SISTEM NOTIFIKASI GANGGUAN KEAMANAN LOCAL AREA NETWORK (LAN) PADA ADDRESS RESOLUTION PROTOCOL (ARP)

PROYEK TUGAS AKHIR

Diajukan untuk memenuhi salah satu syarat
Mencapai derajat Sarjana S-1 Program Studi Teknik Informatika

Disusun oleh:
Ardika Rommy Sanjaya
5130411060

PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI DAN ELEKTRO
UNIVERSITAS TEKNOLOGI YOGYAKARTA

2017

SISTEM NOTIFIKASI GANGGUAN
KEAMANAN LOCAL AREA NETWORK (LAN)
PADA ADDRESS RESOLUTION PROTOCOL (ARP)

PROYEK TUGAS AKHIR

Disusun oleh:
Ardika Rommy Sanjaya
5130411060

Telah dipertanggung jawabkan di dalam Sidang Proyek Tugas Akhir pada tanggal, ...

(Pelaksanaan Sidang)

Tim Penguji:

_____ (tanda tangan ketua)
Ketua

_____ (tanda tangan anggota)
Anggota

_____ (tanda tangan anggota)
Anggota

Tugas akhir ini telah diterima sebagai salah satu syarat untuk mencapai derajat Sarjana S-1 Program Studi Teknik Informatika.

Yogyakarta ,…….....…….
Ketua Program Studi Teknik Informatika

Dr. Enny Itje Sela, S.Si., M.Kom.

LEMBAR PERNYATAAN

Saya yang bertanda tangan di bawah ini:

N a m a : Ardika Rommy Sanjaya

NPM : ……...

Program Studi : Teknik Informatika

Menyatakan bahwa Proyek Tugas Akhir yang berjudul:

Sistem Notifikasi Gangguan Keamanan Local Area Network (LAN) Pada Address Resolution Protocol (ARP) merupakan karya ilmiah asli saya dan belum pernah dipublikasikan oleh orang lain, kecuali yang tertulis sebagai acuan dalam naskah ini dan disebutkan dalam daftar pustaka. Apabila dikemudian hari, karya saya disinyalir bukan merupakan karya asli saya, maka saya bersedia menerima konsekuensi apa yang diberikan Universitas Teknologi Yogyakarta kepada saya.

Demikian surat pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Yogyakarta
Pada tanggal :

Yang menyatakan

Ardika Rommy Sanjaya

ABSTRAK

Address Resolution Protocol (ARP) adalah protokol yang bertugas untuk menemukan *hardware address (MAC Address)* suatu *host* dengan *Internet Protocol (IP) Address* tertentu di dalam *Local Area Network (LAN)*. *Hardware address* tersebut kemudian disimpan di dalam *ARP cache*. Proses penyimpanan dilakukan tanpa ada pengecekan kesesuain antara *IP* dan *hardware address*. Hal ini menyebabkan *Arp Cache* dapat di-*update* oleh pengguna lain. Dengan begitu data-data milik pengguna dapat dilihat oleh pengguna lain. Oleh karena itu perlu sistem notifikasi yang dapat memberikan saran/pesan bagi pengguna agar dapat mengantisipasi hal tersebut.

Kata kunci: *ARP, LAN,* sistem notifikasi.

ABSTRACT

Address Resulution Protocol is one of critical protocol serving in the OSI model of network architecture. It is responsible for the conversion of network address to physical address at the network layer. ARP protocol is vulnerable so its weakness leads attacks like sniffing, man in the middle attack by poisoning ARP cache (ARP cache poisoning attack). ARP cache poisoning is the most dangerous attack that threats LANs, this attack comes from the way the ARP protocol works. The ARP cache poisoning attack may be launch either denial of service (Dos) attacks or man in the middle attack. By detecting ARP cache poisoning we can minimize the attack. These thesis present the detecting mechanism and notifications.

Keyword: Address Resolution Protocol, Spoofing, Sniffing, Man In The Middle, Spoof Detection.

KATA PENGANTAR

Puji syukur dipanjatkan atas kehadirat Tuhan Yang Maha Esa, karena dengan limpahan karunia-Nya penulis dapat menyelesaikan Proyek Tugas Akhir dengan judul Sistem Notifikasi Gangguan Keamanan Local Area Network (LAN) Pada Address Resolution Protocol (ARP).

Penyusunan Proyek Tugas Akhir diajukan sebagai salah satu syarat untuk memperoleh gelar sarjana pada Program Studi Teknik Informatika Fakultas Teknologi Informasi Universitas Teknologi Yogyakarta.

Proyek Tugas Akhir ini dapat diselesaikan tidak lepas dari segala bantuan, bimbingan, dorongan dan doa dari berbagai pihak, yang pada kesempatan ini penulis ingin menyampaikan ucapan terima kasih kepada:

1. Kepada Dr. Bambang Moertono Setiawan, MM., Akt., CA., selaku Rektor Universitas Teknologi Yogyakarta.
2. Kepada Dr. Erik Iman Heri Ujianto, S.Si., M.Kom., selaku Dekan Fakultas Teknologi Informasi dan Elektro.
3. Kepada Dr. Enny Itje Sela, S.Si., M.Kom., selaku Ketua Program Studi dan Dosen Pembimbing Tugas Akhir.

Akhir kata, penulis menyadari bahwa sepenuhnya akan terbatasnya pengetahuan penyusun, sehingga tidak menutup kemungkinan jika ada kesalahan serta kekurangan dalam penyusunan Proyek Tugas Akhir, untuk itu sumbang saran dari pembaca sangat diharapkan sebagai bahan pelajaran berharga dimasa yang akan datang.

Yogyakarta, ...

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I
PENDAHULUAN

1.1 Latar Belakang

Penggunaan jaringan komputer khususnya *Local Area Network (LAN)* beresiko mengalami gangguan keamanan. Hal ini dapat membahayakan pengguna *LAN* tersebut. Salah satu gangguan keamanan pada *LAN* adalah *spoofing*. *Spoofing* adalah ganggunan yang dapat mengakibatkan informasi pengguna *LAN* dapat dilihat oleh pengguna lain yang tidak berhak. Misalnya terlihatnya *username, password,* foto, vidio, dan lain sebagainya. Informasi tersebut dapat dilihat oleh pengguna yang tidak berhak karena protokol yang bertanggung jawab untuk menterjemahkan *Internet Protocol Address (IP Address)* menjadi alamat fisik *(physical address/MAC Address)* menyimpan didalam *ARP Cache* yang dapat diubah oleh pengguna lain. Pada kondisi normal (tidak ada gangguan) *IP Address* dan *MAC Address* yang terdapat di dalam *ARP Cache* akan sesuai pada saat dilakukan pengecekan dan *LAN* dianggap aman. Namun, jika ada ketidak sesuaian pada *IP Address* dan *MAC Address* maka *LAN* akan dianggap tidak aman (mendapat gangguan).

Gangguan *LAN* ini terjadi karena perubahan *MAC Address* yang bertipe dinamis pada *ARP Cache*. Cara untuk merubah *MAC Address* tersebut adalah dengan mengirimkan *ARP Reply* yang berisi *IPAddress* dan *MAC Address* yang telah diubah kepada pengguna *LAN*. Perubahan tersebut menyebabkan gangguan terhadap pertukaran data pada *LAN* sehingga data-data pengguna, seperti *username, password,* dan sebagainya dapat dilihat oleh pengguna lain.

Permasalahan ini jika tidak diatasi dapat membahayakan data pengguna *LAN*. Oleh karena itu perlu dikembangkan sistem yang dapat memberikan notifikasi berupa langkah-langkah pencegahan agar data tersebut tidak dapat dilihat oleh pengguna lain. Sistem ini melakukan pengecekan setiap *ARP Reply* yang diterima. Ketika *ARP Reply* tersebut meng-*update ARP Cache* yang mengakibatkan perubahan *MAC Address* dan membuat ketidak sesuain *IP*

Address dan *MAC Address* maka *LAN* dianggap tidak aman. Oleh karena *LAN* tidak aman sistem akan memberikan notifikasi kepada pengguna agar pengguna mengikuti langkah-langkah yang diberikan.

1.2 Rumusan Masalah

Berdasarkan latar belakang diatas dapat dirumuskan beberapa masalah dalam peneliatan ini, yaitu:

a. Bagaimana merancang sistem notifikasi gangguan keamanan jaringan pada *Address Resolution Protocol (ARP)*?
b. Apakah sistem notifikasi yang dibuat dapat memberikan pemberitahuan kepada pengguna tentang adanya gangguan keamanan pada jaringan yang digunakan?
c. Apakah sistem notifikasi yang dikembangkan mampu memberikan saran bagi pengguna jaringan terkhusus pada *LAN*.

1.3 Batasan Masalah

Mengingat dengan banyaknya perkembangan masalah yang bisa ditemukan pada penelitian ini, maka perlu adanya batasan-batasan masalah yang jelas mengenai apa yang dibuat dan diselesaikan. Adapun batasan-batasan masalah pada penelitian ini, sebagai beribut:

a. Sistem dapat berjalan pada sistem operasi Windows dan Linux.
b. Sistem dapat memberikan notifikasi pada pengguna yang menggunakan *LAN* terhadap gangguan dari pengguna lain pada jaringan tersebut.
c. Sistem dapat memberikan saran ketika pengguna mendapatkan gangguan keamanan jaringan pada *Address Resolution Protocol (ARP).*

1.4 Tujuan Penelitian

Adapun tujuan dari penelitian ini adalah untuk mengembangkan sistem yang dapat melakukan deteksi serangan pada *Address Resolution Protocol (ARP)* dan dapat menghasilkan *output* berupa notifikasi. Dengan adanya notifikasi ini pengguna jaringan akan dapat melakukan tindakan pencegahan akan sesuatu yang

dapat merugikan dirinya dan memberikan rasa aman dalam penggunaan jaringan *Local Area Network (LAN).*

1.5 Manfaat Penelitian

Diharapkan penelitian ini dapat memberikan manfaat baik bagi pengguna, penulis, maupun peneliti lain.

a. Manfaat Bagi Pengguna: Sistem notifikasi dapat mengurangi tingkat penyalahgunaan jaringan *Local Area Network (LAN)* baik untuk pencurian *password*, manipulasi paket jaringan dan lain sebagainya.
b. Manfaat Bagi Penulis: Menambah wawasan penulis khususnya mengenai keamanan jaringan komputer.
c. Manfaat Bagi Peneliti Lain: Penelitian ini diharapkan dapat memberikan manfaat secara teoritis, sekurang-kurangnya dapat berguna sebagai sumbangan pemikiran dan referensi sehingga dapat memperkaya wawasan peneliti lain khususnye mengenai keamanan *LAN*.

1.6 Sistematika Penulisan

Sistematika penulisan yang digunakan dalam penelitian ini adalah sebagai berikut:

BAB I. Pendahuluan.

Pada bab ini menjelaskan tentang pentingnya kemanan jaringan *Local Area Network (LAN)* terkhusus pada penggunaan *ARP* yang dapat membahayakan pengguna serta pentingnya sebuah sistem notifikasi sebagai acuan untuk melakukan pencegahan jika terjadi aktivitas pada jaringan yang membahayakan. Selain itu disertakan rumusan dari permasalahan keamanan jaringan pada penelitian ini, batasan permasalahan yang diteliti, tujuan dari penelitian, dan manfaat yang didapat dari sistem notifikasi ini.

BAB II. Kajian Pustaka dan Teori.

Memaparkan hasil dari penelitian yang telah dilakukan oleh peneliti-peneliti sebelumnya dengan tujuan untuk mencari solusi yang dapat

memaksimalkan karja sistem notifikasi. Selain itu juga disertakan teori-teori dasar jaringan yang dapat digunakan sebagai acuan dalam pengembangan sistem notifikasi ini.

BAB III. Metode Penelitian.

Menjelaskan tentang protokol dan teknologi yang digunakan dalam pengembangan sistem notifikasi ini. Selain itu dijelaskan juga metode-metode yang digunakan oleh penulis dalam menyelesaikan permasalahan terkhusus pada *Address Resolution Protocol (ARP)*. Metode tersebut diantaranya metode pengumpulan data, metode analisis dan perancangan, pembuatan program, implementasi, dan pengujian sistem.

BAB IV. Analisis dan Perancangan.

Pada bab ini akan dijelaskan tentang fungsi dan bagaimana cara kerja dari *Address Resolution Protocol (ARP)* beserta kelemahannya. Selain itu dijelaskan juga kebutuhan fungsional maupun non fungsional sistem agar sistem dapat melakukan tugasnya. Untuk perancangan sistem notifikasi, algoritma akan ditampilkan dalam bentuk *flow chart* beserta rancangan *Graphical User Interface (GUI)* dari sistem.

BAB V. Implementasi sistem.

Menjelaskan bagaimana sistem notifikasi ini aplikasikan pada *LAN* serta pengujian dari sistem ketika melakukan deteksi terhadap aktivitas jaringan yang membahayakan. Selain itu dijelaskan juga bagaimana cara menggunakan sistem, konfigurasi sistem, dan tampilan dari *Graphical User Interface (GUI)* dari sistem notifikasi. Pada bab ini juga disertakan potongan kode sumber yang digunakan untuk melakukan proses pencarian kartu jaringan yang terkoneksi pada jaringan, proses pengiriman dan *capture* paket untuk mendapatkan *MAC Address* dari gateway, serta proses deteksi beserta modul yang digunakan.

BAB VI. Penutup.

Pada bab ini akan sampaikan kesimpulan dari hasil penelitian serta pengujian dari sistem notifikasi terhadap keamanan jaringan. Selain itu dikarenakan sistem ini memiliki kekurangan maka diberikan juga saran yang mana dapat dijadikan sebuah penelitian lagi dengan tujuan agar dapat memberikan kontribusi pada keamanan jaringan terkhusus *Local Area Network (LAN)*.

Daftar Pustaka

Lampiran

BAB II
KAJIAN PUSTAKA DAN TEORI

2.1 Kajian Hasil Penelitian

Penelitian oleh Vinay dan Rahman (2015), menggunakan teknik deteksi aktif dengan cara meng-*capture* paket *ARP* dan menyimpannya paket pertama ke dalam *database*. Ketika hasil *capture* paket *ARP* berikutnya telah ada di dalam *database* dan sesuai maka paket tersebut dinyatakan aman, namun jika yang sama hanya *IP Address* yang sama maka *IP Address* tersebut akan digunakan untuk melakukan pengiriman paket *Internet Control Messaege Protocol (ICMP).* Jika penyerang mengizinkan *IP Packet Routing* maka paket yang tadi dikirimkan ke penyerang akan diteruskan kembali oleh penyerang sesuai dengan *IP Address* tujuan. Dari paket tersebut dapat dilakukan pencocokan paket pada *layer* 2 dengan paket *ARP* hasil *capture* yang sebelumnya untuk memastikan apakah paket *ARP* tersebut aman atau tidak. Namun hal ini dapat diatasi oleh penyerang dengan membuat *firewall* untuk meblokir setiap paket *ICMP* yang masuk. Oleh karena itu peneliti tidak menggunakan paket *ICMP* sebuagai parameter untuk deteksi namun berbagai macam parameter lainnya yang masih mungkin dapat digunakan.

Kaur (2013) penah melakukan penelitian untuk mendeteksi dan mengatasi serangan *ARP Spoofing (ARP Cache Poisoning)* dengan mendeteksi paket-paket yang mencurigakan dan ketika telah dipastikan ada yang melakukan serangan segera diambil tindakan dengan mengirimkan paket *ARP Request* ke *gateway* dengan tujuan untuk memperbaharui *ARP Cache*. Selain itu digunakan juga *ICMP* untuk melakukan pengecekan apakah penyerang mengizinkan *IP Packet Routing* untuk meneruskan paket *IPv4* pada tujuan. Berdasarkan penelitian tersebut peneliti akan menambakan fitur untuk melakukan penyimpanan hasil *capture* paket pada format yang umum digunakan seperti pcap dan pcapng agar lebih mudah untuk dianalisis oleh peneliti lain.

Srinath dkk (2015) telah melakukan penelitian dengan menggunakan tiga model untuk mengatasi serangan *ARP Spoofing* yaitu model perspektif komputer

(*host*), perspektif *server*, dan otentikasi. Dimodel pertama setiap komputer mengirimkan informasi yang didapat setelah terhubung ke jaringan melalui *DHCP* ke *server* dan tugas *server* adalah menyimpan informasi tersebut ke *database* sekaligus melakukan pengecekan informasi. Informasi yang disimpan di *database* dapat ditampilkan dengan menggunakan diagram agar mempermudah pembacaan. Sistem yang peneliti buat hanya digunakan di sisi *client* dengan alasan kebiasaan pengguna jaringan seperti *wifi* yang selalu berpindah-pindah (tidak hanya menggunakan satu jaringan).

Tabel 2.1. Perbandingan Tinjauan Pustaka

No	Judul	Penulis	Metode	Hasil/Kesimpulan
1	*ARP Spoof Detection System using ICMP*	Vinay K. R. dan T. R. Mahibur Rahman	*ICMP Modul*	Teknik ini juga dapat mendeteksi *IP* dan *MAC Address* yang asli *(Correct Address)* selain
2	*Detection and Prevention of ARP Cache*	Inderjeet Kaur	*ARP dan ICMP*	Metode ini cukup efisian untuk mendeteksi dan mengatasi *ARP Cache*
3	*Detection and Prevention of ARP Spoofing using Centralized Server*	D. Srinath, S. Panimalar, A. Jerrin Simla dan J. Deepa	*Centralized Server*	Metode ini cukup baik digunakan untuk mengatasi *ARP Spoofing* selain itu dapat pula digunakan untuk mengatasi *IP Spoofing*.
4	Sistem Notifikasi Gangguan Keamanan Jaringan Pada Address Resolution Protocol (ARP)	Sanjaya, A.R.	*ARP* dan *TCP module*	

Seperti terlihat pada tabel 2.1. perbedaan dari ketiga referensi dengan judul yang diangkat oleh penulis terletak pada metode yang digunakan, masing-masing metode memiliki keunggulannya masing-masing. Peneliti akan menggunakan beberapa keunggulan dari masing-masing referensi dan menambahkan beberapa metode untuk meingkatkan kemampuan dari sistem pendeteksi serangan pada penelitian ini.

2.2 Dasar Teori

2.2.1 *Intrusion Detection System (IDS)*

Intrusion Detection System merupakan sebuah sistem yang dapat digunakan untuk melakukan deteksi terhadap aktivitas yang mencurigakan dalam sebuah sistem atau jaringan yang dapat menggangu konfidensialitas, integritas dan ketersediaan data. *IDS* dapat melakukan inspeksi terhadap lalu lintas *inbound* dan *outbound* dalam sebuah sistem atau jaringan, melakukan analisis dan mencari bukti dari percobaan intrusi.

2.2.2 *Protocol*

Protocol (protokol) pada jaringan komputer merupakan sebuah prosedur atau aturan yang harus disetujui secara bersama oleh perangkat yang akan berkomunikasi. Banyaknya protokol yang berbeda pada jaringan mengakibatkan sulitnya komunikasi antar perangkat yang terkoneksi melalui jaringan.

Salah satu model arsitektur yang banyak digunakan adalah *OSI* (*Open System Interconnection*) yang berupaya membentuk standar umum jaringan komputer untuk menunjang interoperatibilitas antar pemasok (*vendor*) dari yang berbeda. OSI memiliki 7 lapisan/*layer* yang setiap lapisan memiliki fungsinya masing-masing. Menurut Sugeng, W dan Putri, T.D fungsi dari masing-masing lapisan/*layer* yang terdapat pada *OSI* sebagai berikut:

1. Lapisan Fisik (*Physical Layer*), berfungsi dalam mengiriman *raw* bit ke kanal komunikasi. Masalah-masalah yang harus diperhatikan adalah masalah desain (Jika dikirim bit 1 harus diartikan bit 1 disisi penerima), masalah debain ini ditemukan ada hubungannya dengan mekanika, kelistrikan, prosedur *interface*,

dan medium transmisi fisik yang berada di lapisan fisik.

2. Lapisan Jalur Data (*Data Link Layer*), tugas utamanya sebagai fasilitas transmisi *raw* data dan mentransfirmasikan data tersebut ke saluran yang bebas dari kesalahan transmisi. Dimungkinnya melalukan pemecahan data input menjadi sejumlah data *frame* (biasanya jumlahnya ratusan atau ribuan byte). Selanjutnya *frame* tersebut dikirim secara perurutan, dan memproses *acknowledgment frame* yang dikirim kembali oleh penerima. Penambahan bit-bit khusus diawal dan diakhir data guna pengenalan *frame* merupakan bagian pekerjaannya. Jika terjadi *noise* dan *frame* rusak *frame* dikirim ulang, tapi akibatnya akan terjadi duplikasi *frame* jika *acknowledgment frame* hilang.
3. Lapisan Jaringan (*Network Layer*), berfungsi sebagai pengendalian operasi *subnet*. Masalah desain yang penting adalah menentukan *route* pengiriman *packet* dari sumber ke tujuannya. Desain *route* dapat berupa statik atau dinamik. Masalah pengendalian kemacetan (*bottlenect*) merupakan tugasnya. Pada jaringan *broadcast*, masalah penentuan *route* hal yang sederhana, lapisan jaringan bisa tidak ada atau tidak diperlukan.
4. Lapisan Transport (*Transport Layer*), fungsi dasarnya adalah menerima data dari Lapisan Sesi, bila perlu memecah data menjadi bagian-bagian yang lebih kecil, meneruskan potongan ke lapisan jaringan dan menjamin seluruh potongan data sampai dengan benar disisi lainnya. Harus dilaksanakan secara efisien. Tujuan lainnya adalah melindungi seluruh lapisan diatasnya dari perubahan teknologi perangkat keras yang mungkin timbul. Bila diperlukan *throughput* yang tinggi, maka lapisa *transport* hubungan jaringan yang banyak, tetapi dapat pula menggabungkan beberapa hubungan *transport* ke hubungan jaringan yang sama. Penentuan jenis layanan (yang populer adalah saluran *error-free point tot point*) merupakan tugasnya pula. Merupakan *layer end-to-end* sejati dari sumber ke tujuan. Banyak *host* diprogram dengan *multiprogrammed* (banyak hubungan yang masuk dan meninggalkan *host* untuk menyatakan pesan mana). TH adalah tempat informasi tersebut ditempatkan. Pengendalian aliran (*Flow Control*) adalah merupakan tugasnya agar tidak membanjiri *host* yang lambat.

5. Lapisan Sesi (*Session Layer*), mengizinkan para pengguna untuk menetapkan *session* di antara mereka. Sebuah *session* digunakan untuk memungkinkan seseorang pengguna lelakukan *log* ke dalam suatu *remote time sharing system* atau memindahkan suatu *file* dari satu mesin ke mesin yang lain. Jadi tugasnya adalah pengendalian dialog. Funsi lainnya dalah manajemen *token (token management),* sinkronisasi (*synchronization*), penyisipan *checkpoint* diperlukan jika akan mengulangi pengiriman akibat terjadinya *crash* sehingga tidak perlu seluruh data diulang pengirimannya.
6. Lapisan Presentasi (*Presentation Layer*),melakukan fungsi tertentu yang sering diminta untuk menjamin penemuan sebuah penyelesaian umum bagi masalah tertentu. Lapisan Presentasi tidak mengizinkan pengguna untuk menyelesaikan sendiri suatu masalah. Lapisan Presentasi memperhatikan *syntax* dan semantik informasi yang dikirimkan. Contoh layanannya adalah pengodean data (*data encoding*).
7. Lapisan Aplikasi (*Appilcation Layer*), tugasnya melayani *remote* terminal. Lapisan aplikasi terdiri dari bermacam-macam protocol yang bisa digunakan. Diperlukan adanya terminal virtual jaringan (*network virtual terminal*) sebelum suatu editor *remote* digunakan. Fungsi lainnya adalah pemindahan … (biasanya satu sistem ke sistem lain mempunyai konvensi yang berbeda). Tugasnya seperti: E-mail, Telnet, FTP, WWW dan lain sebagainya.

2.2.3 *Address Resolution Protocol (ARP)*

Menurut Sugeng, W dan Putri, T.D (2017), *Address Resolution Protocol (ARP)* adalah protokol yang bertugas untuk menemukan *hardware address* suatu *host* dengan *IP Address* tertentu. Berikut format dari protokol ini:

Tabel 2.2. Format *ARP*

Octet Offset	0	1
0	Hardware type	
2	Protocol type	
4	Hardware Address Length	Protocol Address Length
6	Operation	

Tabel 2.2. Penggalan dari tabel 2.2. pada halaman 10.

8	Sender Hardware Address
10	
12	
14	Sender Protocol Address
16	
18	Target Hardware Address
20	
22	
24	Target Protocol Address
26	

2.2.4 *Ethernet*

Menurut Sugeng, W dan Putri, T.D (2017), pada awalnya Ethernet didesain untuk dijalankan di atas kabel koaksial pada kecepatan maksumum 10 Mbps. Sekarang Ethernet berjalan pada kabel koaksial *thin-wide* (10 base 2) dan *unshielded twisted-pair (UTP) telephone wiring* (10 base 3). *Device* pada *network-PC, workstation, printer, server,* dll secara fisik terhubung ke kabel tunggal yang dikenal sebagai *bus.*

Pada perkembangan berikutnya, muncul teknologi *Switch Ethernet,* untuk menghindari *problem* tabrakan paket. Sebuah *Switch Ethernet* menggantikan pengabelan *hub.* Berikutnya ada *Fast Ethernet,* yang membesarkan *bandwith* LAN dari 10 Mbps menjadi 100 Mbps. Ia menggunakan 2 standar: Gigabit 100base-I (IEEE 802.3u) dan Gigabit 100VG-AnyLAN (IEEE 803.12).

2.2.5 *Ethernet II*

Ethernet II adalah sebuah standar enkapsulasi paket data jaringan berbasis teknologi *Ethernet* yang digunakan oleh protokol *TCP/IP*. Standar ini dikembangkan oleh Digital Equipment Corporation (DEC), Intel Corporation, dan Xerox sebelum akhirnya diserahkan kepada komite IEEE 802 untuk menjadi standar IEEE 802.3. *Ethernet II* juga disebut sebagai *Ethernet II frame format*

atau *DIX frame format* (mengingat pihak-pihak yang mengembangkannya adalah DEC, Intel dan Xerox).

Tabel 2.3. Ethernet II Frame Format

MAC Destination	MAC Source	Ethertype	Payload	FCS
6 octets	6 octets	2 octets	46-1500 octets	4 octet

2.2.6 *IP Address*

Menurut Sugeng, W dan Putri, T.D (2017), *IP (Internet Protocol) Address* atau *IP Address* yang bahasa awamnya bisa disebut dengan kode pengenal komputer pada jaringan merupaan komponen vital pada internet, karena tanpa *IP Address* sesorang tidak akan dapat terhubung ke *internet.* Setiap komputer yang terhubung ke *internet* setidaknya harus memiliki satu buah *IP Address* pada setiap perangkat yang terhubung ke *internet* dan *IP Address* itu sendiri harus unik karena tidak boleh ada komputer/*server/*perangkat jaringan lainnya yang menggunakan *IP Address* yang sama di *Internet.*

2.2.7 *Media Access Control Address (MAC Address)*

Media Access Control Address (MAC Address) adalah sebuah alamat jaringan yang diimplementasikan pada lapisan *datalink* dalam tujuh lapisan model *OSI*, yang merepresentasikan sebuah node tertentu dalam jaringan. Dalam sebuah jaringan berbasis *Ethernet*, *MAC Address* merupakan alamat yang unik yang memiliki panjang 48-bit (6 byte) yang mengidentifikasikan sebuah komputer, *interface* dalam sebuah *router*, atau node lainnya dalam jaringan. *MAC Address* juga sering disebut sebagai *Ethernet address, physical address,* atau *hardware address*.

Dalam sebuah komputer, *MAC Address* ditetapkan ke sebuah kartu jaringan *(network interface card/NIC)* yang digunakan untuk menghubungkan komputer yang bersangkutan ke jaringan. *MAC Address* umumnya tidak dapat diubah karena telah dimasukkan ke dalam ROM. Beberapa kartu jaringan menyediakan utilitas yang mengizinkan pengguna untuk mengubah *MAC Address*, meski hal ini kurang disarankan. Jika dalam sebuah jaringan terdapat dua

kartu jaringan yang memiliki *MAC Address* yang sama, maka akan terjadi konflik alamat dan komputer pun tidak dapat saling berkomunikasi antara satu dengan lainnya. Beberapa kartu jaringan, seperti halnya kartu Token Ring mengharuskan pengguna untuk mengatur *MAC Address* (tidak dimasukkan ke dalam ROM), sebelum dapat digunakan.

MAC Address memang harus unik, dan untuk itulah, Institute of Electrical and Electronics Engineers (IEEE) mengalokasikan blok-blok dalam *MAC Address*. 24 bit pertama dari *MAC Address* merepresentasikan siapa pembuat kartu tersebut, dan 24 bit sisanya merepresentasikan nomor kartu tersebut. Setiap kelompok 24 bit tersebut dapat direpresentasikan dengan menggunakan enam digit bilangan heksadesimal, sehingga menjadikan total 12 digit bilangan heksadesimal yang merepresentasikan keseluruhan *MAC Address*.

Agar antara komputer dapat saling berkomunikasi satu dengan lainnya, *frame-frame* jaringan harus diberi alamat dengan menggunakan alamat *Layer-2* atau *MAC Address*. Tetapi, untuk menyederhanakan komunikasi jaringan, digunakanlah alamat *Layer-3* yang merupakan *IP Address* yang digunakan oleh jaringan *TCP/IP*. Protokol dalam *TCP/IP* yang disebut sebagai *Address Resolution Protocol (ARP)* dapat menerjemahkan alamat *Layer-3* menjadi alamat *Layer-2*, sehingga komputer pun dapat saling berkomunikasi.

2.2.8 *Internet Protocol Version 4 (IPv4)*

Menurut Sugeng, W dan Putri, T.D (2017), *Internet Protocol Version 4 (IPv4) Address* pada awalnya adalah sederetan bilangan biner sepanjang 32 bit yang dipakai untuk mengidentifikasikan *host* pada jaringan. *IP Address* ini diberikan secara unik pada masing-masing komputer/*host* yang terhubung ke *internet.* Prinsip kerjanya adalah paket yang membawa data dimuati *IP Address* dari komputer pengirim data kepada *IP Address* pada komputer yang akan dituju, kemudian data tersebut dikirim ke jaringan. Paket ini kemudian dikirim dari *router* ke *router* dengan berpedoman pada *IP Address* tersebut menuju ke komputer yang dituju. Seluruh komputer/*host* yang tersambung ke iinternet,

dibedakan hanya berdasarkan *IP Address* untuk setiap komputer yang terhubung ke jaringan *internet.*

IPv4 terdiri dari 14 *field*, namun satu *field* terakhir hanya bersifat *optional*. Berikut format dari paket *IPv4*.

Tabel 2.4. Format *IPv4*

<table>
<tr><td>Offsets</td><td>Octet</td><td colspan="4">0</td><td colspan="4">1</td><td colspan="4">2</td><td colspan="4">3</td></tr>
<tr><td>Octet</td><td>Bit</td><td>0</td><td>2</td><td>4</td><td>6</td><td>8</td><td>10</td><td>12</td><td>14</td><td>16</td><td>18</td><td>20</td><td>22</td><td>24</td><td>26</td><td>28</td><td>30</td></tr>
<tr><td>0</td><td>0</td><td colspan="2">Version</td><td colspan="2">IHL</td><td colspan="3">DSCP</td><td>E</td><td colspan="8">Total Length</td></tr>
<tr><td>4</td><td>32</td><td colspan="10">Identification</td><td>Flags</td><td colspan="5">Fragment Offset</td></tr>
<tr><td>8</td><td>64</td><td colspan="5">Time to Live</td><td colspan="5">Protocol</td><td colspan="6">Header Checksum</td></tr>
<tr><td>12</td><td>96</td><td colspan="16">Source IP Address</td></tr>
<tr><td>16</td><td>128</td><td colspan="16">Destinatin IP Address</td></tr>
<tr><td>20</td><td>160</td><td colspan="16" rowspan="4">Options (if IHL > 5)</td></tr>
<tr><td>24</td><td>192</td></tr>
<tr><td>28</td><td>224</td></tr>
<tr><td>32</td><td>256</td></tr>
</table>

2.2.9 *Transmission Control Protocol (TCP)*

Transmission Control Protocol (TCP) merupakan protokol yang terletak pada *transport layer*. Protokol ini menyediakan layanan yang dikenal sebagain *connection oriented* yang berarti sebelum melakukan pertukaran data dua *host* yang menggunakan *TCP* harus melakukan pembentukan hubungan *(handshake)* terlebih dahulu. Berikut karakteristik *TCP*:

- Berorientasi sambungan (*connection-oriented*): Sebelum data dapat ditransmisikan antara dua *host*, dua proses yang berjalan pada lapisan aplikasi harus melakukan negosiasi untuk membuat sesi koneksi terlebih dahulu. Koneksi *TCP* ditutup dengan menggunakan proses terminasi koneksi *TCP (TCP connection termination).*

- *Full-duplex*: Untuk setiap *host TCP*, koneksi yang terjadi antara dua *host* terdiri atas dua buah jalur, yakni jalur keluar dan jalur masuk. Dengan menggunakan teknologi lapisan yang lebih rendah yang mendukung *full-duplex*, maka data pun dapat secara simultan diterima dan dikirim. *TCP Header* berisi nomor urut *(TCP sequence number)* dari data yang ditransmisikan dan sebuah *acknowledgment* dari data yang masuk.
- Dapat diandalkan (*reliable*): Data yang dikirimkan ke sebuah koneksi *TCP* akan diurutkan dengan sebuah nomor urut paket dan akan mengharapkan paket *positive acknowledgment* dari penerima. Jika tidak ada paket *Acknowledgment* dari penerima, maka segmen *TCP* (*protocol data unit* dalam protokol *TCP*) akan ditransmisikan ulang. Pada pihak penerima, segmen-segmen duplikat akan diabaikan dan segmen-segmen yang datang tidak sesuai dengan urutannya akan diletakkan di belakang untuk mengurutkan segmen-segmen *TCP*. Untuk menjamin integritas setiap segmen *TCP*, *TCP* mengimplementasikan penghitungan *TCP Checksum*.
- *Byte stream*: *TCP* melihat data yang dikirimkan dan diterima melalui dua jalur masuk dan jalur keluar *TCP* sebagai sebuah *byte stream* yang berdekatan (kontigu). Nomor urut *TCP* dan nomor *acknowlegment* dalam setiap *TCP Header* didefinisikan juga dalam bentuk *byte*. Meski demikian, *TCP* tidak mengetahui batasan pesan-pesan di dalam *byte stream TCP* tersebut. Untuk melakukannya, hal ini diserahkan kepada protokol lapisan aplikasi (dalam *DARPA Reference Model*), yang harus menerjemahkan *byte stream TCP* ke dalam "bahasa" yang ia pahami.
- Memiliki layanan *flow control*: Untuk mencegah data terlalu banyak dikirimkan pada satu waktu, yang akhirnya membuat "macet" jaringan *internetwork IP*, *TCP* mengimplementasikan layanan *flow control* yang dimiliki oleh pihak pengirim yang secara terus menerus memantau dan membatasi jumlah data yang dikirimkan pada satu waktu. Untuk mencegah pihak penerima untuk memperoleh data yang tidak dapat disangganya (*buffer*), *TCP* juga mengimplementasikan *flow control* dalam

pihak penerima, yang mengindikasikan jumlah *buffer* yang masih tersedia dalam pihak penerima.

- Melakukan segmentasi terhadap data yang datang dari lapisan aplikasi (dalam *DARPA Reference Model*).
- Mengirimkan paket secara "*one-to-one*": hal ini karena memang *TCP* harus membuat sebuah sirkuit logis antara dua buah protokol lapisan aplikasi agar saling dapat berkomunikasi. *TCP* tidak menyediakan layanan pengiriman data secara *one-to-many*.

Proses pembuatan koneksi *TCP* disebut juga dengan *"Three-way Handshake"*. Tujuan metode ini adalah agar dapat melakukan sinkronisasi terhadap nomor urut dan nomor *acknowledgement* yang dikirimkan oleh kedua pihak dan saling bertukar ukuran *TCP Window*. Prosesnya dapat digambarkan sebagai berikut:

- *Host* pertama (yang ingin membuat koneksi) akan mengirimkan sebuah segmen *TCP* dengan *flag SYN* diaktifkan kepada *host* kedua (yang hendak diajak untuk berkomunikasi).
- *Host* kedua akan meresponsnya dengan mengirimkan segmen dengan *acknowledgment* dan juga *SYN* kepada *host* pertama.
- *Host* pertama selanjutnya akan mulai saling bertukar data dengan host kedua.

TCP menggunakan proses jabat tangan (*handshake*) yang sama untuk mengakhiri koneksi yang dibuat. Hal ini menjamin dua *host* yang sedang terkoneksi tersebut telah menyelesaikan proses transmisi data dan semua data yang ditransmisikan telah diterima dengan baik. Itulah sebabnya, mengapa *TCP* disebut dengan koneksi yang *reliable*.

Penjelasan *field-field* pada *TCP* dapat dilihat pada lampiran. Sedangkan penjelasan *TCP flags* pada *TCP* dapat dilihat pada tabel berikut.

Tabel 2.5. Penjelasan *TCP Flags*

Nama flag	**Keterangan**
URG	Mengindikasikan bahwa beberapa bagian dari segmen *TCP* mengandung data yang sangat penting, dan *field Urgent Pointer* dalam *header TCP* harus digunakan untuk menentukan lokasi di mana data penting tersebut berada dalam segmen.
ACK	Mengindikasikan *field Acknowledgment* mengandung oktet selanjutnya yang diharapkan dalam koneksi. *Flag* ini selalu diset, kecuali pada segmen pertama pada pembuatan sesi koneksi *TCP*.
PSH	Mengindikasikan bahwa isi dari *TCP Receive buffer* harus diserahkan kepada protokol lapisan aplikasi. Data dalam *receive buffer* harus berisi sebuah blok data yang berurutan (kontigu), dilihat dari ujung paling kiri dari *buffer*. Dengan kata lain, sebuah segmen yang memiliki *flag PSH* diset ke nilai 1, tidak bolah ada satu byte pun data yang hilang dari aliran byte segmen tersebut; data tidak dapat diberikan kepada protokol lapisan aplikasi hingga segmen yang hilang tersebut datang. Normalnya, *TCP Receive buffer* akan dikosongkan (dengan kata lain, isi dari *buffer* akan diteruskan kepada protokol lapisan aplikasi) ketika *buffer* tersebut berisi data yang kontigu atau ketika dalam "proses perawatan". *Flag PSH* ini dapat mengubah hal seperti itu, dan membuat akan *TCP* segera mengosongkan *TCP Receive buffer*. *Flag PSH* umumnya digunakan dalam protokol lapisan aplikasi yang bersifat interaktif, seperti halnya Telnet, karena setiap penekanan tombol dalam sesi terminal virtual akan dikirimkan dengan sebuah *flag PSH* diset ke nilai 1. Contoh dari penggunaan lainnya dari *flag* ini adalah pada segmen terakhir dari berkas yang *ditransfer* dengan menggunakan protokol *File Transfer Protocol (FTP)*. Segmen yang dikirimkan dengan *flag PSH* aktif tidak harus segera di-*acknowledge* oleh penerima.

Tabel 2.5. Penggalan dari tabel 2.5. pada halaman 20.

RST	Mengindikasikan bahwa koneksi yang dibuat akan digagalkan. Untuk sebuah koneksi *TCP* yang sedang berjalan (aktif), sebuah segmen dengan *flag RST* diset ke nilai 1 akan dikirimkan sebagai *respons* terhadap sebuah segmen *TCP* yang diterima yang ternyata segmen tersebut bukan yang diminta, sehingga koneksi pun menjadi gagal. Pengiriman segmen dengan *flag RST* diset ke nilai 1 untuk sebuah koneksi aktif akan menutup koneksi secara paksa, sehingga data yang disimpan dalam *buffer* akan dibuang (dihilangkan). Untuk sebuah koneksi *TCP* yang sedang dibuat, segmen dengan *flag RST* aktif akan dikirimkan sebagai *respons* terhadap *request* pembuatan koneksi untuk mencegah percobaan pembuatan koneksi.
SYN	Mengindikasikan bahwa segmen *TCP* yang bersangkutan mengandung *Initial Sequence Number (ISN)*. Selama proses pembuatan sesi koneksi *TCP*, *TCP* akan mengirimkan sebuah segmen dengan *flag SYN* diset ke nilai 1. Setiap *host TCP* lainnya akan memberikan jawaban (*acknowledgment*) dari segmen dengan *flag SYN* tersebut dengan menganggap bahwa segmen tersebut merupakan sekumpulan *byte* dari data. *Field Acknowledgment Number* dari sebuah segmen *SYN* diatur ke nilai *ISN* + 1.
FIN	Menandakan bahwa pengirim segmen *TCP* telah selesai dalam mengirimkan data dalam sebuah koneksi *TCP*. Ketika sebuah koneksi *TCP* akhirnya dihentikan (akibat sudah tidak ada data yang dikirimkan lagi), setiap *host TCP* akan mengirimkan sebuah segmen *TCP* dengan *flag FIN* diset ke nilai 1. Sebuah *host TCP* tidak akan mengirimkan segmen dengan *flag FIN* hingga semua data yang dikirimkannya telah diterima dengan baik (menerima paket *acknowledgment*) oleh penerima. Setiap *host* akan menganggap sebuah segmen *TCP* dengan *flag FIN* sebagai sekumpulan *byte* dari data. Ketika dua *host TCP* telah mengirimkan segmen *TCP* dengan *flag FIN* dan menerima *acknowledgment* dari segmen tersebut, maka koneksi *TCP* pun akan dihentikan.

2.2.10 *Sniffing* dan *Spoofing*

Sniffing adalah proses penyadapan paket pada jaringan dengan menggunakan sebuah aplikasi yang biasa disebut *Network Analyzer*. Aplikasi ini menangkap tiap-tiap paket dan dapat juga menguraikan paket tersebut berdasarkan *RFC (Request of Comments)*. Sedangkan *spoofing* merupakan proses pemalsuan paket-paket jaringan yang dapat mendukung proses *sniffing*.

Sniffing sendiri dapat dikategorikan menjadi 2, yaitu aktif dan pasif. *Sniffing* pasif merupakan proses analisa paket jaringan tanpa melakukan perubahan atau pembuatan paket tertentu yang kemudian dikirimkan melelui jaringan. Sebaliknya *sniffing* aktif merupakan proses *sniffing* yang pada kondisi tertentu dapat melakukan perubahan ataupun pembuatan paket yang kemudian dikirimkan melalui jaringan.

2.2.11 *Promiscuous Mode*

Promiscuous mode atau *promisc mode* merupakan konfigurasi pada *Network Interface Card (NIC)* yang dapat menghambat atau meneruskan setiap paket yang melewatinya. Ketika *NIC* berada pada *promiscuous mode* maka setiap paket yang melewatinya (termasuk paket yang tidak ditujukan kepadanya) akan diteruskan ke *CPU* dan diproses.

2.2.12 *Maximum Transmission Unit (MTU)*

Maximum Transmission Unit (MTU) dalam jaringan komputer merupakan maksimum dari ukuran paket yang dapat ditransimikan oleh media jaringan. Ukuran dari *MTU* berfariasi tergantung pada media transmisi yang digunakan. Salah satu media transmisi yang umum digunakan adalah *Ethernet* dengan maksimum *MTU* adalah 1500 yang berarti paket yang ditransmisikan pada *Ethernet Frame (datalink layer)* tidak dapat melebihi 1500 *bytes.*

2.2.13 *Pcap File Format*

Format .pcap *(Packet Capture)* merupakan format standar yang digunakan untuk penyimpanan hasil *capture* data jaringan. Paket yang tersimpan di dalam

format pcap tidak selalu berisi semua data seperti paket yang terdapat di jaringan jika *snaphot length* yang digunakan lebih kecil dari panjang paket yang terdapat di jaringan. Untuk mengatasi permasalahan ini kita dapat menerapkan *snapshot length* sepanjang 65535 (maksimum). Versi setelah pcap adalah pcap-ng, untuk lebih detailnya dapat dilihat di https://github.com/the-tcpdump-group/pcapng.

2.2.14 Libpcap/Npcap

Libpcap/Npcap merupakan *library* yang digunakan untuk mengirimkan atau meng-*capture* paket jaringan. Berikut beberapa fungsi yang dapat digunakan:

a. Pcap_findalldevs: Digunakan untuk mencari melihat kartu jaringan.
b. Pcap_open_live: Digunakan untuk sebagai *handler* fungsi dasar *library.*
c. Pcap_datalink: Digunakan untuk mendapakan tipe *datalink.*
d. Pcap_sendpacket: Digunakan untuk mengirimkan paket.
e. Pcap_next: Digunakan untuk meng-*capture* paket.
f. Pcap_compile: Digunakan untuk meng-*compile* filter paket.
g. Pcap_set_filter: Digunakan untuk mem-filter paket.
h. Pcap_close: Digunakan untuk menutup *handler.*

BAB III
METODE PENELITIAN

3.1 Objek Penelitian

Dalam *Local Area Network (LAN)*, paket *Internet Protocol (IP)* umumnya dikirim melalui *Ethernet Card* (kartu jaringan/*NIC*). Untuk keperluan komunikasi sesama *Ethernet Card* digunakan *Ethernet Address*, dalam hal ini adalah *MAC Address* yang besarnya 48 bit dan setiap kartu jaringan memiliki alamat yang berbeda-beda. Pada waktu pengiriman data dengan *IP* tertentu, suatu *host* perlu mengetahui di atas *Ethernet Card* mana *IP* tersebut terletak. Untuk keperluan pemetaan *IP Address* dengan *Ethernet Address (MAC Address)* inilah *ARP* digunakan.

Hasil dari pemetaan *IP Address* ini akan disimpan di dalam *ARP Cachce/ARP Table* dengan bertujuan agar tidak mempersibuk jaringan ketika akan melakukan komunikasi antar *Ethernet Card*. *Address Resolution Protocol (ARP)* bertanggung bertanggung jawab dalam pencarian *Media Access Control (MAC) Address* dari setiap komputer yang akan berkomunikasi melalui jaringan *Local Area Network (LAN)* dengan memanfaatkan *Internet Protocol Address (IP Address)* versi 4 yang telah didapat saat sebuah komputer terkoneksi ke dalam jaringan.

3.2 Metode Penelitian

3.2.1 Pengumpulan Data

Metode dan prosedur yang penulis gunakan untuk mendapatkan suatu data atau informasi tentang apa saja yang harus dikerjakan pada saat pengembangan sistem adalah sebagai berikut:

1. **Observasi**

   Kegiatan yang dilakukan adalah dengan mengamati dan menganalisa setiap paket *ARP* yang dapat di-*capture* pada jaringan. Hasil dari kegiatan ini akan dijadikan acuan untuk menentukan metode yang tepat untuk menyelesaikan masalah

2. **Analisis Kebutuhan**

   Pada kegiatan ini akan dilakukan analisis kebutuhan sistem baik perangkat keras maupun perangkat lunak. Selain itu juga akan dilakukan analisis akan kebutuhan calon pengguna sistem yang dibuat akan tepat guna.

3.2.2 Analisis Perancangan

Dalam memenuhi kebutuhan pengguna, sistem ini membutuhkan dukungan *hardware* dan *software* diantaranya *Network Interface Card (NIC)*, Libpcap untuk GNU/Linux sebagai *packet capture library* dan Npcap yang merupakan versi lain dari Libpcap bagi pengguna Windows. Libpcap/Npcap ini juga digunakan untuk menyimpan hasil dari paket-paket yang berhasil di-*capture*.

3.2.3 Pembuatan Program

Sistem ini akan diimplementasikan dengan menggunakan bahasa pemrograman Java dan C (digunakan untuk pembuatan *library/modul packet capture* dan *packet sender*). Sedangkan penyimpanan hasil *capture* paket akan menggunakan format pcap ataupun pcapng.

3.2.4 Implementasi dan Pengujian

Sistem ini akan diimplementasikan pada beberapa komputer yang menggunakan sistem operasi Linux dan Windows, selain itu akan dilakukan beberapa kali pengujian sebelum dan saat sistem digunakan oleh pengguna.

BAB IV
ANALISA DAN PERANCANGAN SISTEM

4.1 Analisa Sistem yang Diusulkan

Paket *IP* pada *LAN* akan dikirim melalui *Ethernet Card* yang alamat fisiknya (*MAC Address*) disimpan di dalam *ARP Cache*. Perubahan *ARP Cache* dapat terjadi ketika host menerima paket *ARP Reply* yang berisi alamat fisik (*MAC Address* dari *Ethernet Card*) dimana sebuah *IP Address* diletakkan.

Setiap paket *IP* pada *LAN* akan dikirimkan sesuai dengan alamat yang tersimpan di dalam *ARP Cache*. Jika *ARP Cache* tersebut di-*update* oleh *host* lain dengan mengirimkan paket *ARP Reply* yang mana paket tersebut telah dibuat sesuai dengan keinginan pengguna maka paket *IP* dapat terkirim ke *host* lain sesuai dengan *ARP Cache* yang telah ter-*update*.

Perubahan alamat dari *Ethernet Card (MAC Address)* dimana *IP address* diletakan merupakan ciri utama dari serangan *ARP Spoofing*. Untuk mengetahui apakah penyerang mengizinkan *IP Packet Routing* atau tidak dapat dilakukan dengan mengirimkan *TCP syn packet* dengan *destination* adalah *IP* milik *host* pengirim tersebut. Jika paket *TCP syn* tersebut di-*forward* oleh *attacker* maka dapat dipastikan penyerang megizinkan *IP Packet routing* miliknya.

4.2 Analisa Kebutuhan

4.2.1 Kebutuhan Fungsional

Sistem ini dapat melakukan beberapa fungsi, diantaranya:

a. Melakukan pemilihan kartu jaringan secara otomatis.
b. Melakukan pengecekan paket *ARP*.
c. Memberikan notifikasi kepada pengguna.
d. Menyimpan dan membaca hasil dari paket yang telah di-*capture.*

4.2.2 Kebutuhan Non Fungsional

Dibutuhkan beberapa *hardware* maupun *software* agar sistem ini dapat berjalan, diantaranya:

a. Kebutuhan Perangkat Keras
   - *Router dan Switch.*
   - Komputer penyerang (dengan *Ethernet Card*).
   - Komputer target (dengan *Ethernet Card*).

b. Kebutuhan Perangkat Lunak
   - Sistem Operasi Windows/GNU Linux.
   - Npcap untuk sistem operasi Windows.
   - Java Runtime Environment 1.8.0 (minimal).

4.3 Analisa Pengembangan Sistem

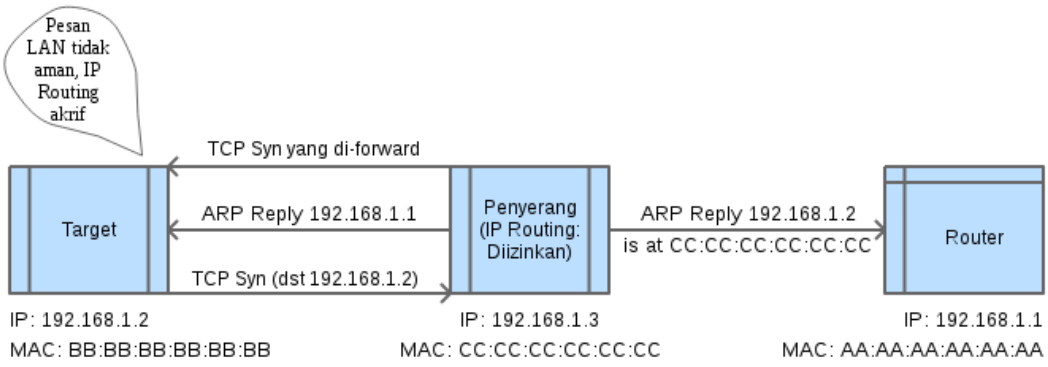
Sistem notifikasi ini melakukan deteksi dengan menggunakan paket *ARP* yang berhasil di-*capture* sebagai *input* yang akan diproses. Proses dilakukan dengan mengektraksi paket *ARP* kemudian memastikan bahwa paket tersebut ditujukan pada sistem. Jika paket tidak ditujukan pada sistem maka akan dilakukan sistem akan melakukan *capture* ulang pada paket ARP. Namun jika paket ditujukan pada sistem maka *IP* dan *MAC Address* akan disimpan ke dalam tabel. Apabila *IP* sebelumnya sudah tersimpan di dalam tabel maka dilakukan pengecekan terhadap *MAC Address*. Jika terjadi perubahan pada *MAC Address* maka akan dijalankan modul *TCP*. Sebaliknya jika tidak terjadi perubahan maka akan dilakukan pegecekan Ethernet frame. *Jika Ethernet frame* memiliki ukuran kurang dari 60 *bytes* dan *Sender Hardware Address (MAC Address)* tidak terdapat di dalam daftar *Organization Unique Identifier (OUI)* maka modul *TCP* akan dijalankan. Selain dari pada itu maka dianggap jaringan tidak mendapatkan gangguan.

Modul *TCP* melakukan pengecekan apakah penyerang mengizinkan *IP Paket Routing* pada sistem operasinya. Proses pengecekan dilakukan dengan mengirimkan paket *TCP syn* kepada penyerang dengan *destination IP* milik sistem yang akan mengirim. Jika *IP Paket Routing* penyerang *aktif* (diizinkan oleh

penyerang) maka paket *TCP syn* tersebut akan di-*forward* secara otomatis oleh penyerang kepada target. Dengan begitu target akan menerima paket yang sebelumnya ia kirimkan kepada penyerang dan sistem akan memberikan *output* berupa notifikasi bahwa jaringan LAN mendapatkan gangguan dan *IP Packet Routing* milik penyerang aktif. Jika paket *TCP syn* tidak di-*forward* oleh penyerang maka sistem akan memberikan notifikasi bahwa jaringan *LAN* mendapatkan gangguan dengan *IP Packet Routing* yang tidak diaktifkan oleh penyerang.

4.4 Rancangan Sistem

Sistem ini melakukan deteksi dengan cara melakukan *filtering* terhadap paket *ARP* yang di-*capture* oleh kartu jaringan *(Network Interface Card)* pada *promiscuous mode.* Setelah paket *ARP* diterima maka akan dilakukan ektraksi agar dapet diproses. Jika paket yang diterima merupakan pakct *ARP Reply* (*Ethertype*: 0x0806, dan *ARP Opcode*: 2) maka akan dilakukan pengecekan untuk memastikan bahwa paket tersebut adalah paket yang memang ditujuan kepada sistem yang sedang digunakan. Berikut adalah gambaran dari proses *ARP Spoofing* dimana *router* dan target mendapatkan paket *ARP Reply* dari penyerang dimana paket tersebut memberitahukan kepada *router* bahwa *MAC Address* dari target adalah CC:CC:CC:CC:CC:CC dan kepada target bahwa *MAC Address* dari *router* adalah CC:CC:CC:CC:CC:CC, sedangkan *MAC Address* tersebut adalah milik penyerang.



ARP Table Target

IP	MAC
192.168.1.1	CC:CC:CC:CC:CC:CC
192.168.1.2	BB:BB:BB:BB:BB:BB
192.168.1.3	CC:CC:CC:CC:CC:CC

ARP Table Router

IP	MAC
192.168.1.1	AA:AA:AA:AA:AA:AA
192.168.1.2	CC:CC:CC:CC:CC:CC
192.168.1.3	CC:CC:CC:CC:CC:CC

ARP Table Target pada kondisi normal

IP	MAC
192.168.1.1	AA:AA:AA:AA:AA:AA
192.168.1.2	BB:BB:BB:BB:BB:BB
192.168.1.3	CC:CC:CC:CC:CC:CC

ARP Table Router pada kondisi normal

IP	MAC
192.168.1.1	AA:AA:AA:AA:AA:AA
192.168.1.2	BB:BB:BB:BB:BB:BB
192.168.1.3	CC:CC:CC:CC:CC:CC

Gambar 4.1. Proses *ARP Spoofing*

Pada gambar 4.1. terlihat bahwa *ARP Table* milik *router* memiliki *IP* dari target dengan *MAC Address* CC:CC:CC:CC:CC:CC begitu juga dengan *ARP Table* milik target bahwa *IP router* memiliki *MAC Address CC:CC:CC:CC:CC:CC.* Dengan *ARP Table* sepeti gambar diatas maka paket *IP* akan dikirimkan kepada penyerang dan paket tersebut dapat juga diteruskan oleh penyerang dengan mengizinkan *IP Packet Routing.*

Sistem deteksi serangan pada *ARP* ini akan diperjelas dengan menggunkan algoritma sebagai berikut:

a. Penyerang
    1. Mengaktifkan *IP Packet Routing*.
    2. Mengirimkan ARP Reply ke target dengan *Sender Hardware Address (MAC Address)* yang telah diubah.

b. Target
    1. *Capture* paket *ARP*.

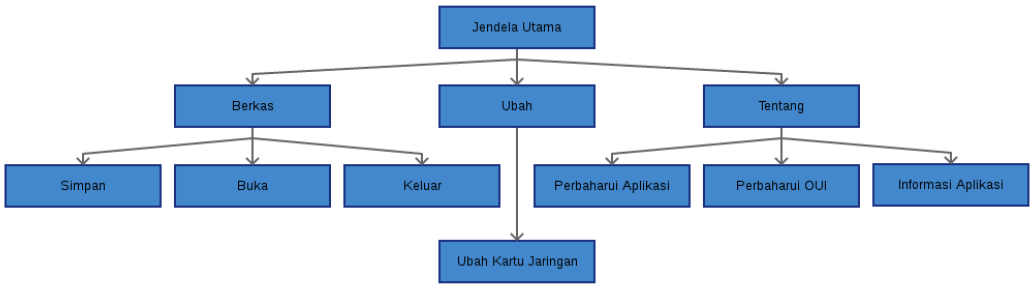
2. Jika bukan merupakan paket *ARP Reply*, maka *capture* paket berikutnya.
3. Apabila paket merupakan *ARP Reply* maka lakukan pengecekan apakah *IP* sudah tersimpan dalam tabel. Jika *IP* belum tersimpan maka simpan *IP* berserta *MAC Address* yang terdapat pada paket tersebut.
4. Jika *IP* dari paket *ARP Reply* tersebut sudah tersimpan di dalam tabel maka lakukan pengecekan apakah ada perubahan *MAC Address* dari *IP* tersebut. Jika terjadi perubahan yang menyebabkan ketidaksesuaian antara *IP* dan *MAC Address* maka jalankan modul *TCP*.
3. Jika tidak terjadi perubahan *MAC Address* pada *IP* tersebut maka lakukan pengecekan pada *Ethernet frame*. Jika *Ethernet frame* memiliki ukuran lebih kecil dari 60 *bytes* dan *Sender Hardware Address* tidak terdapat di dalam daftar *Organization Unique Identifier (OUI)* maka jalankan modul *TCP*.
4. Selain dari itu *LAN* dianggap tidak mengalami gangguan.

c. Modul *TCP*

1. Kirimkan *TCP syn packet* kepada penyerang dengan *destination* adalah *IP Address* dari pengirim.
2. Jika paket yang dikirimkan di-*forward* oleh penyerang maka tampilkan pesan bahwa jaringan tidak aman dengan *IP Routing* yang diaktifkan oleh penyerang.
3. Jika paket yang dikirimkan tidak di-*forward* oleh penyerang maka tampilkan pesan bahwa jaringan tidak aman.

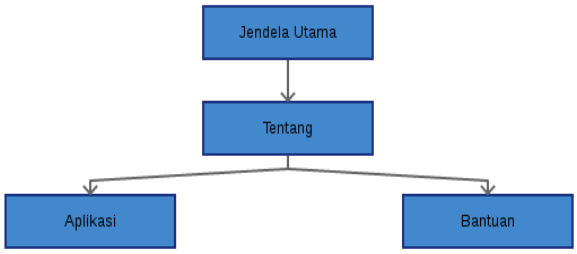
4.5 Rancangan Menu Dan Antar Muka

Gambar 4.2. menunjukan rancangan struktur menu aplikasi serang pada *ARP* yang dirancang untuk mengatur sistem.



Gambar 4.2. Struktur Menu Serangan

Gambar 4.3. menunjukan rancangan struktur menu aplikasi deteksi serang pada *ARP* yang dirancang untuk mengatur sistem.

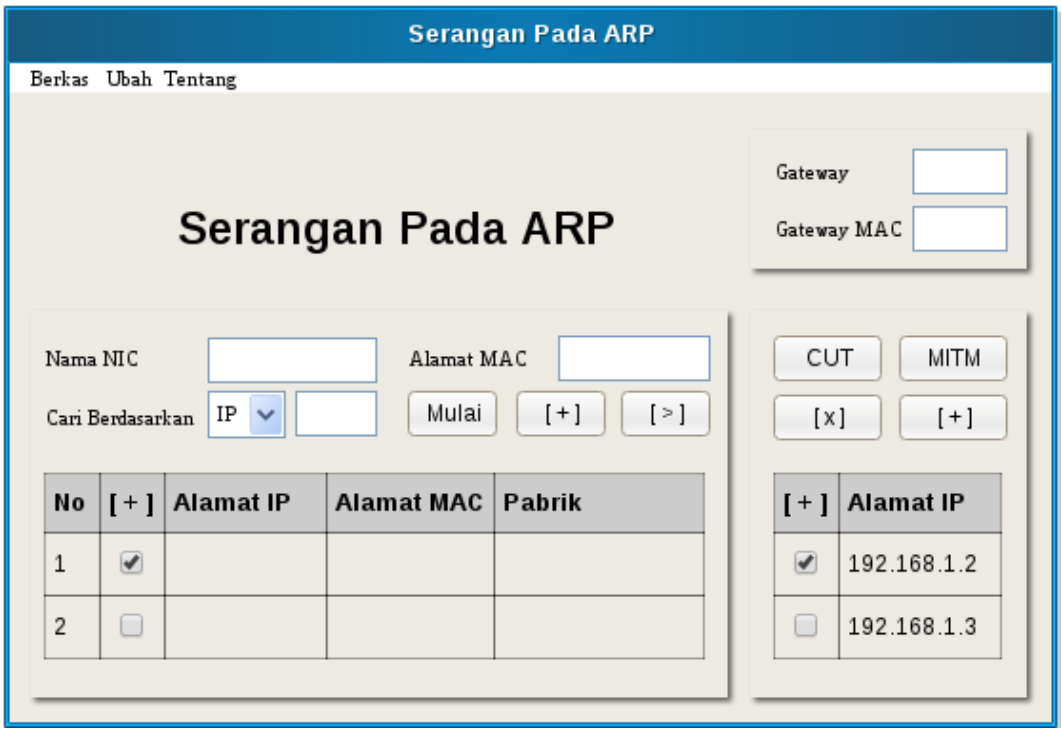


Gambar 4.3. Struktur Menu Deteksi Serangan

4.6 Rancangan *Graphical User Interface (GUI)*

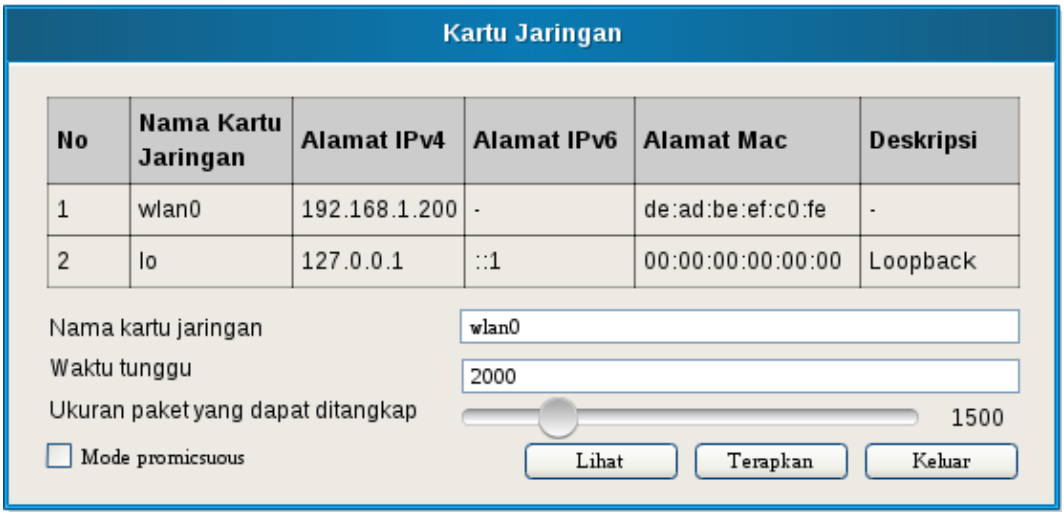
4.6.1 *Serangan Pada ARP*

Jendela ini digunakan untuk melakukan pencarian *host* yang terkoneksi pada *LAN* menggunakan *ARP* dimana hasilnya dapat digunakan untuk melakukan serangan. Pada jendela ini juga ditampilkan informasi jaringan seperti, kartu jaringan yang digunakan beserta *MAC Address*, *Gateway Address*, dan *Gateway MAC* Address.



Gambar 4.4. Serangan Pada *ARP*

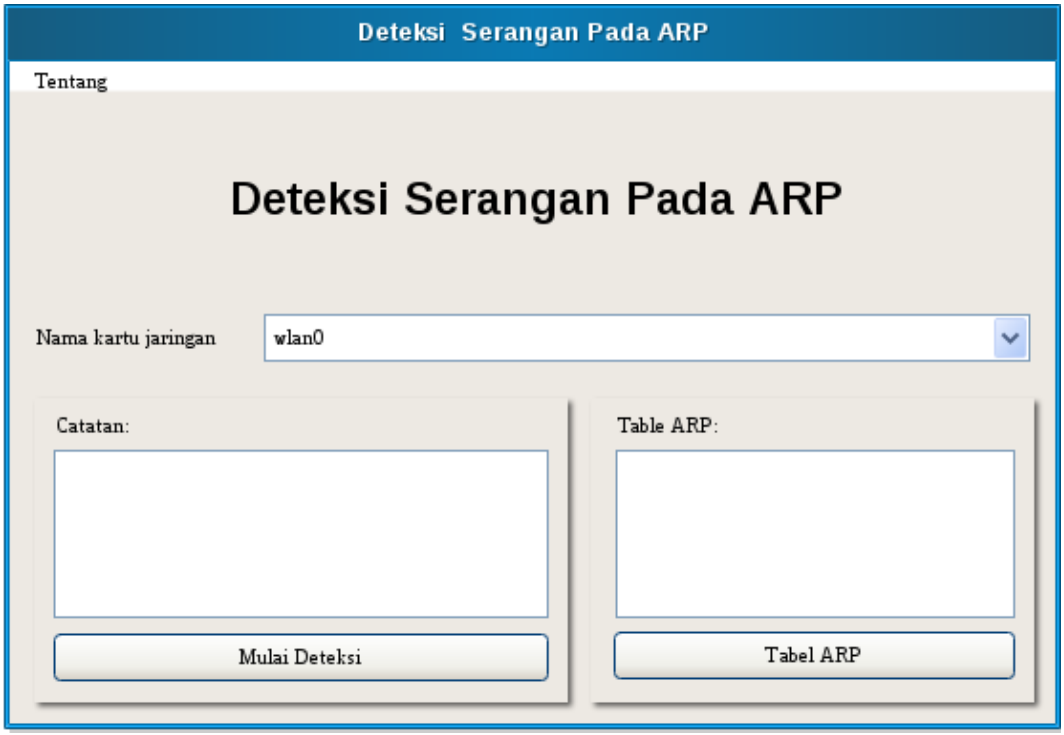
Jendela ini digunakan untuk melakukan konfigurasi kartu jaringan yang akan digunakan untuk melakukan serangan. Namun kartu jaringan yang dapat dipilih hanyalah kartu jaringan yang terkoneksi pada *LAN*.



Gambar 4.5. Konfigurasi Kartu Jaringan

4.6.2 *Deteksi Serangan Pada ARP*

Jendela berikut digunakan untuk mendeteksi serangan pada *ARP* dimana hasil deteksi akan ditampilkan dalam bentuk teks mupun notifikasi *(pop up)* yang bertujuan agar pengguna mengetahui tindakan apa yang harus dilakukan. Selain itu pada jendela ini ditambahkan fungsi untuk melakukan pengecekan *ARP Cache* (Tabel *ARP*) dimana dapat digunakan untuk membandingan ketika sistem sedang diserang atau tidak.



Gambar 4.6. Deteksi Serangan Pada *ARP*

BAB V
IMPLEMENTASI SISTEM

5.1 Implementasi

Sistem ini dapat diimplementasikan pada sistem operasi GNU/Linux (i686, amd64, dan armv7) dan Windows (x86/amd64). Agar dapat menjalankan fungsinya sistem ini juga membutuhkan koneksi pada *LAN*.

Dalam tahap ini akan diketahui bagaimana cara memulai, menjalankan, dan mengakhiri program yang telah dirancang pada bab sebelumnya.

5.2 Perangkat Keras (*Hardware*) yang Digunakan

Perangkat keras khusus yang dibutuhkan untuk mengoperasikan sistem ini adalah:

a. Notebook ASUS X200CA
b. Processor Genuine Intel Celeron(R) CPU 1007U 1.50 GHz
c. RAM 2GB
d. Hardisk 500GB
e. Network Interface Card (ALFA AWUS 360H)

5.3 Perangkat Lunak (*Software*) yang Digunakan

Perangkat lunak khusus yang dibutuhkan dalam membangun sistem ini adalah:

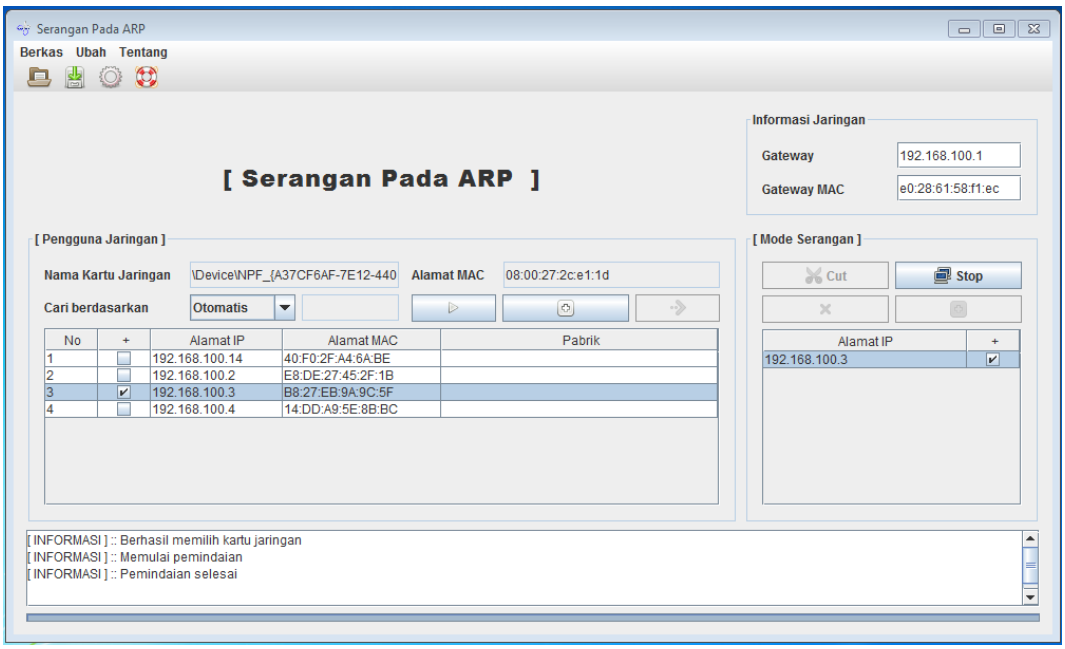
a. GNU/Linux, Windows 7, dan Windows 10
b. Netbeans dan Intellij IDEA
c. Jxnet dan Npcap

5.4 Implementasi Sistem

5.4.1 Implementasi Serangan

Untuk melakukan serangan menggunakan sistem yang telah dikembangkan adalah dengan melakukan pencarian target dengan menggunakan tombol pencarian kemudian menambahkannya ke dalam tabel target. Setelah

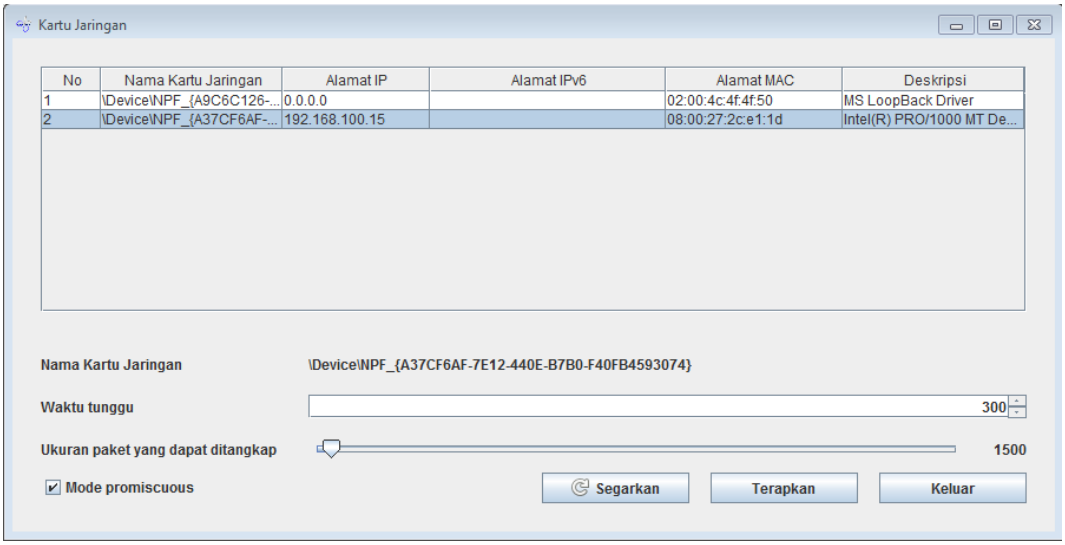
terdapat target di dalam tabel target maka serangan dapat dimulai dengan meng-klik tombol “Cut” ataupun “MITM”. Tombol “Cut” berfungsi untuk mengubah *ARP Cache* pada target dengan Gateway Mac Address yang acak dengan tujuan agar paket yang dikirimkan oleh target ke gateway tidak sampai pada tujuan. Sedangkan tombol “MITM” berfungsi untuk memberitahu kepada target bahwa penyerang adalah *gateway* dan memberi tahu *gateway* yang sesungguhnya bahwa penyerang adalah target. Berikut tampilah dari program untuk melakukan serangan.



Gambar 5.1. Form Serangan

5.4.2 Implementasi Konfigurasi Kartu Jaringan

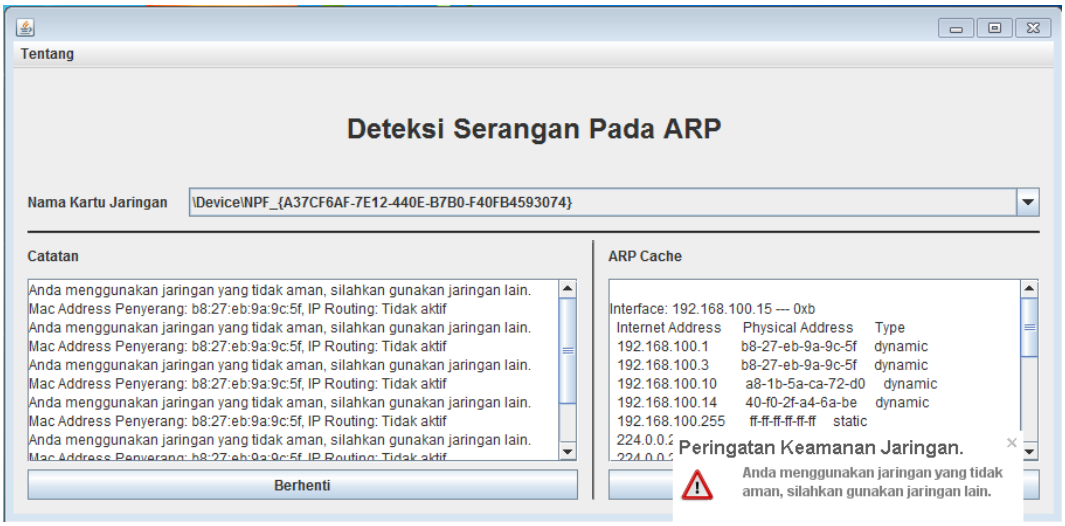
Ketika menjalankan program (untuk serangan) maka secara otomatis program akan melakukan pemilihan kartu jaringan yang akan digunakan dan terkoneksi pada *LAN*. Namun jika ingin melakukan konfigurasi manual dapat dilakukan dengan memilih “Ubah” kemudian klik “Kartu Jaringan”. Kartu jaringan yang dapat dipilih hanyalah kartu jaringan yang digunakan (terkoneksi pada *LAN*). Berikut tampilan dari form konfigurasi kartu jaringan.



Gambar 5.2. Form Konfigurasi Kartu Jaringan

5.4.3 Implementasi Deteksi Serangan

Sebelum memulai deteksi kartu jaringan terlebih dahulu harus terkoneksi pada *LAN*. Setelah terkoneksi untuk memulai deteksi dapat meng-klik tombol "Mulai Deteksi". Sedangkan untuk melihat "*ARP Cache*" untuk membandingkan antara diserang dengan tidak disertang dapat meng-klik tombol "Lihat ARP Cache".



Gambar 5.3. Form Deteksi Serangan

Berikut adalah kode sumber dimana sistem melakukan pencarian kartu jaringan yang terkoneksi pada *LAN:*

```
public static String LookupNetworkInterface(Inet4Address
 address, Inet4Address netmask, Inet4Address netaddr,
 Inet4Address broadaddr, Inet4Address dstaddr,
 MacAddress macAddress, StringBuilder description) {
  Preconditions.CheckNotNull(address);
  Preconditions.CheckNotNull(netmask);
  Preconditions.CheckNotNull(netaddr);
  Preconditions.CheckNotNull(broadaddr);
  Preconditions.CheckNotNull(dstaddr);
  Preconditions.CheckNotNull(description);
  StringBuilder errbuf = new StringBuilder();
  List<PcapIf> ifs = new ArrayList<PcapIf>();
  if (Jxnet.PcapFindAllDevs(ifs, errbuf) != Jxnet.OK)
   return null;
  description.setLength(0);
  for (PcapIf dev : ifs) {
   for (PcapAddr addr : dev.getAddresses()) {
    if (addr.getAddr().getData() == null ||
      addr.getBroadAddr().getData() == null ||
      addr.getNetmask().getData() == null) {
     continue;
    }
    if (addr.getAddr().getSaFamily() == SockAddr.Family.AF_INET
      &&!Inet4Address.valueOf(addr.getAddr().getData())
      .equals(Inet4Address.ZERO) &&!Inet4Address
      .valueOf(addr.getAddr().getData())
      .equals(Inet4Address.LOCALHOST)
      &&!Inet4Address.valueOf(addr.getBroadAddr()
      .getData()).equals(Inet4Address.ZERO) &&
      !Inet4Address.valueOf(addr.getNetmask()
      .getData()).equals(Inet4Address.ZERO)) {
       address.update(Inet4Address
         .valueOf(addr.getAddr().getData()));
       netmask.update(Inet4Address
         .valueOf(addr.getNetmask().getData()));
       netaddr.update(Inet4Address
         .valueOf(address.toInt() & netmask.toInt()));
       broadaddr.update(Inet4Address
         .valueOf(addr.getBroadAddr().getData()));
       if (addr.getDstAddr().getData() != null)
        dstaddr.update(Inet4Address
          .valueOf(addr.getDstAddr().getData()));
       } else { dstaddr.update(Inet4Address.ZERO); }
       macAddress.update(MacAddress.fromNicName(dev.getName()));
       if (dev.getDescription() != null) {
        description.append(dev.getDescription());
       }
       return dev.getName();
      }
     }
    }
   return null;
  }
```

Berikut adalah kode sumber dimana dilakukan pengiriman paket *ARP Request* dan proses *capture* paket *ARP Reply* guna mendapatkan *MAC Address* dari *gateway*:

```
public static MacAddress getGwHwAddrFromArp() {
 Packet arp = new ARP()
   .setHardwareType(DataLinkType.EN10MB)
   .setProtocolType(ProtocolType.IPV4)
   .setHardwareAddressLength((byte) 6)
   .setProtocolAddressLength((byte) 4)
   .setOperationCode(ARPOperationCode.ARP_REQUEST)
   .setSenderHardwareAddress(MAC_ADDRESS)
   .setSenderProtocolAddress(ADDRESS)
   .setTargetHardwareAddress(MacAddress.ZERO)
   .setTargetProtocolAddress(GATEWAY_ADDRESS).build();
 Packet ethernet = new Ethernet()
   .setDestinationMacAddress(MacAddress.BROADCAST)
   .setSourceMacAddress(MAC_ADDRESS)
   .setEthernetType(ProtocolType.ARP)
   .setPacket(arp).build();
 ByteBuffer buffer =
   FormatUtils.toDirectBuffer(ethernet.toBytes());
 PcapPktHdr pktHdr = new PcapPktHdr();
 for (int i=0; i<50; i++) {
  if (Jxnet.PcapSendPacket(ARP_HANDLER, buffer,
    buffer.capacity()) != 0) { return null; }
  Map<Class, Packet> packets =
    PacketHelper.next(ARP_HANDLER, pktHdr);
  if (packets == null) continue;
  ARP arpCap = (ARP) packets.get(ARP.class);
  if (arpCap == null) continue;
  if (arpCap.getOperationCode() == ARPOperationCode.ARP_REPLY &&
    arpCap.getSenderProtocolAddress().equals(GATEWAY_ADDRESS)) {
   return arpCap.getSenderHardwareAddress();
  }
  try{
   Thread.sleep(StaticField.TIMEOUT);
  }catch(InterruptedException e){
   System.out.println(e);}
  }
 return null;
}
```

Berikut adalah kode sumber proses deteksi terhadap *ARP Spoofing* dimana terdapat juga modul untuk melakukan deteksi apakah penyerang megizinkan *IP Packet Routing* atau tidak:

```
@Override
public void run() {
 PacketHandler<String> packetHandler =(arg, pktHdr, packets)-> {
  Ethernet ethernet = (Ethernet) packets.get(Ethernet.class);
  if (ethernet == null || ethernet.getEthernetType() !=
    ProtocolType.ARP) { return; }
   ARP arp = (ARP) packets.get(ARP.class);
   if (arp == null) { return; }
   MacAddress ethDst = ethernet.getDestinationMacAddress();
   MacAddress ethSrc = ethernet.getSourceMacAddress();
   MacAddress sha = null; MacAddress tha = null;
   Inet4Address spa = null; Inet4Address tpa = null;
   sha = arp.getSenderHardwareAddress();
   tha = arp.getTargetHardwareAddress();
   spa = arp.getSenderProtocolAddress();
   tpa = arp.getTargetProtocolAddress();
   if (arp.getOperationCode() != ARPOperationCode.ARP_REPLY ||
     !ethDst.equals(StaticField.MAC_ADDRESS) ||
     tpa.equals(StaticField.MAC_ADDRESS) ||
     ethSrc.equals(StaticField.GATEWAY_MAC_ADDRESS)) { return; }
   if (!ethSrc.equals(sha) || !ethDst.equals(tha)) {
    TCPTrap.newThread(sha).start();
   } else {
    MacAddress shaCache = StaticField.ARP_CACHE.get(spa);
    if  (shaCache == null) {
     StaticField.ARP_CACHE.put(spa, sha);
    } else {
     if (!sha.equals(shaCache)) {
      TCPTrap.newThread(sha).start();
     } else {
      boolean UNPADDED_ETHERNET_FRAME = false;
      boolean UNKNOWN_OUI = false;
      boolean BAD_DELTA_TIME = false;
     UNPADDED_ETHERNET_FRAME=(pktHdr.getCapLen()<60?true:false);
      if(OUI.searchVendor(arp.getSenderHardwareAddress().
        toString()).equals("")) { UNKNOWN_OUI = true; }
      Long epochTimeCache = StaticField.EPOCH_TIME.get(spa);
      if (epochTimeCache == null || epochTimeCache == 0) {
       StaticField.EPOCH_TIME.put(spa, pktHdr.getTvUsec());
      } else {
       long time = (pktHdr.getTvUsec() - epochTimeCache);
       if (time < StaticField.TIME) {
        BAD_DELTA_TIME = true;
       }
       StaticField.EPOCH_TIME.put(spa, pktHdr.getTvUsec());
      }
     if((UNPADDED_ETHERNET_FRAME&&UNKNOWN_OUI)||BAD_DELTA_TIME){
       TCPTrap.newThread(sha).start();
      } else { // }
      StaticField.ARP_CACHE.put(spa, sha);
     }
    }
 };
 PacketHelper.loop(StaticField.ARP_HANDLER, -1, packetHandler,
   null);
}
```

```
public void run() {
 if (StaticField.TCP_HANDLER == null) { return; }
 short sourcePort=(short)StaticField.random.nextInt(65535-1+1);
 Inet4Address sourceAddress =
   Inet4Address.valueOf(StaticField.random.nextInt());
 Packet tcp = new TCP()
   .setSourcePort(sourcePort).setDestinationPort((short) 80)
   .setSequence(0).setAcknowledge(0).setDataOffset((byte) 40)
   .setFlags(TCPFlags.newInstance((short) 2))
   .setWindowSize((short) 29200).setUrgentPointer((short) 0)
   .setOptions(OPTIONS).build();
 Packet ipv4 = new IPv4()
   .setVersion((byte) 0x4).setDiffServ((byte) 0x0)
   .setExpCon((byte) 0).setIdentification((short) 29257)
   .setFlags((byte) 0x02).setFragmentOffset((short) 0)
   .setTtl((byte) 64).setProtocol(IPProtocolType.TCP)
   .setSourceAddress(sourceAddress)
   .setDestinationAddress(StaticField.ADDRESS).setPacket(tcp)
   .build();
 Packet tcpTrap = new Ethernet()
   .setDestinationMacAddress(dha)
   .setSourceMacAddress(StaticField.MAC_ADDRESS)
   .setEthernetType(ProtocolType.IPV4).setPacket(ipv4).build();
 ByteBuffer buffer =
   FormatUtils.toDirectBuffer(tcpTrap.toBytes());
 Map<Class, Packet> packetMap;
 PcapPktHdr pktHdr = new PcapPktHdr();
 if (Jxnet.PcapSendPacket(StaticField.TCP_HANDLER, buffer,
   buffer.capacity()) != 0) { return; }
 Map<Class, Packet> packets =
   PacketHelper.next(StaticField.TCP_HANDLER, pktHdr);
 if (packets != null) {
  Ethernet ethernet = (Ethernet) packets.get(Ethernet.class);
  if (ethernet != null) {
   if (ethernet.getDestinationMacAddress()
     .equals(StaticField.MAC_ADDRESS)) {
    TCP tcpCap = (TCP) packets.get(TCP.class);
    IPv4 ipv4Cap = (IPv4) packets.get(IPv4.class);
    if (tcpCap != null && ipv4Cap != null) {
     if (tcpCap.getDestinationPort() == (short) 80 &&
       tcpCap.getSourcePort() == sourcePort &&
       ipv4Cap.getDestinationAddress()
       .equals(StaticField.ADDRESS) &&
       ipv4Cap.getSourceAddress()
       .equals(sourceAddress)) {
      if (StaticField.LOGGER != null)
       StaticField.LOGGER.log("IP Routing aktif");
      if (StaticField.IPS) { ARPPing.newThread().start(); }
      return;
     }
    }
   }
  }
 }
```

```
if (StaticField.LOGGER != null)
      StaticField.LOGGER.log("IP Routing aktif");
      if (StaticField.IPS) { ARPPing.newThread().start();
}
```

BAB VI
PENUTUP

6.1 Kesimpulan

Setelah sistem notifikasi ini diaplikasikan dapat disimpulkan bahwa sistem ini dapat melakukan pengecekan paket-paket *ARP* sehingga dapat mendeteksi serangan *ARP Spoofing/Arp Cache Poisoning* dan memberikan notifikasi berupa saran kepada pengguna *LAN*.

6.2 Saran

Dari hasil pengaplikasian sistem notifikasi ini diharapkan adanya pengembangan sistem yang disesuaikan dengan perkembangan teknologi dan berbagai macam jenis serangan yang dapat membahayakan, diantaranya:

1. Tidak hanya mampu mendeteksi *address resolution* pada *IPv4 (ARP),* namun juga *address resolution* pada *IPv6 (NDP).*
2. Mampu mendeteksi serangan pengembangan dari *ARP Spoofing* seperti *Sniffing, DNS Spooing,* dan lain sebagainya.

Dimana sistem tersebut dapat memberikan rasa aman bagi para penggunanya ketika menggunakan *LAN*.

DAFTAR PUSTAKA

Kaur, I., (2013), Detection and Prevention of ARP Cache Poisoning, Thesis, Computer Science and Engineeting Department, Thapar University, Patiala.

Srinath, D., Panimalar S., Simla, A. J., dan Deepa, J., (2015), Detection and Prevention of ARP Spoofing using Centralized Server, Internation Journal of Computer Applications, *113*, Departement of Computer science and Engineering, Panimalar Institute of Technology, India.

Vinay, K.R., dan Gudur, B.K., (2014), ARP Spoof Detection System Using ICMP Protocol: An Active Approach, International Journal of Engineering Research and Technologi (IJERT), Vol 3.

Sugeng, W., Putri, T.D., (2015), Jaringan Komputer dengan TCP/IP, Bandung: Modula.

LAMPIRAN

Tabel Penjelasan *Field-field TCP*

Nama field	**Ukuran**	**Keterangan**
Source Port	16 bit	Mengindikasikan sumber protokol lapisan aplikasi yang mengirimkan segmen *TCP* yang bersangkutan. Gabungan antara *field Source IP Address* dalam *header IP* dan *field Source Port* dalam *field header TCP* disebut juga sebagai *source socket*, yang berarti sebuah alamat global dari mana segmen dikirimkan.
Destination Port	16 bit	Mengindikasikan tujuan protokol lapisan aplikasi yang menerima segmen *TCP* yang bersangkutan. Gabungan antara *field Destination IP Address* dalam *header IP* dan *field Destination Port* dalam *field header TCP* disebut juga sebagai *socket* tujuan, yang berarti sebuah alamat global ke mana segmen akan dikirimkan.
Sequence Number	32 bit	Mengindikasikan nomor urut dari oktet pertama dari data di dalam sebuah segmen *TCP* yang hendak dikirimkan. *Field* ini harus selalu diset, meskipun tidak ada data (*payload*) dalam segmen. Ketika memulai sebuah sesi koneksi *TCP*, segmen dengan *flag SYN (Synchronization)* diset ke nilai 1, field ini akan berisi nilai *Initial Sequence Number (ISN)*. Hal ini berarti, oktet pertama dalam aliran *byte (byte stream)* dalam koneksi adalah *ISN*+1.

Penggalan Tabel Penjelasan *Field-field TCP* pada lampiran halaman 41

Acknowledgment Number	32 bit	Mengindikasikan nomor urut dari oktet selanjutnya dalam aliran *byte* yang diharapkan oleh untuk diterima oleh pengirim dari si penerima pada pengiriman selanjutnya. *Acknowledgment* number sangat dipentingkan bagi segmen-segmen *TCP* dengan *flag ACK* diset ke nilai 1.
Data Offset	4 bit	Mengindikasikan di mana data dalam segmen *TCP* dimulai. *Field* ini juga dapat berarti ukuran dari *header TCP*. Seperti halnya *field Header Length* dalam *header IP*, *field* ini merupakan angka dari *word* 32-bit dalam *header TCP*. Untuk sebuah segmen *TCP* terkecil (di mana tidak ada opsi *TCP* tambahan), *field* ini diatur ke nilai 0x5, yang berarti data dalam segmen *TCP* dimulai dari oktet ke 20 dilihat dari permulaan segmen *TCP*. Jika *field Data Offset* diset ke nilai maksimumnya (2^4=16) yakni 15, *header TCP* dengan ukuran terbesar dapat memiliki panjang hingga 60 *byte*.
Reserved	6 bit	Direservasikan untuk digunakan pada masa depan. Pengirim segmen *TCP* akan mengeset bit-bit ini ke dalam nilai 0.
Flags	6 bit	Mengindikasikan *flag-flag TCP* yang memang ada enam jumlahnya, yang terdiri atas: *URG (Urgent), ACK (Acknowledgment), PSH (Push), RST (Reset), SYN (Synchronize),* dan *FIN (Finish)*.

Penggalan Tabel Penjelasan *Field-field TCP* pada lampiran halaman 42

Window	16 bit	Mengindikasikan jumlah *byte* yang tersedia yang dimiliki oleh *buffer host* penerima segmen yang bersangkutan. *Buffer* ini disebut sebagai *Receive Buffer*, digunakan untuk menyimpan *byte stream* yang datang. Dengan mengimbuhkan ukuran *window* ke setiap segmen, penerima segmen *TCP* memberitahukan kepada pengirim segmen berapa banyak data yang dapat dikirimkan dan disangga dengan sukses. Hal ini dilakukan agar si pengirim segmen tidak mengirimkan data lebih banyak dibandingkan ukuran *Receive Buffer*. Jika tidak ada tempat lagi di dalam *Receive buffer*, nilai dari *field* ini adalah 0. Dengan nilai 0, maka si pengirim tidak akan dapat mengirimkan segmen lagi ke penerima hingga nilai *field* ini berubah (bukan 0). Tujuan hal ini adalah untuk mengatur lalu lintas data atau *flow control*.
Checksum	16 bit	Mampu melakukan pengecekan integritas segmen *TCP* (*header*-nya dan *payload*-nya). Nilai *field Checksum* akan diatur ke nilai 0 selama proses kalkulasi *checksum*.
Urgent Pointer	16 bit	Menandakan lokasi data yang dianggap "*urgent*" dalam segmen.
Options	32 bit	Berfungsi sebagai penampung beberapa opsi tambahan *TCP*. Setiap opsi *TCP* akan memakan ruangan 32 bit, sehingga ukuran *header TCP* dapat diindikasikan dengan menggunakan *field Data offset*.

Kode sumber pencarian *host* yang memungkinkan untuk melakukan koneksi pada *LAN*.

```
List<Inet4Address> ips = new ArrayList<Inet4Address>();
SubnetUtils su = new SubnetUtils(
    StaticField.NETWORK_ADDRESS.toString(),
    StaticField.NETMASK_ADDRESS.toString()
  );

 String[] strips = su.getInfo().getAllAddresses();
for (String ip : strips) {
  ips.add(Inet4Address.valueOf(ip.trim()));
}
```

Kode sumber untuk menampilkan pesan.

```
public interface Logger {
    public void log(String message, String message2);
}
```

Kode sumber pencarian pabrik pembuat *Ethernet Card*.

```
public static String searchVendor(String MacAddr) {
  if (MacAddr == null) return "";
    MacAddr = MacAddr.trim().substring(0, 8).toUpperCase();
    final String vendorId = MacAddr;
    String res = null;
    try (Stream<String> lines = Files.lines(new
        File("oui.txt").toPath(), Charset.defaultCharset())) {
      try {
        res = lines.filter(l -> l.startsWith(vendorId))
            .findFirst().orElse("");
      } catch (Exception e) {
        return "";
      }
    } catch (IOException ex) {
      return "";
    }
    if (res == null) return "";
    String[] vendorName = res.split("#");
    if (vendorName == null) return "";
    return (vendorName[vendorName.length-1] == null) ? "" :
        vendorName[vendorName.length-1].trim();
}
```