BAB III

METODOLOGI PENELITIAN

3.1 Gambaran Umum Penelitian

Penelitian ini dilakukan untuk menghindari komputer *client* terkena serangan ARP *spoofing* dan mencegahnya terjadi kembali dengan menggunakan program aplikasi dengan metode ESV-ARP.

Algoritma dalam penelitian ini disebut sebagai metode ESV-ARP yang menggunakan penggabungan dua metode berbeda yaitu *Efficient and Secure* (ES-ARP) dan *Voting*. Pembuatan aplikasi ini untuk membuktikan bahwa ide permasalahan pada penelitian dapat diwujudkan.

3.2 Metode Penelitian

Metode penelitian yang digunakan pada penelitian kali ini dapat dijelaskan sebagai berikut:

a. Studi Literatur

Melakukan studi kepustakaan yang berkaitan dengan penelitian yang dilakukan dengan mengumpulkan berbagai referensi. Topik-topik atau teori yang dikaji antara lain meliputi pengenalan ARP, pengenalan *spoofing*, ARP *spoofing*, metode-metode pendeteksian dan pencegahan ARP *spoofing* yang telah diteliti sebelumnya, serta konsep pendukung

lainnya. Referensi-referensi yang digunakan dapat berupa buku, jurnal ilmiah, forum, artikel, dan lain-lain.

b. Analisis dan Perancangan Aplikasi

Menganalisa berbagai metode untuk menentukan metode mana yang akan dipilih sebagai acuan serta melakukan perancangan awal aplikasi yang akan dibuat. Perancangan meliputi perancangan *flowchart* dan *user interface* agar aplikasi menjadi mudah digunakan.

c. Pembuatan Aplikasi

Melakukan pembuatan aplikasi dengan mengimplementasikan rancangan berdasarkan metode dan bahasa pemrograman yang telah ditentukan.

d. Uji Coba dan Pembahasan

Dilakukan uji coba terhadap aplikasi yang telah dibuat dan mengevaluasi hasil yang didapat.

3.3 Spesifikasi Umum Sistem

Telah dijelaskan bahwa metode yang diusulkan memiliki kelemahan dan kelebihan masing-masing. ES-ARP memiliki kelemahan pada validasi paket yang datang. Dengan kata lain, jika penyerang lebih dulu menyebarluaskan *mapping* yang palsu, maka satu jaringan akan terkena ARP *spoofing*. Sedangkan kelemahan metode *Voting* terletak pada jumlah paket yang dikirimkan pada satu jaringan, yaitu sebanyak 50 paket *voting request*

untuk satu kali bertanya dan 50 paket *voting reply* untuk satu kali menjawab. Hal tersebut dapat memenuhi *traffic* pada jaringan. Solusi yang diajukan yaitu dengan mengadopsi kelebihan dari teknik kedua metode, antara lain paket *request* dan *reply* dilakukan secara *broadcast* sehingga lebih efisien, menggunakan dua tabel ARP sebagai acuan validasi paket, menggunakan paket baru dengan *opcode* yang baru pula, serta hanya 1 paket *voting request* dan 2 paket *voting reply* yang dikirimkan.

Pada prinsipnya, program aplikasi ini akan aktif bila di-*execute* dan segera *listening* semua paket ARP yang ada pada jaringan secara terus menerus. Setiap kali mendapatkan IP *conflict*, program akan *throw* paket ARP *voting request* secara *broadcast* dengan *opcode* eksperimental yaitu 24. Komputer yang bersangkutan akan merespon balik dengan mengirimkan 2 paket ARP *voting reply* secara *broadcast* juga, dengan syarat komputer tersebut telah memiliki program aplikasi yang dimaksud.

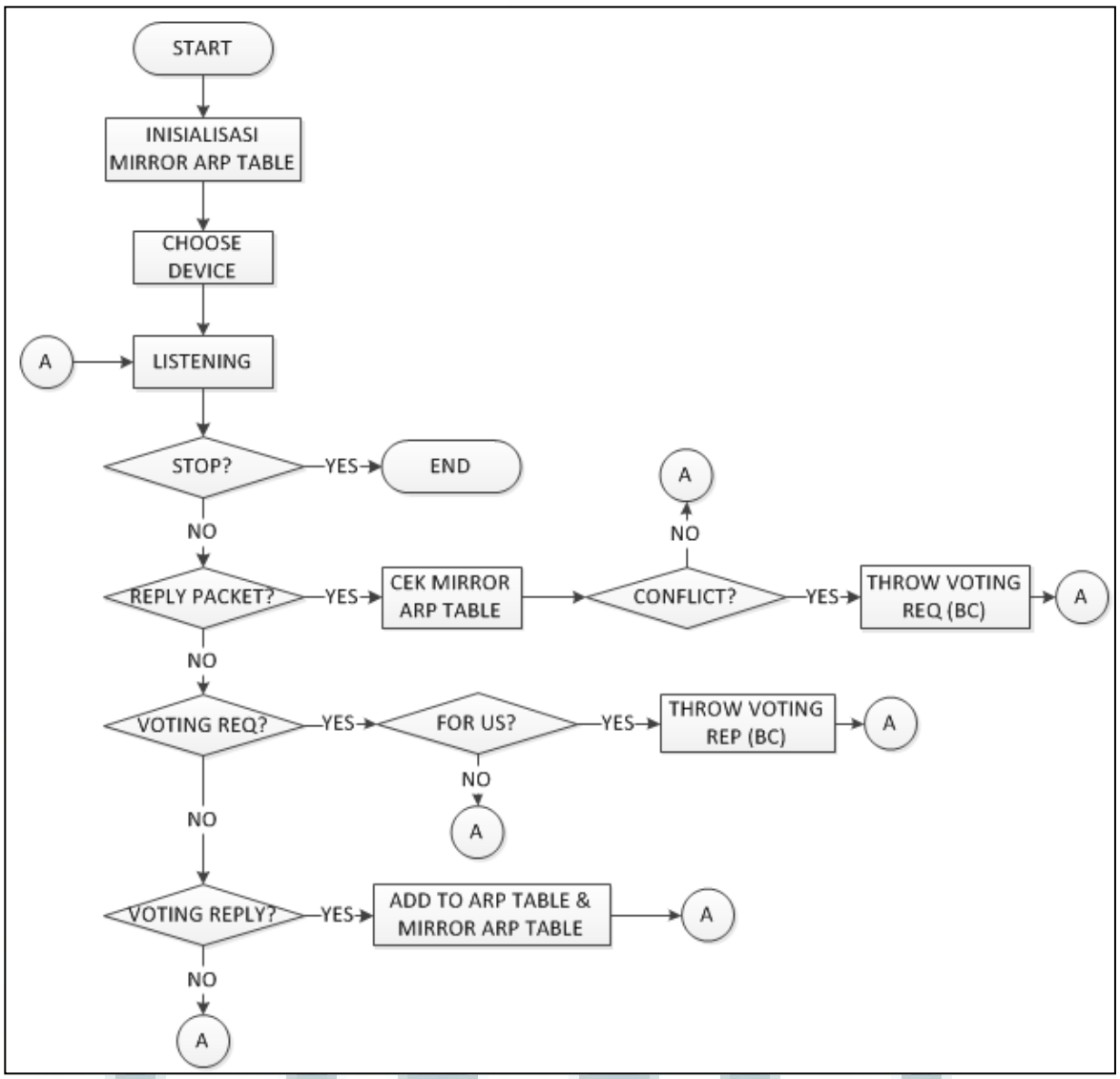
Komputer yang mendapatkan paket ARP *voting reply* akan melakukan *polling*, yaitu: jika terdapat 2 paket *voting reply*, maka *mapping* IP-MAC tersebut akan ditambahkan ke tabel ARP masing-masing. Namun, berdasarkan acuan jurnal, setidaknya terdapat satu paket *voting reply* yang diterima akan ditambahkan ke tabel ARP *default* maupun *mirror ARP table*. *Mirror ARP table* merupakan tabel acuan untuk mendeteksi apakah paket *reply* yang datang terdapat *IP conflict*. Data pada *mirror ARP table* didapat dari tabel

ARP *default* dari *driver* yang diinisialisasi saat pertama kali program dijalankan.

Apabila telah dilakukan *broadcast* paket ARP *voting request* tetapi tidak ada balasan, maka akan dianggap ada penyerang ARP *spoofing*, karena penelitian ini berasumsi bahwa penyerang tidak memiliki program aplikasi ESV-ARP.

3.4 *Flowchart*

Penelitian kali ini memiliki dua fungsi sekaligus yaitu deteksi dan pencegahan ARP *spoofing*. Proses deteksi memerlukan dua tabel yaitu tabel ARP Windows (*default*) dan tabel ARP acuan atau yang selanjutnya disebut sebagai *mirror ARP table*. Namun kedua tabel tersebut merupakan dua buah *array* 2D yang berbeda. Sedangkan proses pencegahan hanya melakukan proses *broadcast* paket ARP yang kemudian *mapping* IP-MAC sumber baik *voting request* maupun *voting reply* disimpan ke kedua tabel ARP. Proses program seluruhnya dapat dijelaskan melalui *flowchart* atau diagram alir pada Gambar 3.1.



Gambar 3. 1 *Flowchart* program ESV-ARP

Dari *flowchart* pada gambar 3.1 dapat dilihat bahwa program dimulai dengan memilih *device* atau *host* yang bersangkutan untuk menjalankan protokol ESV-ARP. Setelah itu, program akan *listening* atau *capture* paket yang datang namun hanya paket ARP yang ditangkap. Proses ini terus berlanjut secara terus menerus dan akan berakhir hingga *user* mengklik tombol "*Stop*".

Ketika ada paket yang datang, program akan selalu memeriksa jenis paket ARP apakah yang masuk. Hal tersebut dicek melalui *opcode* masing-masing paket. Jika *opcode* sama dengan 1, berarti merupakan paket *request*. Paket tersebut akan dihiraukan oleh program karena *by default* akan dijawab oleh *driver*.

Jika terdapat paket dengan *opcode* sama dengan 2, berarti merupakan paket *reply*. Paket tersebut akan diperiksa apakah terdapat IP *conflict* berdasarkan data *mirror ARP table*. Jika tidak, maka akan melanjutkan *listen* paket. Namun jika terdapat IP *conflict*, maka *device* akan mengirimkan paket *voting request* berisi IP yang mengalami *conflict* secara *broadcast*. Kemudian *device* kembali *listen* paket.

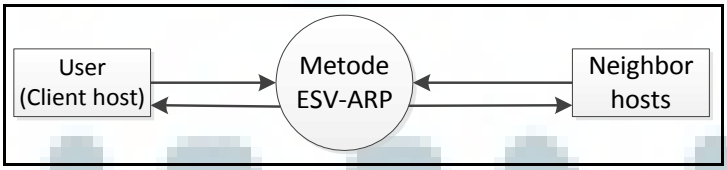
Jika terdapat paket dengan *opcode* sama dengan 24, berarti merupakan paket *voting request*. Paket tersebut akan diperiksa apakah mengandung IP untuk *host* yang bersangkutan (*local host*). Jika ya, maka *host* tersebut akan

mengirimkan paket *voting reply* secara *broadcast*. Jika tidak, *host* akan kembali *listen* paket.

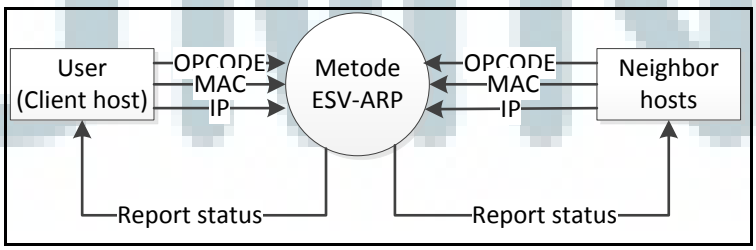
Jika paket yang datang memiliki *opcode* 25, berarti merupakan paket *voting reply*. Paket yang berisi IP dan MAC sumber tersebut akan ditambahkan ataupun di-*update* ke dalam tabel ARP *default* dan *mirror ARP table*. Hal tersebut diharapkan dapat mencegah terjadinya ARP *spoofing* secara berulang. Setelah itu, *host* akan kembali *listen* paket.

3.5 *Data Flow Diagram* (DFD)

Sistem ini melibatkan dua aktor yaitu *client host* itu sendiri dan *neighbor hosts* pada jaringan lokal. Masing-masing aktor saling berhubungan melalui satu entitas proses yaitu penggunaan metode ESV-ARP. Hal ini dapat ditunjukkan melalui diagram konteks pada Gambar 3.2.

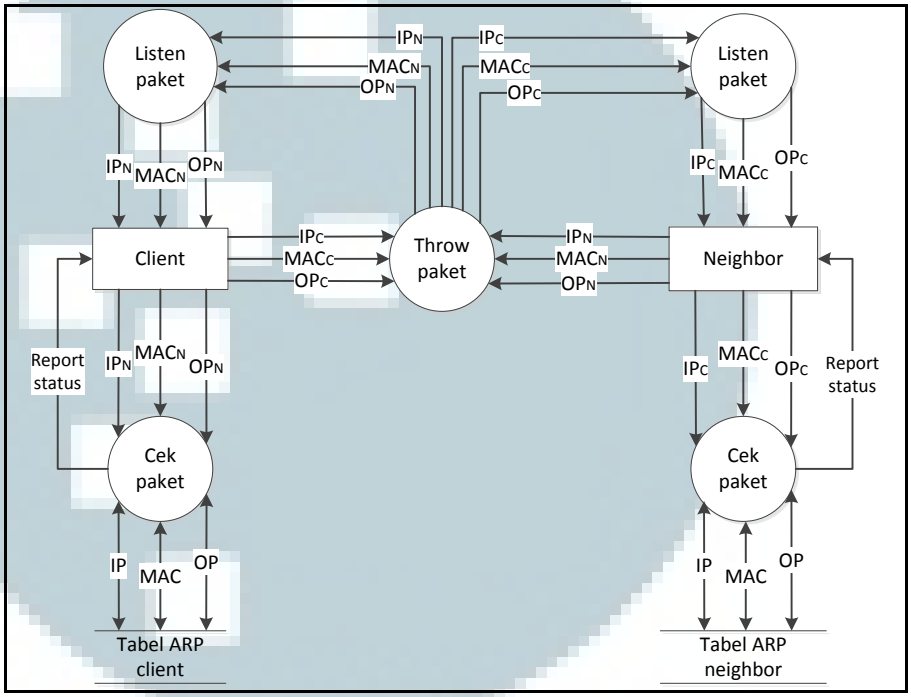


Gambar 3. 2 Diagram konteks sistem

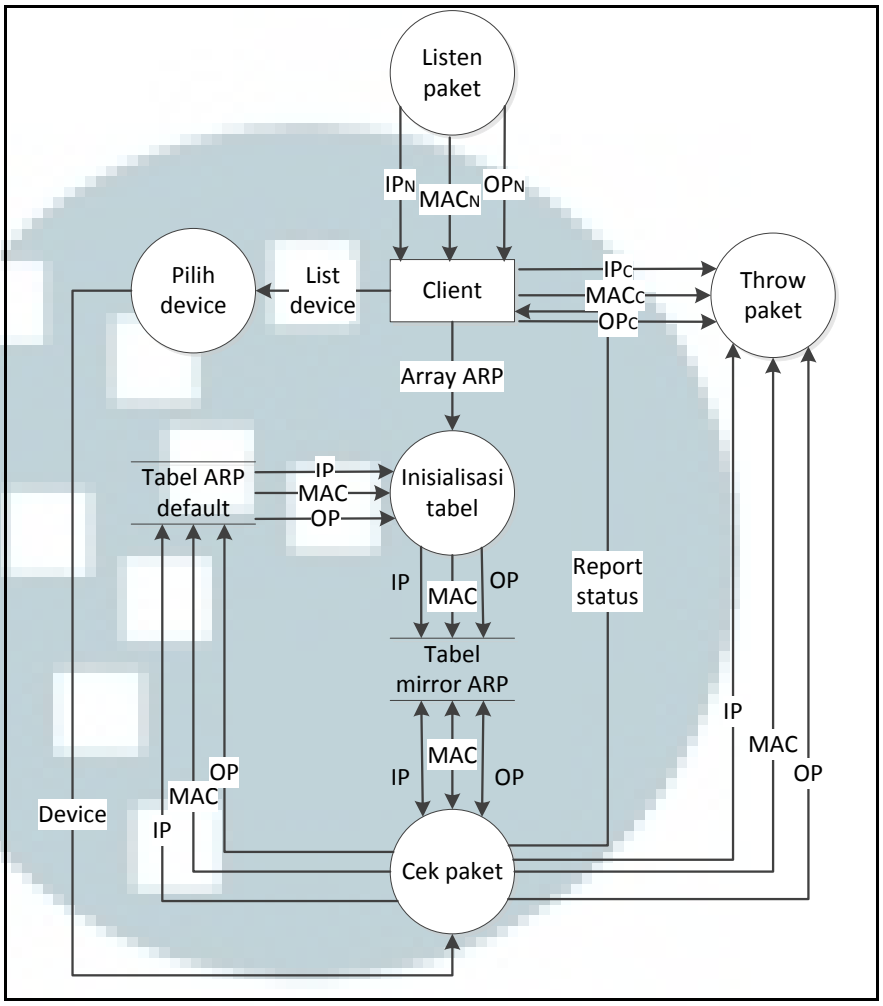


Gambar 3. 3 DFD level *zero* ESV-ARP

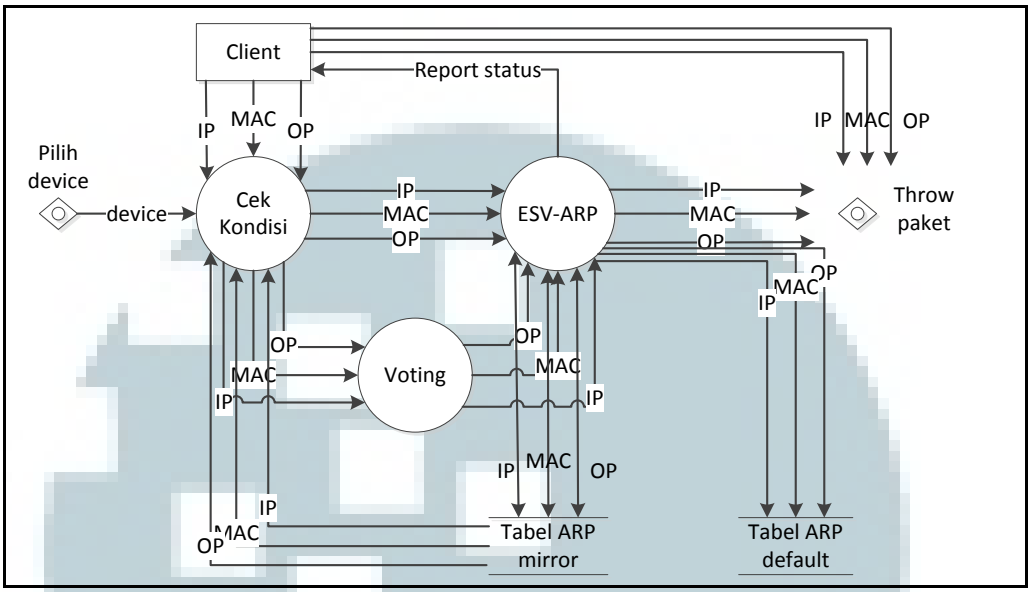
Terdapat tiga aspek utama yang dilibatkan agar metode ESV-ARP dapat diolah antara lain *Opcode*, IP, dan MAC. Setelah diolah, hasil dari proses yang sudah dieksekusi akan ditampilkan ke user.



Gambar 3. 4 DFD level 1 ESV-ARP



Gambar 3. 5 DFD level 2 ESV-ARP sisi *client*



Gambar 3. 6 DFD level 3 proses cek paket

Pada DFD level 1, *client* maupun *neighbor* memiliki proses dan data *store* yang sama kecuali isi dari data yang dibawa atau dikirim. Prosesnya antara lain *Listen* paket, *Throw* paket, dan Cek paket. *Listen* paket wajib dilakukan agar dapat melakukan cek paket. Sedangkan *Throw* paket juga harus dilakukan agar *neighbor* bisa melakukan *listen* paket layaknya yang dilakukan *client*. Ketika interaktor melakukan cek paket, data yang diambil untuk diolah merupakan *mapping* ARP yang berisi IP, MAC, dan *opcode* (OP) dari tabel ARP. Pada DFD level 2 inilah dijelaskan bahwa tabel yang digunakan sebanyak 2 yaitu tabel ARP *default* dan tabel ARP *mirror*.

Pada DFD level 2 yang ditampilkan pada gambar 3.5, tabel ARP *default* hanya digunakan untuk inisialisasi tabel ARP *mirror* yang nantinya

berfungsi untuk acuan pengolahan proses cek paket. Sementara itu, pada level ini memiliki proses tambahan yaitu proses Pilih *device* dan Inisialisasi tabel. *Device* yang dipilih menentukan *adaptor/client* mana yang akan dilakukan *listen* paket.

Pada DFD level 3 dijelaskan bagaimana cek paket tersebut bekerja. Proses yang dikerjakan antara lain proses cek kondisi, *Voting*, dan ESV-ARP. Dari gambar 3.6 digambarkan bahwa tidak semua paket yang telah diproses melewati tahap *Voting*. Hal ini bergantung pada kondisi paket itu sendiri, dan status paketnya apakah paket *request* atau *reply*.

3.6 Pembangunan Aplikasi

Hal pertama yang dilakukan adalah mencoba menangkap paket ARP yang datang menggunakan aplikasi Wireshark. Tujuannya adalah agar mengetahui bit yang mana saja yang nanti akan diolah untuk metode ESV-ARP. Setelah itu, dengan menggunakan aplikasi *console*, dicoba mengambil tabel ARP *host*. Proses ini membutuhkan `iphlpapi.dll`.

Selanjutnya, dengan menggunakan aplikasi *console*, dicoba untuk menangkap paket-paket yang datang pada *host* tersebut. Dalam hal ini dibutuhkan WinPcap agar bisa menangkap paket. Hal-hal yang mendukung pembangunan aplikasi ini dibutuhkan antara lain `PcapDotNet.Base.dll`., `PcapDotNet.Core.dll`,

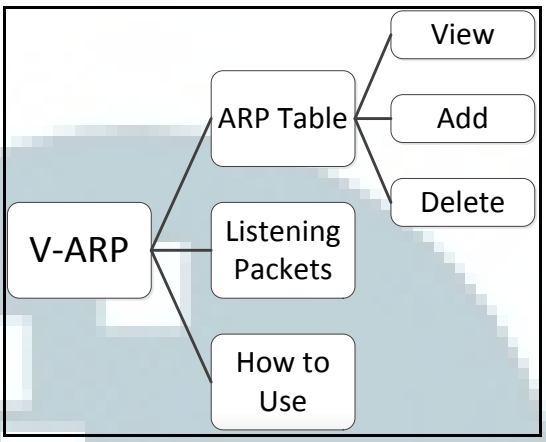
`PcapDotNet.Core.Extensions.dll`, `PcapDotNet.Packets.dll`. Setelah mendapatkan paket, paket tersebut diambil bit ke-21 agar bisa diolah menurut *opcode*-nya. Berdasarkan *opcode* tersebutlah yang menandakan apakah merupakan paket *request* atau *reply*.

Hal yang kemudian dilakukan adalah mengirimkan paket ARP baru yang telah dimodifikasi, yaitu mengganti *opcode default* dengan *opcode* eksperimental. *Opcode* 24 untuk paket *voting request*, dan *opcode* 25 untuk paket *voting reply*.

Terakhir, semua pembangunan di atas diimplementasikan ke dalam bentuk Windows Form supaya menjadi lebih *user friendly*.

3.7 Struktur Navigasi Menu

Aplikasi ini terdapat dua halaman *tab* yang memiliki fungsi berbeda. *Tab* pertama merupakan kontrol yang dioperasikan secara manual oleh *user* antara lain untuk menampilkan, menambahkan, atau menghapus entri ARP. *Tab* kedua yaitu untuk menjalankan metode ESV-ARP. Struktur navigasi menu pada aplikasi ini dapat digambarkan pada Gambar 3.7.



Gambar 3. 7 Struktur navigasi menu aplikasi

Menu utama pada aplikasi yaitu *tab* ARP Table. Rincian kedua *tab* dijelaskan sebagai berikut:

1. *ARP Table*

Pada halaman ini terdapat 3 tombol fungsi, antara lain:

a. *View*

Tombol *View* jika diklik akan menampilkan tabel ARP *default* yang telah di-*update*.

b. *Add*

Ketika tombol *Add* diklik, maka akan muncul *form* baru untuk mengisi input IP dan MAC yang akan disimpan pada tabel ARP *default*.

c. *Delete*

Untuk menghapus salah satu entri ARP, *user* harus memilih entri ARP yang ingin dihapus terlebih dahulu, kemudian tombol *Delete* diklik.

2. *Listening Packets*

Halaman ini akan menjalankan metode ESV-ARP dengan memilih *device* terlebih dahulu. Setelah itu *host* akan memproses paket-paket yang datang.

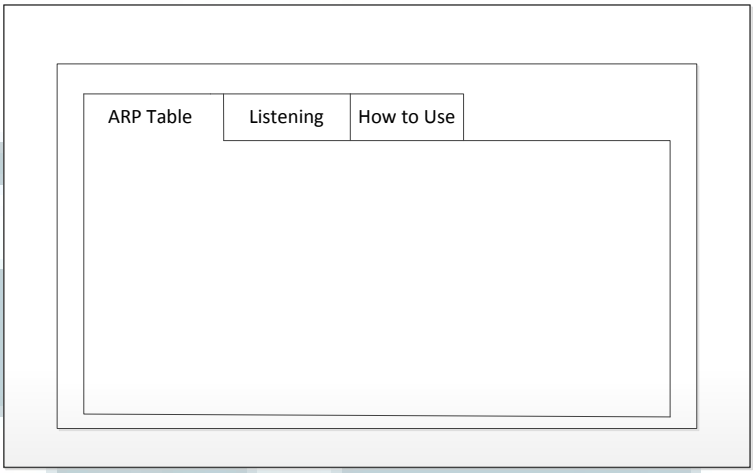
3. *How to Use*

Halaman ini menjelaskan cara-cara bagaimana menjalankan aplikasi ini pada tiap halaman *tab*-nya yaitu ARP *Table* dan *Listening Packets*.

3.8 Desain Antarmuka Aplikasi

Dalam pembuatan aplikasi ini dirancang terlebih dahulu desain antarmuka atau *user interface* sebagai acuan. Rancangan desain antarmuka aplikasi tersebut dapat digambarkan pada penjelasan seperti berikut:

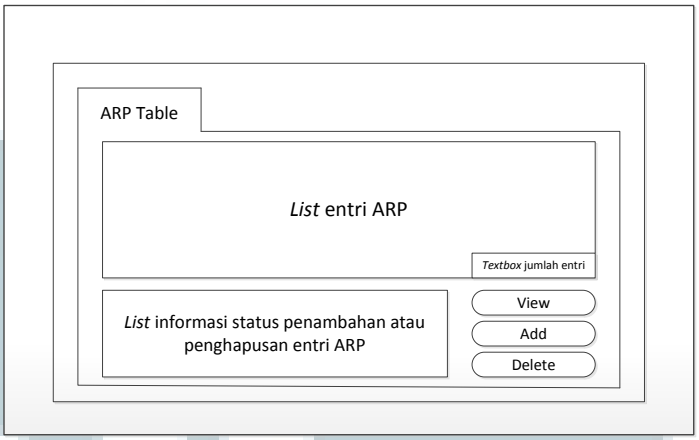
1. Menu Utama



Gambar 3. 8 Desain rancangan menu utama

Pada menu utama, terdapat 3 buah halaman *tab* yaitu *ARP Table*, *Listening*, dan *How to Use*. Menu *ARP Table* berfungsi untuk menampilkan dan memodifikasi tabel ARP *default*. Menu *Listening* berfungsi untuk menjalankan protokol ESV-ARP, sedangkan menu *How to Use* merupakan penjelasan cara menggunakan aplikasi.

2. Menu *ARP Table*

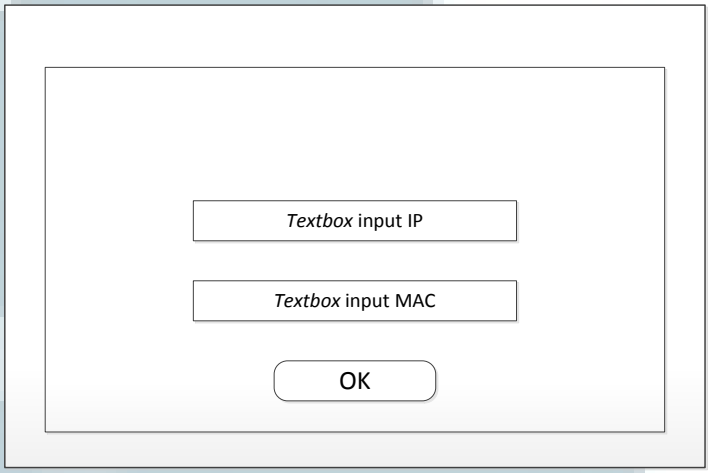


Gambar 3. 9 Desain rancangan menu ARP Table

Pada menu *ARP Table*, terdapat 3 *box* informasi diantaranya terdiri dari dua *listbox* dan satu *textbox*. Kedua *listbox* tersebut berfungsi untuk menampilkan informasi data entri tabel ARP *default* pada *device* dan informasi keterangan modifikasi entri ARP. Di pojok kanan bawah *listbox* entri ARP, terdapat *textbox* yang menampilkan jumlah entri ARP pada *device* tersebut.

Selain tiga *box* di atas, terdapat pula tiga tombol yaitu tombol *View*, *Add*, dan *Delete*. Tombol *View* berfungsi untuk menampilkan isi tabel ARP *default* yang paling baru (*up to date*) pada *device*. Tombol *Add* berfungsi untuk menambahkan entri ARP yang diinginkan pengguna. Ketika tombol ini diklik akan muncul *form* baru untuk mengisi alamat IP dan MAC yang ingin

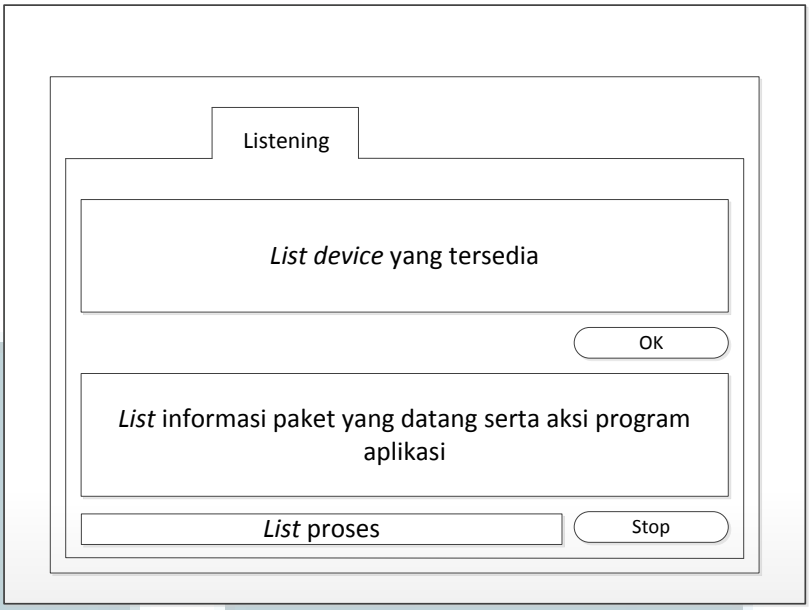
ditambah. *Form* pengisian alamat IP dan MAC ditunjukkan pada Gambar 3.10.



Gambar 3. 10 Desain rancangan menu penambahan entri ARP

Pada *form Add* terdapat dua *textbox* yang datanya diinput dari pengguna sendiri. Input data tersebut terdiri dari alamat IP dan alamat MAC. Setelah menginput kedua alamat pengguna harus menekan tombol OK agar data masuk dalam tabel ARP *default*. Ketika tombol OK diklik maka *form Add* akan "*close*" dan pengguna akan kembali ke menu *ARP Table*.

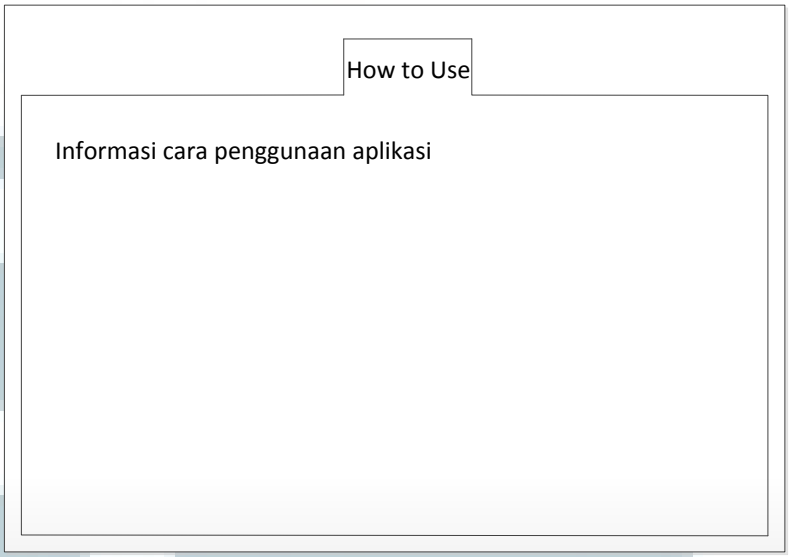
3. Menu *Listening* Paket



Gambar 3. 11 Desain rancangan menu *listening* paket

Dari Gambar 3.11 dapat dilihat bahwa halaman *tab* menu *Listening* memiliki dua buah *listbox* utama diantaranya *list device* yang tersedia dan *list* informasi paket yang datang berikut dengan aksi yang dilakukan. *List device* akan berisi *network adapter device* apa saja yang terdeteksi pada *device* tersebut. Di *list* inilah pengguna akan memilih *network adapter* manakah yang akan dilakukan *listening* paket serta menjalankan metode ESV-ARP. Jika pengguna ingin berhenti *listening*, pengguna akan memilih proses *listening* pada *list* proses kemudian tombol "*Stop*" diklik.

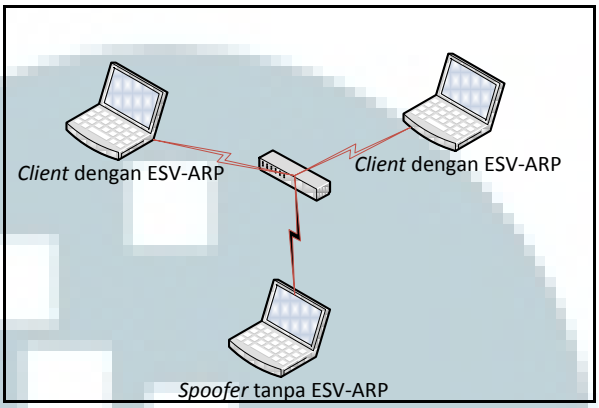
4. Menu *How to Use*



Gambar 3. 12 Desain rancangan menu *How to Use*

Gambar 3.12 menunjukkan bahwa pada menu ini tidak terdapat objek melainkan hanya sebuah *box* berisikan instruksi penggunaan aplikasi ESV-ARP.

3.9 Metode Pengujian



Gambar 3. 13 Ilustrasi penempatan aplikasi ESV-ARP

Semua *client* memiliki program aplikasi ESV-ARP, sedangkan *spoofer* tidak. Masing-masing komputer saling terhubung di satu jaringan lokal atau *Local Area Network* (LAN) yang tampak seperti pada Gambar 3.13. Namun, pada percobaan dilakukan secara virtual. Aplikasi ini akan dapat berfungsi jika komputer terkoneksi dengan jaringan lokal dan akan bekerja secara optimal jika dijalankan pada "Run as Administrator" karena adanya fitur penambahan dan penghapusan entri ARP. Langkah yang dilakukan dalam percobaan dijelaskan di bawah ini:

1. Komputer berbasis sistem operasi Windows 7 yang menyediakan aplikasi VirtualBox disiapkan.
2. Melalui VirtualBox, dua *Virtual Machine* (VM) di-*install* dengan basis operasi Windows 7.

3. Masing-masing VM ditambahkan .NET Framework 4.0 dan WinPcap. Tiap *network adapter* VM diatur menjadi *host-only* agar *host* dan VM menjadi satu jaringan.
4. Kedua VM ditetapkan sebagai *client* yang nantinya akan menjadi target (korban), sedangkan *host* menjadi penyerang dengan menggunakan aplikasi Nighthawk [10].
5. Sebelum menyerang, Nighthawk akan *scan network* untuk mendapatkan IP semua *client*. Setelah itu, target dipilih untuk memulai penyerangan.
6. Pada komputer *client*, proses pengolahan metode ESV-ARP dicatat dan dianalisis baik keberhasilannya maupun *performance*-nya.
7. Setelah sistem berhasil dijalankan, akan diuji kinerjanya dan akan dibandingkan dengan metode pada penelitian sebelumnya berdasarkan waktu CPU *instruction*. Uji coba kinerja akan dilakukan berdasarkan dua skenario yang berbeda yaitu ketika tidak ada *IP conflict* dan ketika terdapat *IP conflict*. Untuk mengetahui kinerja aplikasi digunakan program aplikasi JetBrains dotTrace Performance 5.5.3 [11]. Masing-masing percobaan akan dilakukan sebanyak 5 kali.