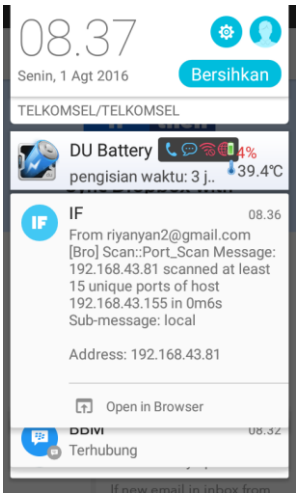
4. Tampilan *email admin* saat terjadi serangan

Gambar 4- 62 Tampilan notifikasi *email* admin



Gambar 4- 63 Tampilan isi notifikasi *email*

5. Tampilan notifikasi yang masuk ke *handphone admin* saat terjadi serangan.



Gambar 4- 64 Tampilan notifikasi yang masuk ke handphone admin

4.3.4.2 Pengujian Menggunakan FTP Brute-force

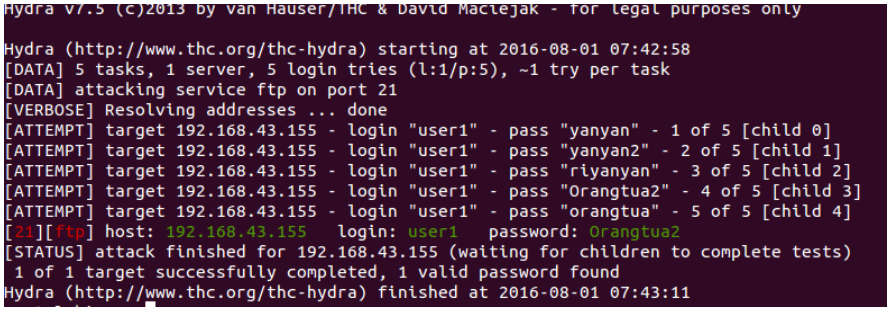
Pada tahap ini akan dilakukan pengujian menggunakan metode serangan *FTP Brute-force* untuk mendapatkan *password* dan *username server*, berikut pengujian dan hasil pengujiannya.

1. Buka *terminal*

2. Lalu masukkan perintah seperti *hydra –l user1 –P /root/password.lst –vV ftp://192.168.43.155*

Gambar 4- 65 Perintah serangan hydra

3. Setelah itu akan muncul *username* dan *password* hasil serangan tersebut



Gambar 4- 66 Hadir serangan hydra

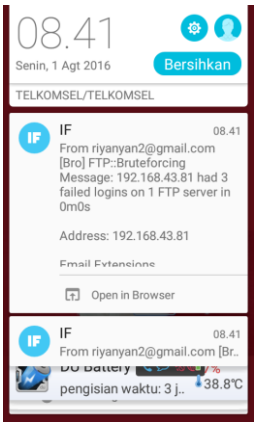
4. Tampilan *log Bro* saat terjadi serangan dan terdeteksi

Gambar 4- 67 Tampilan pada log bro

5. Tampilan *email admin* saat terjadi serangan

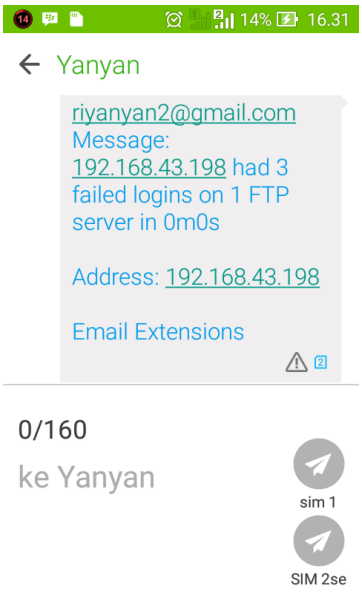
Gambar 4- 68 Tampilan notifikasi yang masuk ke *email*

6. Tampilan notifikasi yang masuk ke *handphone admin* saat terjadi serangan



Gambar 4- 69 Tampilan notifikasi yang masuk ke handphone admin

7. Tampilan notifikasi yang masuk ke *sms client*



Gambar 4- 70 Tampilan notifikasi ke sms client dari serangan FTP Brute-force

8. Tindakan selanjutnya untuk mem-*block* serangan tersebut, kita menggunakan aplikasi bawaan dari *CentOS 7* yaitu *firewalld*, masukkan perintah seperti seperti di bawah pada terminal

```
[root@DESKTOP-C3E1G4P ~]# systemctl start firewalld
[root@DESKTOP-C3E1G4P ~]# firewall-cmd --permanent --direct --add-rule ipv4 filter INPUT_direct 0 -p tcp --dport 21 -m state --state NEW -m recent --s
et
success
[root@DESKTOP-C3E1G4P ~]# firewall-cmd --permanent --direct --add-rule ipv4 filter INPUT_direct 1 -p tcp --dport 21 -m state --state NEW -m recent --u
pdate --seconds 20 --hitcount 3 -j REJECT --reject-with tcp-reset
success
[root@DESKTOP-C3E1G4P ~]# firewall-cmd --reload
success
```

Gambar 4- 71 Perintah block serangan FTP Brute-force

Maksud perintah di atas adalah :

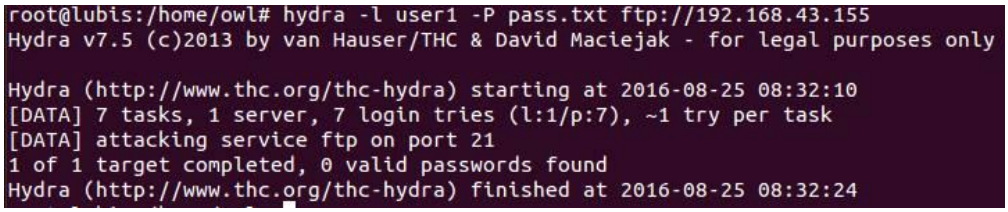
Systemctl start firewalld = untuk menjalankan firewalld

Firewall-cmd –permanent –direct –add-rule ipv4 INPUT_direct 0 –p tcp --dport 21 –a state – state NEW –a recent –set = perintah untuk membuat rule dan menjalankan port 21 pada server

Firewall-cmd --permanent --direct --add-rule ipv4 INPUT_direct 1 –p tcp --dport 21 –a state --state NEW -a recent --update --second 20 --hitcount 3 –j REJECT --reject-with tcp-reset = perintah utnuk membuat rule dimana jika mengakses ftp server dan melakukan kesalahan 3 kali untuk memasukkan *password* atau *username* maka akan di *reject*

Firewall--cmd –reload = perintah untuk *restart firewalld*

9. Hasil serangan setelah di-*block* seperti gambar 4-60

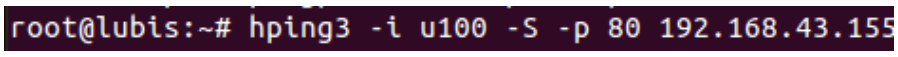


Gambar 4- 72 Serangan berhasil di-block username dan password gagal didapatkan

4.3.4.3 Pengujian Menggunakan Serangan DOS (Denial of Service)

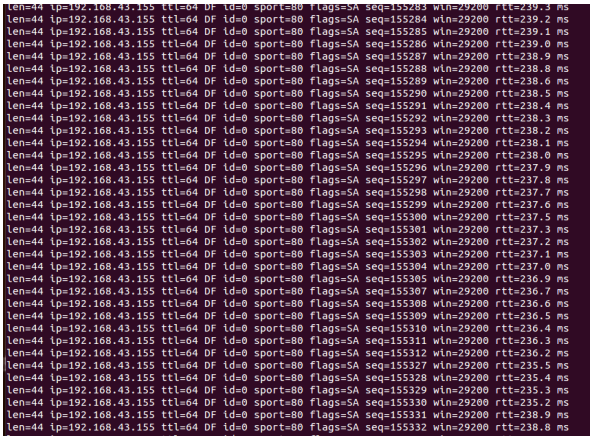
Pada tahap ini akan dilakukan pengujian menggunakan serangan *dos* dengan aplikasi *hping3* yang bertujuan untuk melumpuhkan *server,* berikut pengujian dan hasil pengujiannya :

1. Pertama buka *terminal* lalu ketikkan perintah seperti gambar4-59.



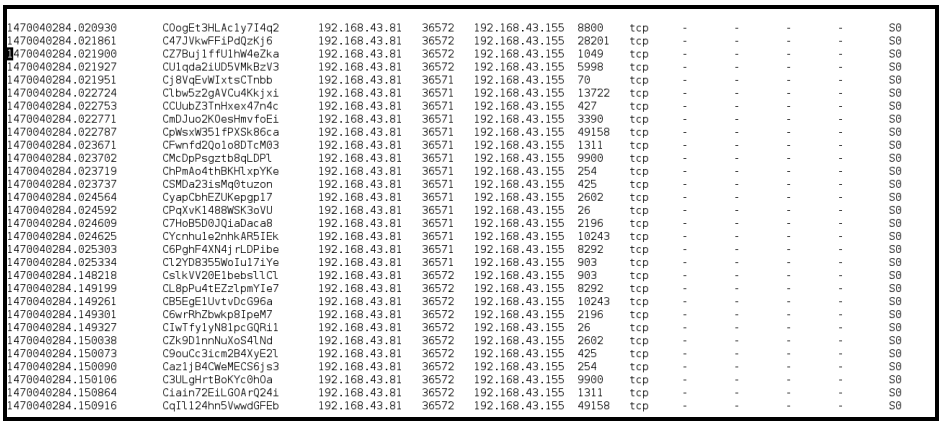
Gambar 4- 73 Perintah serangan hping3

2. Setelah itu *hping3* akan melakukan serangan terhadap target



Gambar 4- 74 Serangan hping3

3. Tampilan *log* saat terjadi serangan *DOS (Denial of Service)*

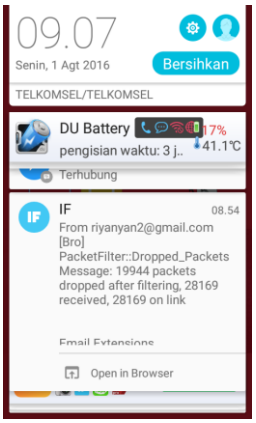


Gambar 4- 75 Tampilan log saat serangan *dos*

4. Tampilan notifikasi ke *email admin* saat terjadi serangan *dos*

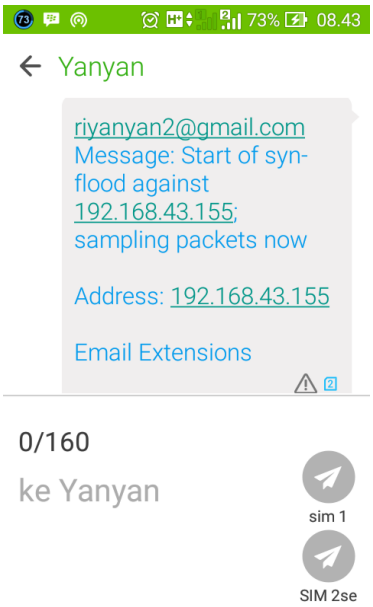
Gambar 4- 76 Tampilan *email saat serangan dos*

5. Tampilan notifikasi ke *handphone admin* saat terjadi serangan *dos*



Gambar 4- 77 Tampilan notifikasi yang masuk ke handphone admin

6. Notifikasi yang masuk ke *handphone client*



Gambar 4- 78 Notifikasi serangan DOS yang masuk ke handphone client

7. Tindakan selanjutnya untuk mem-*block* serangan *dos* yaitu dengan memasukkan perintah seperti gambar 4-66

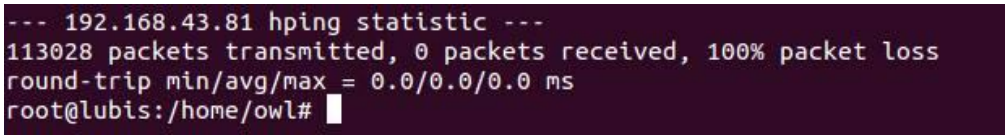
```
[root@DESKTOP-C3E1G4P ~]# firewall-cmd --permanent --add-rich-rule="rule family='ipv4' source address='192.168.43.81' reject"
success
[root@DESKTOP-C3E1G4P ~]# firewall-cmd --reload
```

Gambar 4- 79 Perintah block serangan DOS

Firewall-cmd --permanent --add-rich=”rul family=’ipv4’ source address=’192.168.43.81’ reject” = untuk mem-*block ip address* tertentu

Firewall-cmd --reload = untuk me-*restart firewalld*

8. Hasil serangan *DOS* yang di-*block* seperti gambar 4-67, terlihat 100% packet loss yang dikirimkan ke target gagal dilakukan.



Gambar 4- 80 Serangan DOS yang berhasil di-block

4.4 Hasil Pengujian Saat Menggunakan Bro dan Tidak Menggunakan Bro

Pada tabel berikut merupakan perbandingan yang terjadi terhadap serangan yang terjadi pada *server* menggunakan aplikasi *Bro* dan saat tidak menggunakan aplikasi *Bro*. Hasil tersebut dapat dilihat dari perbandingan tabel 4-4 :

Tabel 4- 4 Kesimpulan dan hasil pengujian

NO	Jenis Serangan	Status	
		Tanpa Bro	Menggunakan Bro
1.	Port Scanning	Berhasil tanpa ada notifikasi	Berhasil dengan notifikasi
2.	FTP Brute-force	Berhasil tanpa ada notifikasi	Berhasil dengan notifikasi
3.	Denial of Service (DOS)	Berhasil tanpa ada notifikasi	Berhasil dengan notifikasi

Pada tabel 4-4 menjelaskan dari hasil pengujian yang dilakukan pada *server* saat menggunakan *Bro* dan tidak menggunakan *Bro*. Saat tidak menggunakan *Bro*, serangan berhasil tetapi tidak ada notifikasi ke *admin*, sedangan saat menggunakan *Bro* serangan tetap berhasil tetapi terdeteksi oleh *admin*, sehingga *admin* dapat melakukan pencegahan untuk kerusakan yang lebih parah.

BAB 5
KESIMPULAN

5.1 Kesimpulan

Berdasarkan hasil pengujian pada bab 4 dapat disimpulkan sebagai berikut :

1. Pemasangan aplikasi *Bro* untuk menambah keamanan pada *server* menggunakan *Linux Centos 7 64 bit* bisa dilakukan, terlihat jika terjadi serangan terhadap *server* serangan tersebut tercatat di dalam *file* log dan dikirimkan ke *email admin*
2. Pengintegrasian *Bro-ids* dengan SMS gateway dapat dilakukan menggunakan aplikasi IFTTT
3. Hasil notifikasi mengenai jaringan yang dikelola baik saat terjadi serangan maupun saat tidak terjadi serangan berhasil dilakukan.

5.2 Saran

Saran penulis dari proyek akhir ini adalah :

1. Untuk pengembangan selanjutnya bisa dipadukan dengan *IPS* untuk menindak lanjuti serangan yang terdeteksi seperti melakukan block terhadap paket, *ip address* atau *mac address* penyerang, untuk *IPS* sendiri bisa digabungkan dengan aplikasi lainnya
2. Untuk notifikasi bisa tidak hanya melalui *handhphone* tetapi juga bisa ditambahkan melalui *media social* yaitu dengan menambah *API media social* yang diinginkan atau menggunakan aplikasi *IFTTT*
3. Untuk pengujian selanjutnya dapat ditambahkan dengan metode penyerangan lainnya misalnya, *phising*, *arp spoofing*, *sql injection* dan lain sebagainya.

DAFTAR PUSTAKA

[1] S.Gupta, "A Graphical User Interface Framework for Detection Intruions Using Bro IDS," Juny 2012. [Online]. Available: http://dspace.thapar.edu:8080/dspace/bitsream/10266/1891/3/1891.pdf. [Accessed 26 January 2016].

[2] K. A. Hermawan, "Implementasi intrusion Prevention System Dalam Jaringan menggunakan Suricata pada OS Ubuntu," Politeknik Telkom, 2012. [Online]. Available: http://www.academia.edu/9585705/P_IMPLEMENTASI_PREVENTION_SYSTEM_DALAM_JARINGA_MENGGUNAKAN_SURICATA_PADA_OS_UBUNTU-PROGRAM_STUDI_TEKNIK_KOMPUTER_JURUSAN_TEKNOLOGI_INFORMASI_POLITEKNIK_TELKOM_BANDUNG_2012. [Accessed 29 july 2016].B. Fnu, "Jalan Tikus," 18 march 2013. [Online]. Available:

[3] J. Gondohanindijo, "unaki," 3 september 2012. [Online]. Available: www.unaki.ac.id/ejournal/index.php/jurnal-informatika/article/download/81/80. [Accessed 23 february 2016]."Technical Term," 12 August 2009. [Online]. Available:

[4] J. Hutchen, "Kali Linux Network Scanning Cookbook," in *Kali Linux Network Scanning Cookbook*, Birmingham, Packet Publishing Ltd, 2014.

[5] K. E. Pramudita, "Brute Force Attack dan Penerapannya pada Password Cracking," *Brute Force Attack dan Penerapannya pada Password Cracking,* p. 1, 2011.

[6] http://www.techterms.com/definition/thirdparty. [Accessed 3 Juny 2016].

[7] http://www.jalantikus.com/news/pengertian-dan-fitur-ifttt. [Accessed 3 juny 2016].

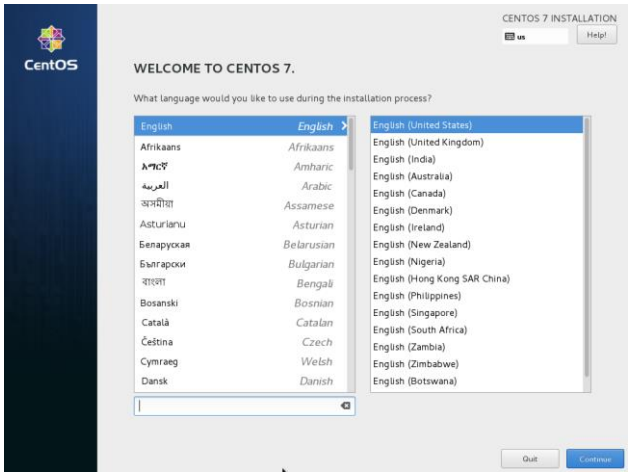
LAMPIRAN

Lampiran 1 Instalasi Linux CentOs

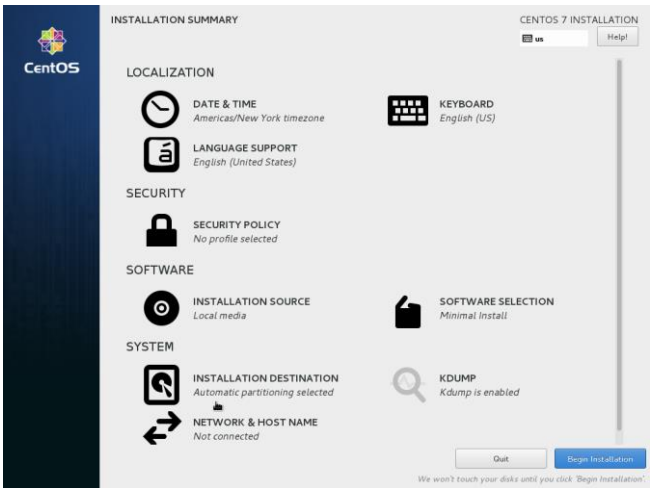
1. Sambungkan *USB Installer* yang berisi *ISO Linux CentOS 7 64 bit* ke laptop yang diinstal, di sini penulis menggunakan aplikasi *Win32Diskimager-0.9.5*. Kemudian *restart* laptop dan konfigurasi pada *BIOS* untuk membaca *USB Installer* saat *booting*

2. Pada tampilan menu *boot CentOS 7* pilih *Install CentOS 7*, lalu tekan *enter*

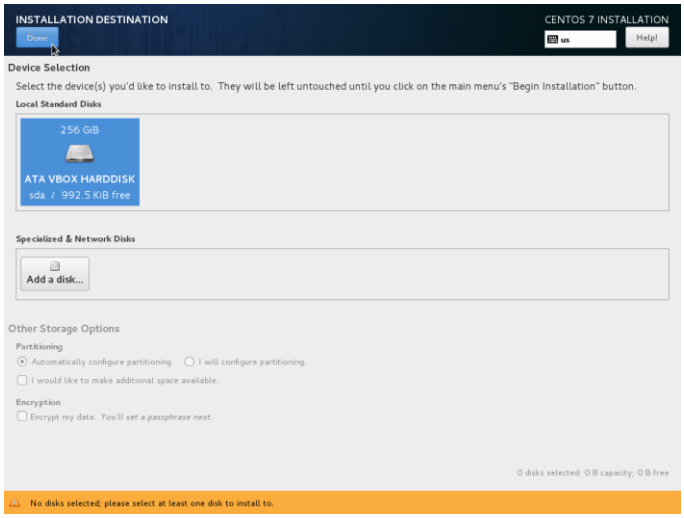
3. Setelah itu pilih bahasa yang akan digunakan, lalu klik *continue*



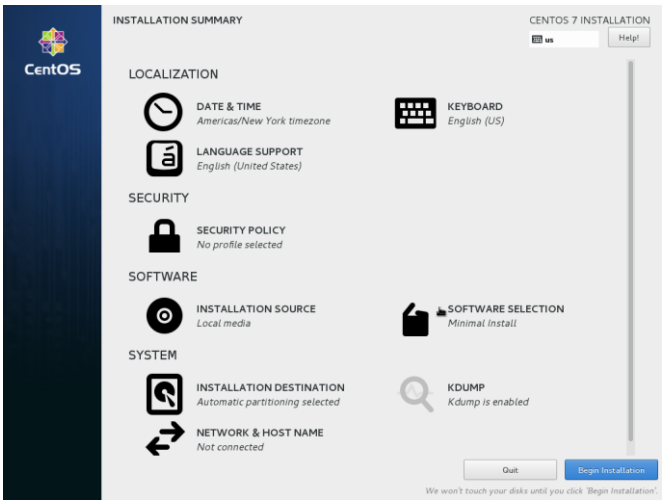
4. Kemudian klik *INSTALLATION DESTINATION*



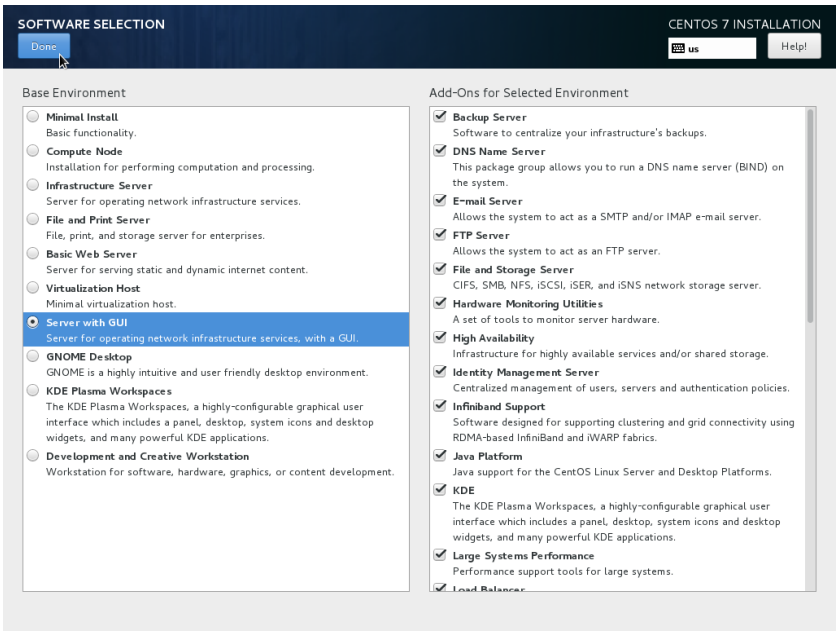
5. Lalu pilih *harddisk* yang akan digunakan, lalu klik *done*



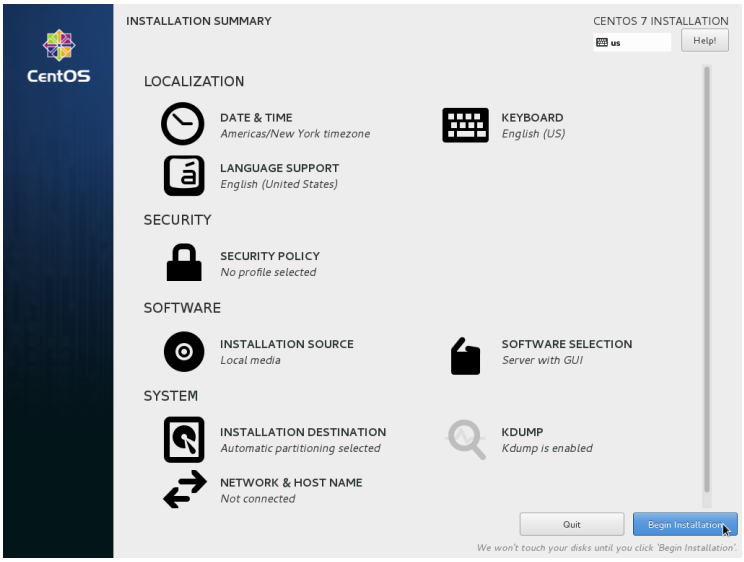
6. Kemudian klik *SOFTWARE SELECTION*



7. Lalu pilih *Server with GUI* dan centang *service* yang dibutuhkan lalu klik *done*

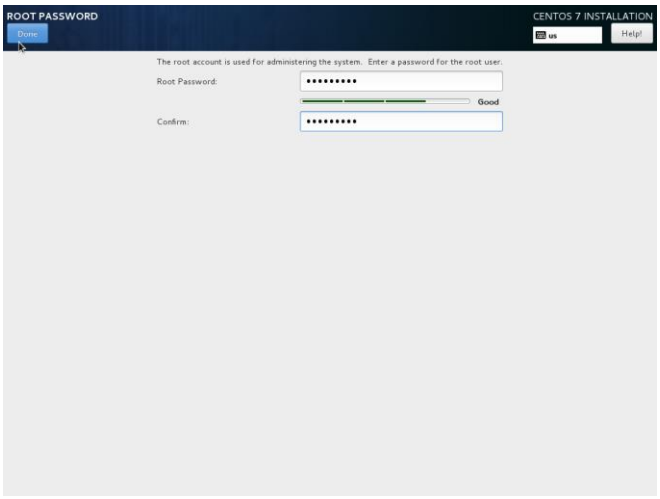


8. Kemudian klik *Begin Installation*

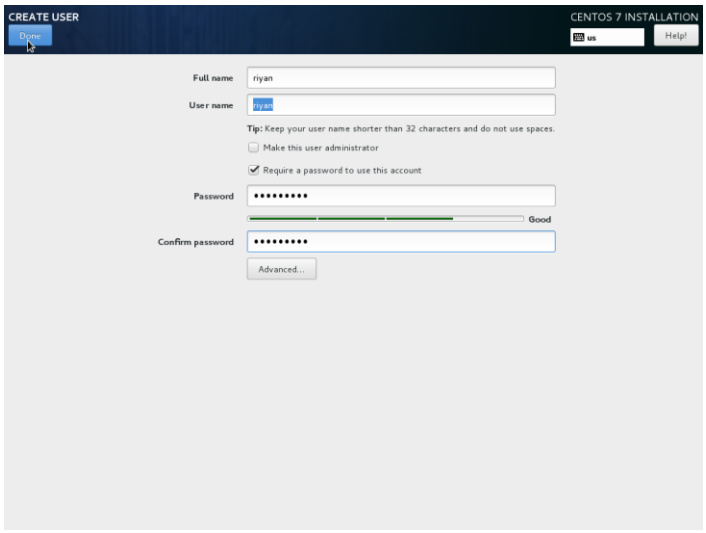


9. Setelah itu klik *ROOT PASSWORD* untuk membuat *password* untuk *root*

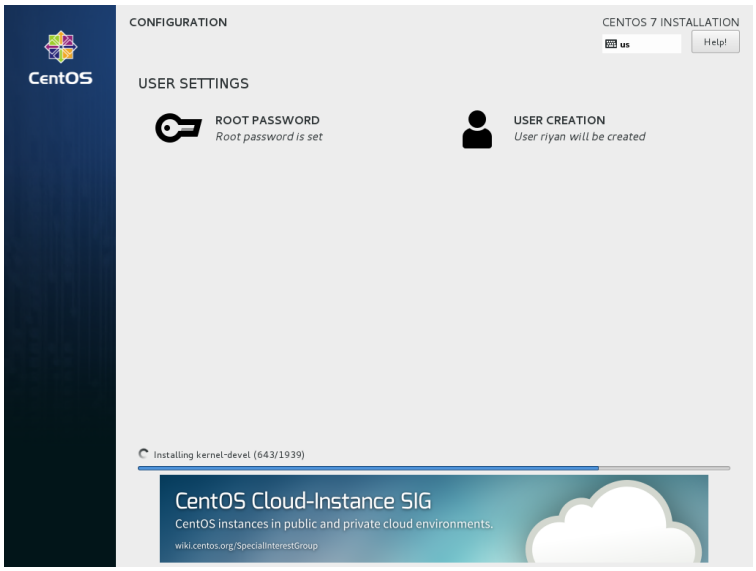
10. Kemudian masukkan *password* di kolom *Root Password* dan di kolom *Confirm*, lalu klik *Done*



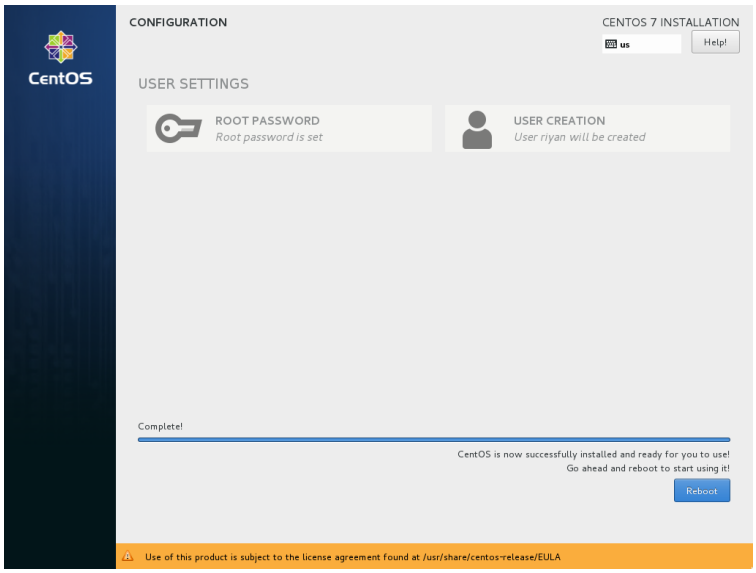
11. Setelah itu klik *USER CREATION* untuk membuat *user* baru



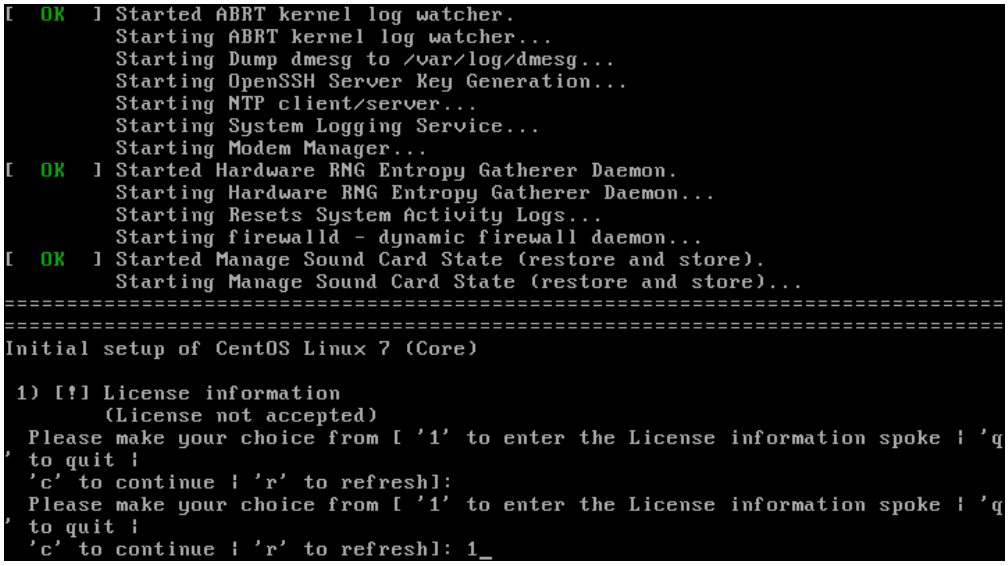
12. Setelah itu isikan pada kolom *Full name* dan kolom *User name* untuk *user name* yang akan digunakan oleh pengguna yang baru. Kemudian isikan kolom *Password* dan Confirm *password* untuk *password* yang akan digunakan oleh *user* baru



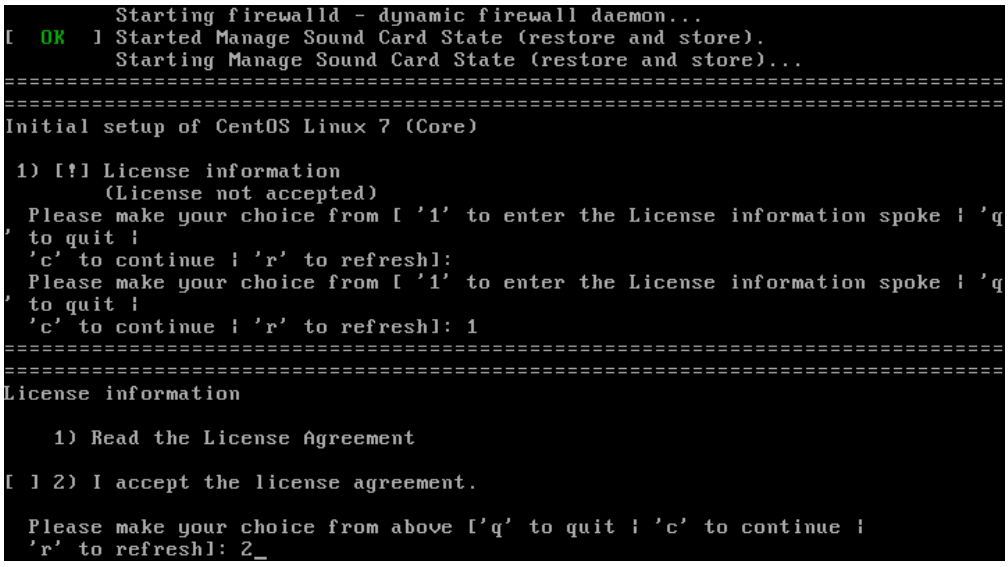
13. Setelah Instalasi selesai klik *Reboot*



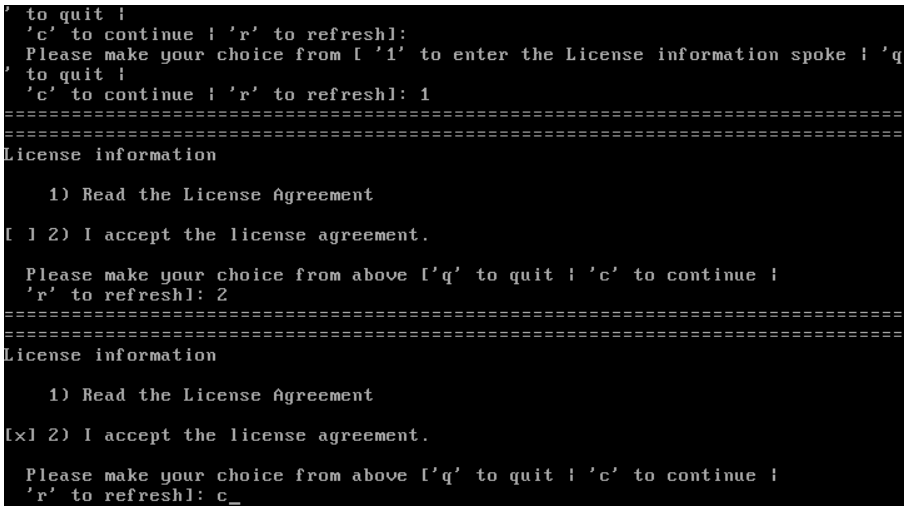
14. Setelah itu tekan angka 1 kemudian tekan huruf c lalu tekan *enter*



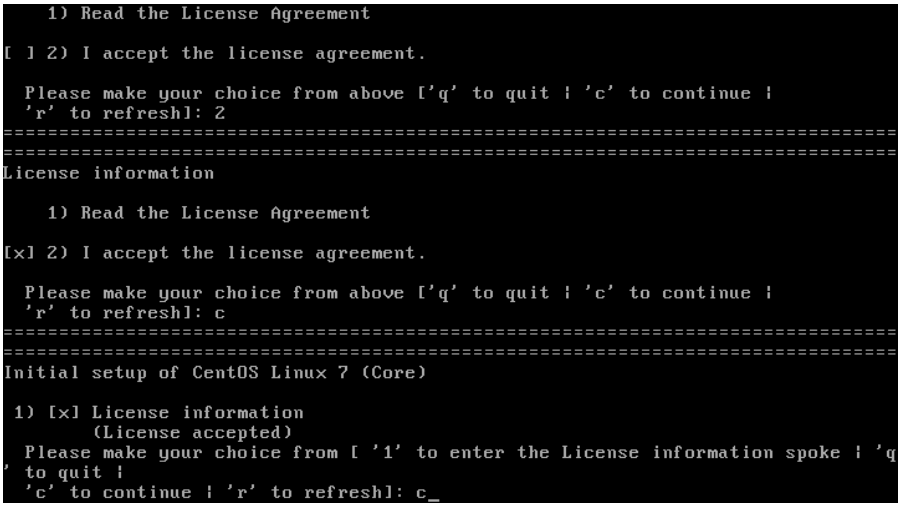
15. Setelah itu tekan angka 2 kemudian tekan *enter*



16. Setelah itu tekan huruf c kemudian tekan *enter*



17. Setelah itu maka *Linux CentOS 7 64 bit* telah selesai ter-*install*



18. Tampilan setelah *login* di CentOS

Lampiran 2 Installasi Bro-IDS pada Server

Berikut langkah – langkah instalasi *Bro* pada *server Linux CentOS 7 64-bit* :

1. *Install packet – packet* yang diperlukan agar aplikasi *Bro* dapat berjalan dengan baik pada *server Linux CentOS*

```
Yum install cmake make gcc gcc-c++ bison libcap-devel openssl-devel
python-devel swigz zlib-devel perl
```

2. Kemudian *Install packet GeoIP-devel*

```
[root@localhost ~]# yum install GeoIP-devel
```

3. Setelah itu *download* aplikasi *GeoLiteCity*

```
[root@localhost download]# wget http://geolite.maxmind.com/download/geoip/database/GeoLiteCity.dat.gz
```

4. Pindahkan *GeoLiteCity.dat* menuju *folder /usr/local/share/GeoIP/GeoIP.dat*

```
[root@localhost download]#  mv GeoLiteCity.dat /usr/share/GeoIP/GeoIPCity.dat
```

5. Kemudian *install* aplikasi *gawk* dengan perintah *yum install gawk*

```
[root@localhost download]# yum install gawk
```

6. Kemudian *install gperftools* dengan perintah *yum install gperftools*

```
[root@localhost download]# yum install gperftools
```

7. Kemudian *download* aplikasi *ipsumdump*

```
[root@localhost download]# wget http://www.read.seas.harvard.edu/~kohler/ipsumdump/ipsumdump-1.85.tar.gz
```

8. Setelah itu masuk ke *folder ipsumdump* yang sudah di-*download* tadi lalu ketikkan perintah *./configure –prefix=/usr/*

```
[root@localhost ipsumdump-1.85]# ./configure --prefix=/usr/
```

9. Kemudian ketikkan perintah *make* di dalam *folder* aplikasi tersebut

```
[root@localhost ipsumdump-1.85]# make
```

10. Setelah itu ketikkan kembali perintah *make install* untuk mulai meng-*install* aplikasi tersebut

```
[root@localhost ipsumdump-1.85]# make install
```

11. Setelah itu *download* aplikasi *Bro* pada *website* *www.bro.org/release/bro-2.4.1.tar.gz*.

```
[root@localhost download]# wget https://www.bro.org/downloads/release/bro-2.4.1.tar.gz
```

12. Kemudian masuk ke *folder Bro* setelah itu masukkan perintah *./configure*

```
[root@localhost bro-2.4.1]# ./configure
```

13. Kemudian masukkan perintah *make*

```
[root@localhost bro-2.4.1]# make
```

14. Setelah itu masukkan kembali perintah *make install*, maka aplikasi *Bro* akan ter-*install* pada *server Linux CentOS*.

```
[root@localhost bro-2.4.1]# make install
```

Lampiran 3 Konfigurasi Sendmail pada Server

1. *Install packet – packet sendmail* dengan perintah *yum install sendmail mailutils sendmail-bin*

```
[root@DESKTOP-C3E1G4P ~]# yum install sendmail mailutils sendmail-bin
```

2. Setelah itu buat *folder authinfo* sebagai tempat autentikasi *email* yang akan digunakan

```
[root@DESKTOP-C3E1G4P ~]# mkdir -m 700 /etc/mail/authinfo
```

3. Setelah itu masuk ke *folder* yang telah dibuat tadi

```
[root@DESKTOP-C3E1G4P ~]# cd /etc/mail/authinfo/
```

4. Kemudian buat sebuah *file* dengan nama *gmail-auth*

```
[root@DESKTOP-C3E1G4P authinfo]# nano gmail-auth
```

5. Pada *file* yang telah dibuat tadi isikan alamat *email* serta *password* yang digunakan setelah itu simpan *file* tersebut

```
  GNU nano 2.3.1            File: gmail-auth

AuthInfo: "U:root" "I:riyanyan2@gmail.com" "P:Orangtua2"
```

6. Setelah itu buat *database gmail-auth* dengan perintah *makemap hash*

```
[root@DESKTOP-C3E1G4P authinfo]# makemap hash gmail-auth < gmail-auth
```

7. Kemudian *edit file sendmail.mc*

```
[root@DESKTOP-C3E1G4P authinfo]# nano /etc/mail/sendmail.mc
```

8. Lalu isikan seperti gambar dibawah

```
 GNU nano 2.3.1                          File: /etc/mail/sendmail.mc

dnl #
dnl FEATURE(masquerade_envelope)dnl
dnl #
dnl # masquerade not just @mydomainalias.com, but @*.mydomainalias.com as well
dnl #
dnl FEATURE(masquerade_entire_domain)dnl
dnl #
dnl MASQUERADE_DOMAIN(localhost)dnl
dnl MASQUERADE_DOMAIN(localhost.localdomain)dnl
dnl MASQUERADE_DOMAIN(mydomainalias.com)dnl
dnl MASQUERADE_DOMAIN(mydomain.lan)dnl

define(`SMART_HOST',`[smtp.gmail.com]')dnl
define(`RELAY_MAILER_ARGS', `TCP $h 587')dnl
define(`ESMTP_MAILER_ARGS', `TCP $h 587')dnl
define(`confAUTH_OPTIONS', `A p')dnl
TRUST_AUTH_MECH(`EXTERNAL DIGEST-MD5 CRAM-MD5 LOGIN PLAIN')dnl
define(`confAUTH_MECHANISMS', `EXTERNAL GSSAPI DIGEST-MD5 CRAM-MD5 LOGIN PLAIN')dnl
FEATURE(`authinfo',`hash -o /etc/mail/authinfo/gmail-auth.db')dnl

MAILER(smtp)dnl
MAILER(procmail)dnl
dnl MAILER(cyrusv2)dnl

###########################################################
```

9. Setelah itu masukkan perintah *make –C*

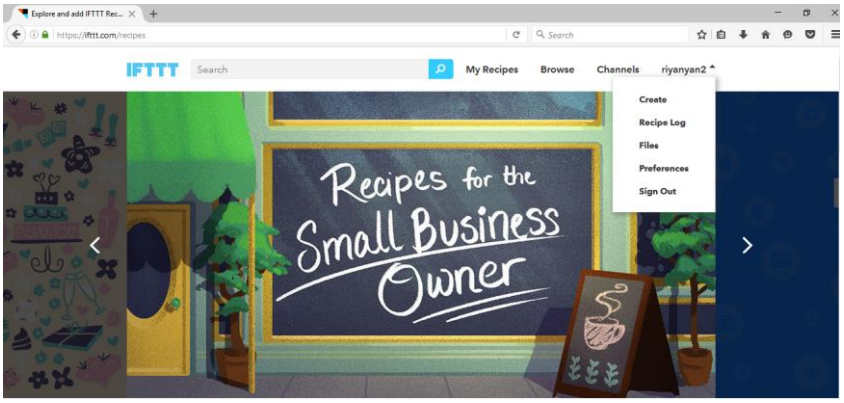
```
[root@DESKTOP-C3E1G4P authinfo]# make -C /etc/mail
```

10. Kemudian *restart sendmail.*

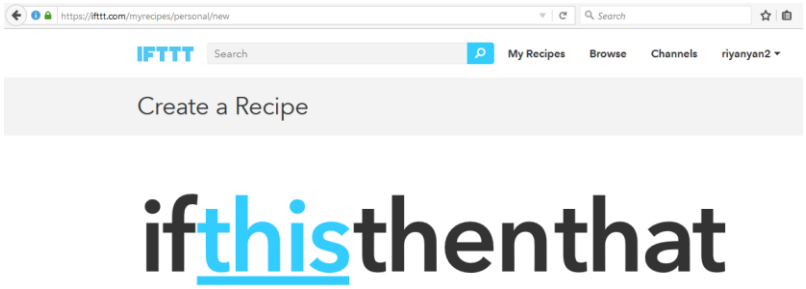
```
[root@DESKTOP-C3E1G4P authinfo]# systemctl restart sendmail
```

Lampiran 4 Cara membuat Recipe pada IFTTT

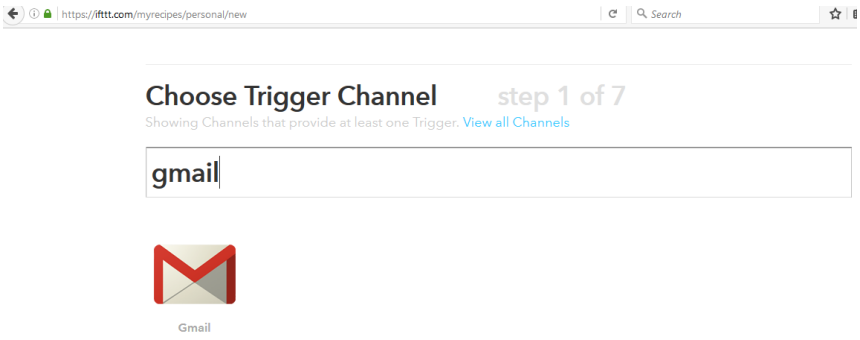
1. Klik nama *user* yang telah digunakan lalu pilih *create*



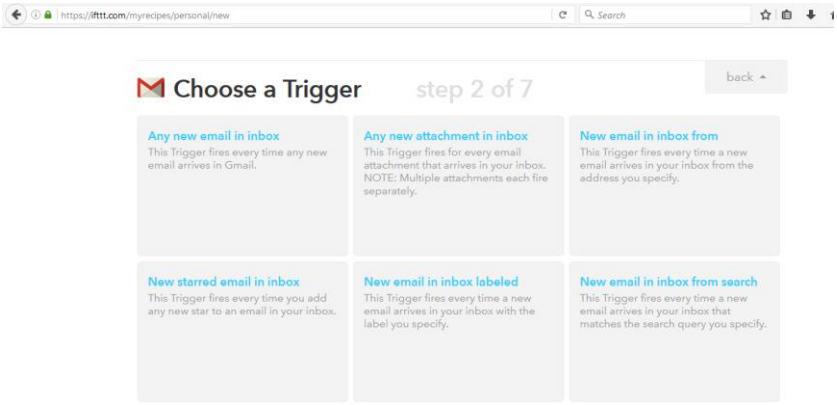
2. Setelah itu klik tulisan *this* yang digaris bawahi berwarna biru



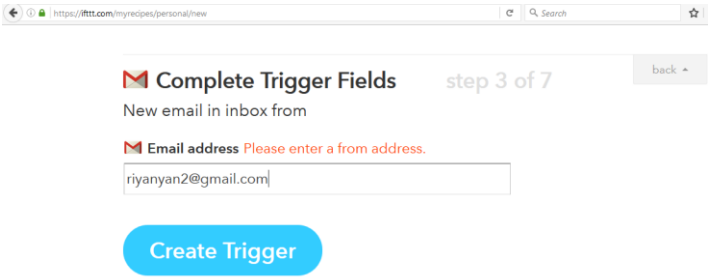
3. Pada kolom *Choose Trigger Channel* ketikkan *Gmail*, lalu klik *Gmail* tersebut



4. Setelah itu klik *New email in inbox from*

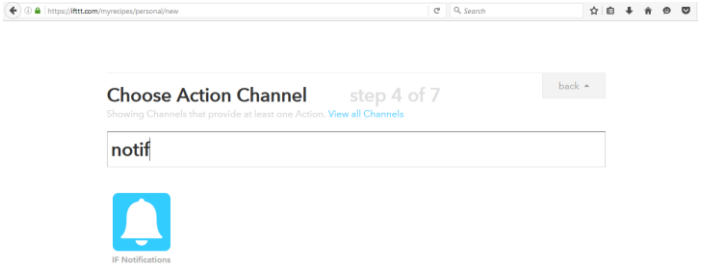


5. Kemudian isikan alamat *email* yang akan dijadikan sebagai acuan untuk mengirim notifikasi, setelah itu klik *Create Trigger*

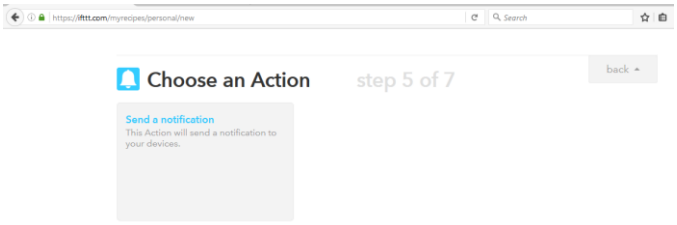


6. Kemudian klik tulisan *that* dengan garis bawah dan berwarna biru

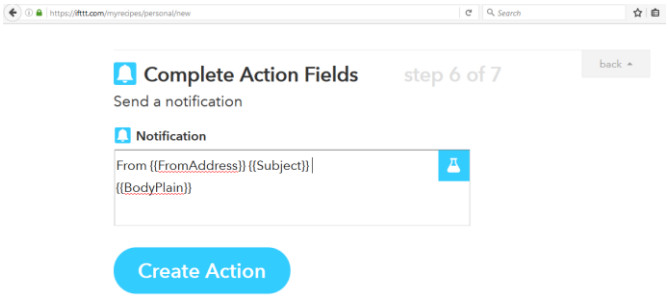
7. Setelah itu pada kolom *choose Action Channel* ketikkan *notif*, lalu pilih *IF Notificaions*



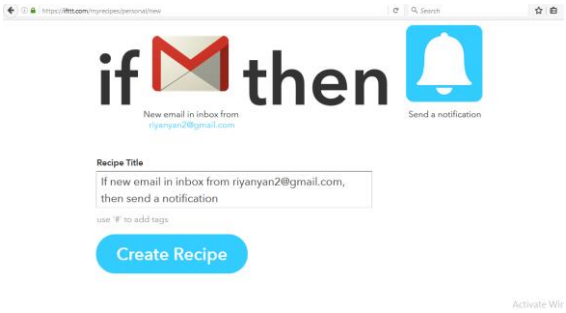
8. Kemudian klik *send a notification*



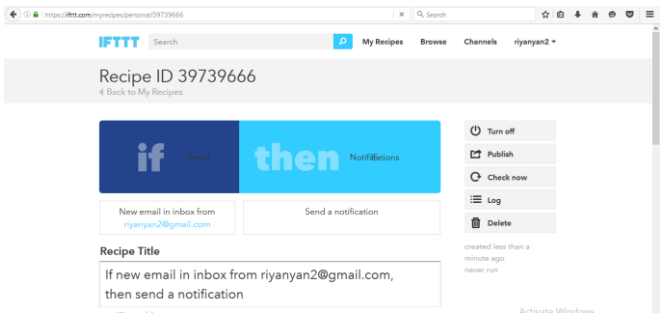
9. Setelah itu isikan pesan dengan menambahkan *{{FromAddres}}, {{Subject}} dan {{BodyPlain}}*, kemudian klik *create action*



10. Kemudian klik *Create Recipe*

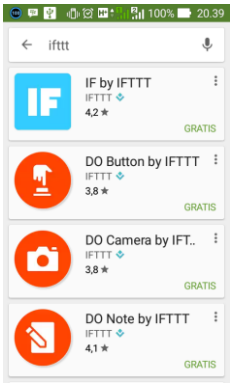


11. Setelah itu untuk mengaktifkannya, ubah *Turn off* menjadi *Turn on* dengan mengkliknya.

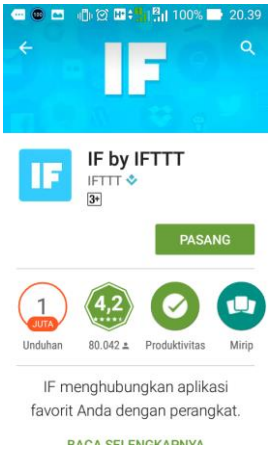


Lampiran 5 Konfigurasi *IFTTT* pada *Handphone Admin*

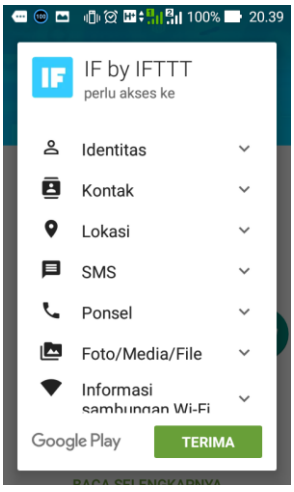
1. Buka *playstore* dan ketikkan di kotak pencarian *IFTTT*



2. Setelah itu klik PASANG untuk melakukan instalasi pada *handphone*

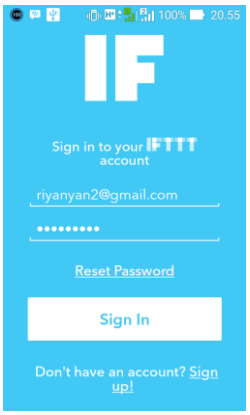


3. Kemudian klik TERIMA untuk melanjutkan instalasi

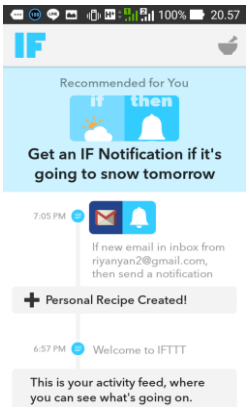


4. Setelah itu selesai ter-*install* klik BUKA, untuk menjalankan aplikasi tersebut

5. Kemudian masukkan *email/username* dan *password* yang sudah kita daftarkan sebelumnya pada *web IFTTT*



6. Setelah itu klik *icon* pada pojok kanan atas



7. Lalu pilih *recipe* yang akan digunakan di sini untuk mengaktifkan r*ecipe* yang telah dibat tinggal mengklik *recipe* tersebut.

Lampiran 6 Rule Scan.bro

Berikut merupakan *script* dari *rule scan.bro* :

```
##! Scan detector ported from Bro 1.x.
##!
##! This script has evolved over many years and is quite a mess right now. We
##! have adapted it to work with Bro 2.x, but eventually Bro 2.x will
##! get its own rewritten and generalized scan detector.

@load base/frameworks/notice/main

module Scan;

export {
        redef enum Notice::Type += {
                ## The source has scanned a number of ports.
                PortScan,
                ## The source has scanned a number of addresses.
                AddressScan,
                ## Apparent flooding backscatter seen from source.
                BackscatterSeen,

                ## Summary of scanning activity.
                ScanSummary,
                ## Summary of distinct ports per scanner.
                PortScanSummary,
                ## Summary of distinct low ports per scanner.
                LowPortScanSummary,

                ## Source reached :bro:id:`Scan::shut_down_thresh`
                ShutdownThresh,
                ## Source touched privileged ports.
                LowPortTrolling,
        };

        # Whether to consider UDP "connections" for scan detection.
```

```
# Can lead to false positives due to UDP fanout from some P2P apps.
const suppress_UDP_scan_checks = F &redef;

const activate_priv_port_check = T &redef;
const activate_landmine_check = F &redef;
const landmine_thresh_trigger = 5 &redef;

const landmine_address: set[addr] &redef;

const scan_summary_trigger = 25 &redef;
const port_summary_trigger = 20 &redef;
const lowport_summary_trigger = 10 &redef;

# Raise ShutdownThresh after this many failed attempts
const shut_down_thresh = 100 &redef;

# Which services should be analyzed when detecting scanning
# (not consulted if analyze_all_services is set).
const analyze_services: set[port] &redef;
const analyze_all_services = T &redef;

# Track address scaners only if at least these many hosts contacted.
const addr_scan_trigger = 0 &redef;

# Ignore address scanners for further scan detection after
# scanning this many hosts.
# 0 disables.
const ignore_scanners_threshold = 0 &redef;

# Report a scan of peers at each of these points.
const report_peer_scan: vector of count = {
       20, 100, 1000, 10000, 50000, 100000, 250000, 500000, 1000000,
} &redef;
```

```
const report_outbound_peer_scan: vector of count = {
        100, 1000, 10000,
} &redef;

# Report a scan of ports at each of these points.
const report_port_scan: vector of count = {
        50, 250, 1000, 5000, 10000, 25000, 65000,
} &redef;

# Once a source has scanned this many different ports (to however many
# different remote hosts), start tracking its per-destination access.
const possible_port_scan_thresh = 20 &redef;

# Threshold for scanning privileged ports.
const priv_scan_trigger = 5 &redef;
const troll_skip_service = {
        25/tcp, 21/tcp, 22/tcp, 20/tcp, 80/tcp,
} &redef;

const report_accounts_tried: vector of count = {
        20, 100, 1000, 10000, 100000, 1000000,
} &redef;

const report_remote_accounts_tried: vector of count = {
        100, 500,
} &redef;

# Report a successful password guessing if the source attempted
# at least this many.
const password_guessing_success_threshhold = 20 &redef;

const skip_accounts_tried: set[addr] &redef;
```

```
const addl_web = {
        81/tcp, 443/tcp, 8000/tcp, 8001/tcp, 8080/tcp, }
&redef;

const skip_services = { 113/tcp, } &redef;
const skip_outbound_services = { 21/tcp, addl_web, }
        &redef;

const skip_scan_sources = {
        255.255.255.255,        # who knows why we see these, but we do
} &redef;

const skip_scan_nets: set[subnet] = {} &redef;

# List of well known local server/ports to exclude for scanning
# purposes.
const skip_dest_server_ports: set[addr, port] = {} &redef;

# Reverse (SYN-ack) scans seen from these ports are considered
# to reflect possible SYN-flooding backscatter, and not true
# (stealth) scans.
const backscatter_ports = {
        80/tcp, 8080/tcp, 53/tcp, 53/udp, 179/tcp, 6666/tcp, 6667/tcp,
} &redef;

const report_backscatter: vector of count = {
        20,
} &redef;

global check_scan:
        function(c: connection, established: bool, reverse: bool): bool;

# The following tables are defined here so that we can redef
```

```
# the expire timeouts.
# FIXME: should we allow redef of attributes on IDs which
# are not exported?

# How many different hosts connected to with a possible
# backscatter signature.
global distinct_backscatter_peers: table[addr] of table[addr] of count
        &read_expire = 15 min;

# Expire functions that trigger summaries.
global scan_summary:
        function(t: table[addr] of set[addr], orig: addr): interval;
global port_summary:
        function(t: table[addr] of set[port], orig: addr): interval;
global lowport_summary:
        function(t: table[addr] of set[port], orig: addr): interval;

# Indexed by scanner address, yields # distinct peers scanned.
# pre_distinct_peers tracks until addr_scan_trigger hosts first.
global pre_distinct_peers: table[addr] of set[addr]
        &read_expire = 15 mins &redef;

global distinct_peers: table[addr] of set[addr]
        &read_expire = 15 mins &expire_func=scan_summary &redef;
global distinct_ports: table[addr] of set[port]
        &read_expire = 15 mins &expire_func=port_summary &redef;
global distinct_low_ports: table[addr] of set[port]
        &read_expire = 15 mins &expire_func=lowport_summary &redef;

# Indexed by scanner address, yields a table with scanned hosts
# (and ports).
global scan_triples: table[addr] of table[addr] of set[port];
```

```
        global remove_possible_source:
                function(s: set[addr], idx: addr): interval;
        global possible_scan_sources: set[addr]
                &expire_func=remove_possible_source &read_expire = 15 mins;

        # Indexed by source address, yields user name & password tried.
        global accounts_tried: table[addr] of set[string, string]
                &read_expire = 1 days;

        global ignored_scanners: set[addr] &create_expire = 1 day &redef;

        # These tables track whether a threshold has been reached.
        # More precisely, the counter is the next index of threshold vector.
        global shut_down_thresh_reached: table[addr] of bool &default=F;
        global rb_idx: table[addr] of count
                        &default=1 &read_expire = 1 days &redef;
        global rps_idx: table[addr] of count
                        &default=1 &read_expire = 1 days &redef;
        global rops_idx: table[addr] of count
                        &default=1 &read_expire = 1 days &redef;
        global rpts_idx: table[addr,addr] of count
                        &default=1 &read_expire = 1 days &redef;
        global rat_idx: table[addr] of count
                        &default=1 &read_expire = 1 days &redef;
        global rrat_idx: table[addr] of count
                        &default=1 &read_expire = 1 days &redef;
}

global thresh_check: function(v: vector of count, idx: table[addr] of count,
                              orig: addr, n: count): bool;
global thresh_check_2: function(v: vector of count,
                              idx: table[addr,addr] of count, orig: addr,
                              resp: addr, n: count): bool;
```

```
function scan_summary(t: table[addr] of set[addr], orig: addr): interval
        {
        local num_distinct_peers = orig in t ? |t[orig]| : 0;

        if ( num_distinct_peers >= scan_summary_trigger )
                NOTICE([$note=ScanSummary, $src=orig, $n=num_distinct_peers,
                        $identifier=fmt("%s", orig),
                        $msg=fmt("%s scanned a total of %d hosts",
                                        orig, num_distinct_peers)]);

        return 0 secs;
        }

function port_summary(t: table[addr] of set[port], orig: addr): interval
        {
        local num_distinct_ports = orig in t ? |t[orig]| : 0;

        if ( num_distinct_ports >= port_summary_trigger )
                NOTICE([$note=PortScanSummary, $src=orig, $n=num_distinct_ports,
                        $identifier=fmt("%s", orig),
                        $msg=fmt("%s scanned a total of %d ports",
                                        orig, num_distinct_ports)]);

        return 0 secs;
        }

function lowport_summary(t: table[addr] of set[port], orig: addr): interval
        {
        local num_distinct_lowports = orig in t ? |t[orig]| : 0;

        if ( num_distinct_lowports >= lowport_summary_trigger )
                NOTICE([$note=LowPortScanSummary, $src=orig,
                        $n=num_distinct_lowports,
```

```
                        $identifier=fmt("%s", orig),
                        $msg=fmt("%s scanned a total of %d low ports",
                                        orig, num_distinct_lowports)]);

        return 0 secs;
        }

function clear_addr(a: addr)
        {
        delete distinct_peers[a];
        delete distinct_ports[a];
        delete distinct_low_ports[a];
        delete scan_triples[a];
        delete possible_scan_sources[a];
        delete distinct_backscatter_peers[a];
        delete pre_distinct_peers[a];
        delete rb_idx[a];
        delete rps_idx[a];
        delete rops_idx[a];
        delete rat_idx[a];
        delete rrat_idx[a];
        delete shut_down_thresh_reached[a];
        delete ignored_scanners[a];
        }

function ignore_addr(a: addr)
        {
        clear_addr(a);
        add ignored_scanners[a];
        }

function check_scan(c: connection, established: bool, reverse: bool): bool
        {
```

```
local id = c$id;

local service = "ftp-data" in c$service ? 20/tcp
                : (reverse ? id$orig_p : id$resp_p);
local rev_service = reverse ? id$resp_p : id$orig_p;
local orig = reverse ? id$resp_h : id$orig_h;
local resp = reverse ? id$orig_h : id$resp_h;
local outbound = Site::is_local_addr(orig);

# The following works better than using get_conn_transport_proto()
# because c might not correspond to an active connection (which
# causes the function to fail).
if ( suppress_UDP_scan_checks &&
     service >= 0/udp && service <= 65535/udp )
        return F;

if ( service in skip_services && ! outbound )
        return F;

if ( outbound && service in skip_outbound_services )
        return F;

if ( orig in skip_scan_sources )
        return F;

if ( orig in skip_scan_nets )
        return F;

# Don't include well known server/ports for scanning purposes.
if ( ! outbound && [resp, service] in skip_dest_server_ports )
        return F;

if ( orig in ignored_scanners)
```

```
        return F;

if ( ! established &&
        # not established, service not expressly allowed

        # not known peer set
        (orig !in distinct_peers || resp !in distinct_peers[orig]) &&

        # want to consider service for scan detection
        (analyze_all_services || service in analyze_services) )
        {
        if ( reverse && rev_service in backscatter_ports &&
             # reverse, non-priv backscatter port
             service >= 1024/tcp )
                {
                if ( orig !in distinct_backscatter_peers )
                        {
                        local empty_bs_table:
                                table[addr] of count &default=0;
                        distinct_backscatter_peers[orig] =
                                empty_bs_table;
                        }

                if ( ++distinct_backscatter_peers[orig][resp] <= 2 &&
                     # The test is <= 2 because we get two check_scan()
                     # calls, once on connection attempt and once on
                     # tear-down.

                     distinct_backscatter_peers[orig][resp] == 1 &&

                     # Looks like backscatter, and it's not scanning
                     # a privileged port.


                    thresh_check(report_backscatter, rb_idx, orig,
                                |distinct_backscatter_peers[orig]|)
                   )
                        {
                        NOTICE([$note=BackscatterSeen, $src=orig,
                                $p=rev_service,
                                $identifier=fmt("%s", orig),
                                $msg=fmt("backscatter seen from %s (%d hosts; %s)",
                                        orig, |distinct_backscatter_peers[orig]|, rev_service)]);
                        }

                if ( ignore_scanners_threshold > 0 &&
                     |distinct_backscatter_peers[orig]| >
                                ignore_scanners_threshold )
                        ignore_addr(orig);
                }

  else
                { # done with backscatter check
                local ignore = F;

                if ( orig !in distinct_peers && addr_scan_trigger > 0 )
                        {
                        if ( orig !in pre_distinct_peers )
                                pre_distinct_peers[orig] = set();

                        add pre_distinct_peers[orig][resp];
                        if ( |pre_distinct_peers[orig]| < addr_scan_trigger )
                                ignore = T;
                        }

                if ( ! ignore )
                        { # XXXXX
```

```
        if ( orig !in distinct_peers )
                distinct_peers[orig] = set() &mergeable;

        if ( resp !in distinct_peers[orig] )
                add distinct_peers[orig][resp];

        local n = |distinct_peers[orig]|;

        # Check for threshold if not outbound.
        if ( ! shut_down_thresh_reached[orig] &&
             n >= shut_down_thresh &&
             ! outbound && orig !in Site::neighbor_nets )
                {
                shut_down_thresh_reached[orig] = T;
                local msg = fmt("shutdown threshold reached for %s", orig);
                NOTICE([$note=ShutdownThresh, $src=orig,
                        $identifier=fmt("%s", orig),
                        $p=service, $msg=msg]);
                }

        else
                {
                local address_scan = F;
                if ( outbound &&
                     # inside host scanning out?
                     thresh_check(report_outbound_peer_scan, rops_idx, orig, n) )
                        address_scan = T;

                if ( ! outbound &&
                     thresh_check(report_peer_scan, rps_idx, orig, n) )
                        address_scan = T;

                if ( address_scan )
```

```
                                if ( address_scan )
                                        NOTICE([$note=AddressScan,
                                                $src=orig, $p=service,
                                                $n=n,
                                                $identifier=fmt("%s-%d", orig, n),
                                                $msg=fmt("%s has scanned %d hosts (%s)",
                                                        orig, n, service)]);

                                if ( address_scan &&
                                     ignore_scanners_threshold > 0 &&
                                     n > ignore_scanners_threshold )
                                        ignore_addr(orig);
                                }
                        }
                } # XXXX
        }

if ( established )
        # Don't consider established connections for port scanning,
        # it's too easy to be mislead by FTP-like applications that
        # legitimately gobble their way through the port space.
        return F;

# Coarse search for port-scanning candidates: those that have made
# connections (attempts) to possible_port_scan_thresh or more
# distinct ports.
if ( orig !in distinct_ports || service !in distinct_ports[orig] )
        {
        if ( orig !in distinct_ports )
                distinct_ports[orig] = set() &mergeable;

        if ( service !in distinct_ports[orig] )
                add distinct_ports[orig][service];
```

```
                                    if ( address_scan )
                                            NOTICE([$note=AddressScan,
                                                    $src=orig, $p=service,
                                                    $n=n,
                                                    $identifier=fmt("%s-%d", orig, n),
                                                    $msg=fmt("%s has scanned %d hosts (%s)",
                                                            orig, n, service)]);

                                    if ( address_scan &&
                                         ignore_scanners_threshold > 0 &&
                                         n > ignore_scanners_threshold )
                                            ignore_addr(orig);
                                    }
                                }
                        } # XXXX
                }

        if ( established )
                # Don't consider established connections for port scanning,
                # it's too easy to be mislead by FTP-like applications that
                # legitimately gobble their way through the port space.
                return F;

        # Coarse search for port-scanning candidates: those that have made
        # connections (attempts) to possible_port_scan_thresh or more
        # distinct ports.
        if ( orig !in distinct_ports || service !in distinct_ports[orig] )
                {
                if ( orig !in distinct_ports )
                        distinct_ports[orig] = set() &mergeable;

                if ( service !in distinct_ports[orig] )
                        add distinct_ports[orig][service];
```

```
                if ( |distinct_ports[orig]| >= possible_port_scan_thresh &&
                        orig !in scan_triples )
                        {
                        scan_triples[orig] = table() &mergeable;
                        add possible_scan_sources[orig];
                        }
                }

        # Check for low ports.
        if ( activate_priv_port_check && ! outbound && service < 1024/tcp &&
             service !in troll_skip_service )
                {
                if ( orig !in distinct_low_ports ||
                     service !in distinct_low_ports[orig] )
                        {
                        if ( orig !in distinct_low_ports )
                                distinct_low_ports[orig] = set() &mergeable;

                        add distinct_low_ports[orig][service];

                        if ( |distinct_low_ports[orig]| == priv_scan_trigger &&
                             orig !in Site::neighbor_nets )
                                {
                                local svrc_msg = fmt("low port trolling %s %s", orig, service);
                                NOTICE([$note=LowPortTrolling, $src=orig,
                                        $identifier=fmt("%s", orig),
                                        $p=service, $msg=svrc_msg]);
                                }

                        if ( ignore_scanners_threshold > 0 &&
                             |distinct_low_ports[orig]| >
                                        ignore_scanners_threshold )
                                ignore_addr(orig);
```

```
                        }
                }

        # For sources that have been identified as possible scan sources,
        # keep track of per-host scanning.
        if ( orig in possible_scan_sources )
                {
                if ( orig !in scan_triples )
                        scan_triples[orig] = table() &mergeable;

                if ( resp !in scan_triples[orig] )
                        scan_triples[orig][resp] = set() &mergeable;

                if ( service !in scan_triples[orig][resp] )
                        {
                        add scan_triples[orig][resp][service];

                        if ( thresh_check_2(report_port_scan, rpts_idx,
                                            orig, resp,
                                            |scan_triples[orig][resp]|) )
                                {
                                local m = |scan_triples[orig][resp]|;
                                NOTICE([$note=PortScan, $n=m, $src=orig,
                                        $p=service,
                                        $identifier=fmt("%s-%d", orig, n),
                                        $msg=fmt("%s has scanned %d ports of %s",
                                        orig, m, resp)]);
                                }
                        }
                }

        return T;
        }
```

```
# Hook into the catch&release dropping. When an address gets restored, we reset
# the source to allow dropping it again.
event Drop::address_restored(a: addr)
        {
        clear_addr(a);
        }

event Drop::address_cleared(a: addr)
        {
        clear_addr(a);
        }

# When removing a possible scan source, we automatically delete its scanned
# hosts and ports.  But we do not want the deletion propagated, because every
# peer calls the expire_function on its own (and thus applies the delete
# operation on its own table).
function remove_possible_source(s: set[addr], idx: addr): interval
        {
        suspend_state_updates();
        delete scan_triples[idx];
        resume_state_updates();

        return 0 secs;
        }

# To recognize whether a certain threshhold vector (e.g. report_peer_scans)
# has been transgressed, a global variable containing the next vector index
# (idx) must be incremented.  This cumbersome mechanism is necessary because
# values naturally don't increment by one (e.g. replayed table merges).
function thresh_check(v: vector of count, idx: table[addr] of count,
                      orig: addr, n: count): bool
        {
        if ( ignore_scanners_threshold > 0 && n > ignore_scanners_threshold )
```

```
                {
                ignore_addr(orig);
                return F;
                }

        if ( idx[orig] <= |v| && n >= v[idx[orig]] )
                {
                ++idx[orig];
                return T;
                }
        else
                return F;
        }

# Same as above, except the index has a different type signature.
function thresh_check_2(v: vector of count, idx: table[addr, addr] of count,
                        orig: addr, resp: addr, n: count): bool
        {
        if ( ignore_scanners_threshold > 0 && n > ignore_scanners_threshold )
                {
                ignore_addr(orig);
                return F;
                }

        if ( idx[orig,resp] <= |v| && n >= v[idx[orig, resp]] )
                {
                ++idx[orig,resp];
                return T;
                }
        else
                return F;
        }


event connection_established(c: connection)
        {
        local is_reverse_scan = (c$orig$state == TCP_INACTIVE);
        Scan::check_scan(c, T, is_reverse_scan);
        }

event partial_connection(c: connection)
        {
        Scan::check_scan(c, T, F);
        }

event connection_attempt(c: connection)
        {
        Scan::check_scan(c, F, c$orig$state == TCP_INACTIVE);
        }

event connection_half_finished(c: connection)
        {
        # Half connections never were "established", so do scan-checking here.
        Scan::check_scan(c, F, F);
        }

event connection_rejected(c: connection)
        {
        local is_reverse_scan = c$orig$state == TCP_RESET;

        Scan::check_scan(c, F, is_reverse_scan);
        }

event connection_reset(c: connection)
        {
        if ( c$orig$state == TCP_INACTIVE || c$resp$state == TCP_INACTIVE )
                # We never heard from one side - that looks like a scan.
```

```
                Scan::check_scan(c, c$orig$size + c$resp$size > 0,
                                 c$orig$state == TCP_INACTIVE);
        }

event connection_pending(c: connection)
        {
        if ( c$orig$state == TCP_PARTIAL && c$resp$state == TCP_INACTIVE )
                Scan::check_scan(c, F, F);
        }

# Report the remaining entries in the tables.
event bro_done()
        {
        for ( orig in distinct_peers )
                scan_summary(distinct_peers, orig);

        for ( orig in distinct_ports )
                port_summary(distinct_ports, orig);

        for ( orig in distinct_low_ports )
                lowport_summary(distinct_low_ports, orig);
        }

hook Notice::policy(n: Notice::Info)
        {
                add n$actions[Notice::ACTION_EMAIL];
        }
```

Lampiran 7 Rule FTP Brute-force.bro

Berikut merupakan *script* dari *rule ftp_brute-force.bro*:

```
##! FTP brute-forcing detector, triggering when too many rejected usernames or
##! failed passwords have occurred from a single address.

@load base/protocols/ftp
@load base/frameworks/sumstats

@load base/utils/time

module FTP;

export {
        redef enum Notice::Type += {
                ## Indicates a host bruteforcing FTP logins by watching for too
                ## many rejected usernames or failed passwords.
                Bruteforcing
        };

        ## How many rejected usernames or passwords are required before being
        ## considered to be bruteforcing.
        const bruteforce_threshold: double = 3 &redef;

        ## The time period in which the threshold needs to be crossed before
        ## being reset.
        const bruteforce_measurement_interval = 1mins &redef;
}


event bro_init()
        {
        local r1: SumStats::Reducer = [$stream="ftp.failed_auth", $apply=set(SumStats::UNIQUE), $unique_max=double_to_count(bruteforce_threshold+2)];
        SumStats::create([$name="ftp-detect-bruteforcing",
                          $epoch=bruteforce_measurement_interval,
                          $reducers=set(r1),
                          $threshold_val(key: SumStats::Key, result: SumStats::Result) =
                                {
```

```
                                return result["ftp.failed_auth"]$num+0.0;
                                },
                        $threshold=bruteforce_threshold,
                        $threshold_crossed(key: SumStats::Key, result: SumStats::Result) =
                                {
                                local r = result["ftp.failed_auth"];
                                local dur = duration_to_mins_secs(r$end-r$begin);
                                local plural = r$unique>1 ? "s" : "";
                                local message = fmt("%s had %d failed logins on %d FTP server%s in %s", key$host, r$num, r$unique, plural, dur);
                                NOTICE([$note=FTP::Bruteforcing,
                                        $src=key$host,
                                        $msg=message,
                                        $identifier=cat(key$host)]);
                                }]);
        }

event ftp_reply(c: connection, code: count, msg: string, cont_resp: bool)
        {
        local cmd = c$ftp$cmdarg$cmd;
        if ( cmd == "USER" || cmd == "PASS" )
                {
                if ( FTP::parse_ftp_reply_code(code)$x == 5 )
                        SumStats::observe("ftp.failed_auth", [$host=c$id$orig_h], [$str=cat(c$id$resp_h)]);
                }
        }
hook Notice::policy(n: Notice::Info)
        {
                add n$actions[Notice::ACTION_EMAIL];
        }
```

Lampiran 8 Rule FTP Brute-force.bro

Berikut merupakan *script* dari *synflood.bro* :

```
export {
        redef enum Notice::Type += {
        SynFloodStart,    # start of syn-flood against a certain victim
        SynFloodEnd,      # end of syn-flood against a certain victim
        SynFloodStatus,   # report of ongoing syn-flood
        };


# We report a syn-flood if more than SYNFLOOD_THRESHOLD new connections
# have been reported within the last SYNFLOOD_INTERVAL for a certain IP.
# (We sample the conns by one out of SYNFLOOD_SAMPLE_RATE, so the attempt
# counter is an estimated value.). If a victim is identified, we install a
# filter via install_dst_filter and sample the packets targeting it by
# SYNFLOOD_VICTIM_SAMPLE_RATE.
#
# Ongoing syn-floods are reported every SYNFLOOD_REPORT_INTERVAL.

global SYNFLOOD_THRESHOLD = 15000 &redef;
global SYNFLOOD_INTERVAL = 60 secs &redef;
global SYNFLOOD_REPORT_INTERVAL = 1 mins &redef;

# Sample connections by one out of x.
global SYNFLOOD_SAMPLE_RATE = 100 &redef;

# Sample packets to known victims with probability x.
global SYNFLOOD_VICTIM_SAMPLE_RATE = 0.01 &redef;

global conn_attempts: table[addr] of count &default = 0;
global victim_attempts: table[addr,addr] of count
        &default = 0  &read_expire = 5mins;

# We remember up to this many number of sources per victim.
global max_sources = 100;
```

```
global current_victims: table[addr] of set[addr] &read_expire = 60mins;
global accumulated_conn_attempts: table[addr] of count &default = 0;

global sample_count = 0;
global interval_start: time = 0;
}

# Using new_connection() can be quite expensive but connection_attempt() has
# a rather large lag that may lead to detecting flood too late. Additionally,
# it does not cover UDP/ICMP traffic.
event new_connection(c: connection)
        {
        if ( c$id$resp_h in current_victims )
                {
                ++conn_attempts[c$id$resp_h];

                local srcs = current_victims[c$id$resp_h];
                if ( |srcs| < max_sources )
                        add srcs[c$id$orig_h];
                return;
                }

        if ( ++sample_count % SYNFLOOD_SAMPLE_RATE == 0 )
                {
                local ip = c$id$resp_h;

                if ( ++conn_attempts[ip] * SYNFLOOD_SAMPLE_RATE >
                     SYNFLOOD_THRESHOLD )
                        {
                        NOTICE([$note=SynFloodStart, $src=ip,
                                $msg=fmt("Start of syn-flood against %s; sampling packets now", ip)]);

                        add current_victims[ip][c$id$orig_h];

                        # Drop most packets to victim.
                        #install_dst_addr_filter(ip, 0,
                        #                1 - SYNFLOOD_VICTIM_SAMPLE_RATE);
                        # Drop all packets from victim.
                        #install_src_addr_filter(ip, 0, 1.0);
                        }
                }
        }

event check_synflood()
        {
        for ( ip in current_victims )
                {
                accumulated_conn_attempts[ip] =
                        accumulated_conn_attempts[ip] + conn_attempts[ip];

                if ( conn_attempts[ip] * (1 / SYNFLOOD_VICTIM_SAMPLE_RATE) <
                     SYNFLOOD_THRESHOLD )
                        {
                        NOTICE([$note=SynFloodEnd, $src=ip, $n=|current_victims[ip]|,
                                $msg=fmt("end of syn-flood against %s; stopping sampling",
                                     ip)]);

                        delete current_victims[ip];
                        #uninstall_dst_addr_filter(ip);
                        #uninstall_src_addr_filter(ip);
                        }
                }

        clear_table(conn_attempts);
        schedule SYNFLOOD_INTERVAL { check_synflood() };
        }
```

```
event report_synflood()
        {
        for ( ip in current_victims )
                {
                local est_num_conn = accumulated_conn_attempts[ip] *
                                        (1 / SYNFLOOD_VICTIM_SAMPLE_RATE);

                local interv: interval;

                if ( interval_start != 0 )
                        interv = network_time() - interval_start;
                else
                        interv = SYNFLOOD_INTERVAL;

                NOTICE([$note=SynFloodStatus, $src=ip, $n=|current_victims[ip]|,
                           $msg=fmt("syn-flood against %s; estimated %.0f connections in last %s",
                                    ip, est_num_conn, interv)]);
                }

        clear_table(accumulated_conn_attempts);

        schedule SYNFLOOD_REPORT_INTERVAL { report_synflood() };
        interval_start = network_time();
        }

event bro_init()
        {
        schedule SYNFLOOD_INTERVAL { check_synflood() };
        schedule SYNFLOOD_REPORT_INTERVAL { report_synflood() };
        }

hook Notice::policy(n: Notice::Info)
        {
```

```
hook Notice::policy(n: Notice::Info)
        {
                add n$actions[Notice::ACTION_EMAIL];
        }
```