PENGAMANAN SCRIPT PHP DAN HTML DENGAN MENGGUNAKAN FORMAT .TAR PADA SISTEM OPERASI LINUX

TUGAS AKHIR

Disusun Oleh :

ARIEF KURNIAWAN KEMALSYAH
NRP. 5199 100 041

JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INFORMASI
INSTITUT TEKNOLOGI SEPULUH NOPEMBER
SURABAYA
2004

PENGAMANAN SCRIPT PHP DAN HTML DENGAN MENGGUNAKAN FORMAT .TAR PADA SISTEM OPERASI LINUX

TUGAS AKHIR

Diajukan Guna Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Komputer
Pada
Jurusan Teknik Informatika
Fakultas Teknologi Informasi
Institut Teknologi Sepuluh Nopember
Surabaya

Mengetahui / Menyetujui,

Dosen Pembimbing I	Dosen Pembimbing II
Febriliyan Samopa, M.Kom.	Royyana M Ijtihadie, S.Kom.
NIP. 132 206 858	

SURABAYA
JANUARI, 2004

ABSTRAK

Aplikasi Web atau Web base aplication, telah menjadi bagian yang tak terpisahkan lagi dalam kehidupan sehari-hari di era globalisasi ini. Karena aplikasi Web yang dapat memungkinkan jarak semakin sempit dan era informasi yang semakin berkembang, disamping itu jumlah pengguna internet yang semakin hari semakin meningkat, seiring dengan semakin meningkatnya pengguna atau user dari internet tersebut maka semakin meningkat pula masalah-masalah yang ditimbulkan secara teknis misalnya, semakin banyak pengguna internet maka menuntut aplikasi yang kita buat dapat lebih stabil dan aman. Dari alasan terahir (keamanan) aplikasi yang kita buat, seringkali menjadi alasan mengapa banyak user meninggalkan dunia internet, disamping itu masalah keamanan telah menjadi satu hal yang berdampak cukup krusial pada perkembangan dunia internet, timbulnya kejahatan-kejahatan didunia 'maya'. Dan dampak yang sering merugikan adalah pada dunia bisnis atau e-bussines.

Oleh sebab itu di perlukan proses modifikasi engine Apache yang memungkinkan suatu aplikasi dapat berjalan dengan tingkat kestabilan serta keamanan yang lebih baik. Pembuatan module TAR pada Apache web server dengan menggunakan algoritma XOR dan Caesar untuk enkripsi file PHP dan HTML yang telah di TAR. Serta penambahan module Gzip pada Apache server dapat membuat kecepatan akses server Apache lebih cepat, terutama pada jaringan yang padat.

Penggabungan ketiga module di atas, yaitu algoritma enkripsi, module TAR apache, dan module Gzip, dapat membuat server bekerja lebih efektif, yaitu dengan penambahan waktu yang tidak terlalu lama dapat menghasilkan sistem keamanan yang lebih baik.

KATA PENGANTAR

Jika ini dikatakan sumbangsih saya bagi ilmu pengetahuan dan umat manusia, maka hati saya akan menyangkal bahwa saya belum melakukan apa-apa, masih banyak hal yang belum saya lihat dan perbuat bagi umat manusia, akan tetapi ini adalah awal karya saya yang akan saya persembahkan untuk umat. Dengan di buatnya Tugas Akhir yang berjudul :

"PENGAMANAN SCRIPT PHP DAN HTML

DENGAN MENGGUNAKAN FORMAT .TAR

PADA SISTEM OPERASI LINUX "

Sebagai sebuah pintu yang memang harus saya lalui untuk menyelesaikan pendidikan Strata Satu (S-1) di jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember Surabaya.

Dalam penyusunan Tugas Akhir ini penulis berusaha menerapkan ilmu-ilmu yang telah didapatkan selama masa perkuliahan. Dan dengan selesainya Tugas Akhir ini penulis sangat berharap banyak orang akan memanfaatkannya.

Tidak ada manusia yang sempurna. Meski telah berusaha, dengan segala kerendahan hati penulis menerima saran dan kritik bila Tugas Akhir ini masih terdapat banyak kekurangan.

Surabaya, Oktober 2003

Penulis

UCAPAN TERIMA KASIH

Dengan selesainya Tugas Akhir, penulis mengucapkan terima kasih dan penghargaan yang sebesar-besarnya kepada :

1. **Allah SWT** sang radja manusia, yang atas rahmat dan hidayahnya, saya masih memilih Islam sebagai Agama saya, mengimani **Alquran** sebagai kitab saya, dan **Muhammad** sebagai Imam saya.
2. Orang tua yang ada di rumah yang telah bekerja keras dalam membesarkan penulis, dan menuangkan cinta yang besar kepada penulis, Papa **Mohammad Rasul** makasih atas segalanya, dan telah memilihkan mama **Leoni Murniati** sebagai bidadari yang sempurna bagi anak-anaknya, mudah-mudahan kita digolongkan menjadi hambanya yang bersyukur.
3. Bapak **Febriliyan Samopa, M.Kom** yang dengan baik hati telah memberikan waktu dan usahanya untuk membimbing penulis sehingga Tugas Akhir ini selesai dengan baik.
4. Bapak **Royyana M Ijtihadie, S.Kom** dengan penuh kesabaran memberikan waktu dan bimbingan kepada penulis untuk menyelesaikan Tugas Akhir ini, dan terima kasih telah menjadi kakak, rekan main bola dan bahkan teman bercanda, terima kasih Mas atas semuanya, mudah-mudahan Allah selalu bersama kita.
5. Bapak **Ir.FX Arunanto Msc**, sebagai dosen wali, serta Dosen-dosen jurusan Teknik Informatika ITS atas segala hal yang telah diberikan selama masa perkuliahan.

6. **Ir.M Husni M.Kom**, **Wahyu S M.Kom**, **Imam Kuswardayan S.kom** selaku Dosen penguji, atas kritik dan saran yang di berikan kepada penulis.
7. Mas **Faizal Johan**, salah satu dosen favorit penulis, tetep semangat pak, terima kasih nasehat-nasehatnya.
8. Kedua orang bidadariku, **Citra Hayati Cahya Rani** dan **Devi Fatin Adillah**, makasih udah jadi adek yang baik, dan pemakluman atas waktu yang hilang dalam proses pengerjaan tugas akhir ini. Adinda **Bahtiar Murti Muhammad** (alm), mudah-mudahan Allah menyediakan tempat yang terbaik untukmu.
9. **Bapak-bapak Pejabat korup, atas welas kasihnya membiarkan kami menghirup udara "kotor" Indonesia dengan GRATIS.**
10. Seorang "**hawa**" yang lirikan matanya mampu merubah pandangan hidupku dan membuat sadar bahwa cinta tak harus memiliki (kamu telah menunjukkan arti hidup ini).
11. Teman-teman **HMTC dan HMTCp** maaf ternyata energi saya sudah habis, revolusi belum berakhir teman. Bahkan iklim ilmiah pun belum mulai menebarkan harumnya di kampus ini.
12. Orang-orang yang mengatur dunia, walau banyak orang bilang kita gak punya kehidupan, tapi persetan dengan mereka kay! **Dhion**, tetep mencari sendiri ya, jangan mengandalkan ortumu, **Kholimi**, aku pasti akan merindukanmu (jangan tinggalkan hima dulu, dia masih butuh komitmenmu), **Kacung liga**, walaupun persahabatan nomer satu, tetapi

kalo sudah jodoh mau apa (bete dheeeh), **monty anton** kembali ke himpunan dong ☹, **Kentang lintang**, walau kita sering "berselisih" tapi itu hanya bercandaan kok, dan adikku yang selalu menasehati, **Angga**, kejar cita-citamu ya (sebelum terdengar tangisan bayi, berarti kesempatan masih ada dek), **pak Dhe**, ojo ngilang mane yen AC hima mati, cepet kelar ojo proyekan ae.

13. Teman-teman FC.Informatika, gak tahu lagi deh gimana aku melalui hari-hari senggang tanpa main bola, dan gak tahu lagi deh FC.Informatika tanpa pemain seperti aku, **Rai play boy** (cepet insaf dan tobat), **Indie** (hmm... cinta memang gak harus memiliki), **Germos** (aku belum bisa menemukan orang yang berdedikasi penuh sepertimu, aku salah satu pengagummu mas), **Aw** (jangan kumat lagi play boynya ya), **Diqi** (nitip arek-arek hima ya mas), **Mbah Darwan, ADP, Mardun** (himamu urusen ojo pacaran thok) aduh semua deh.
14. Teman-teman Buenger's, hidup memang mengesalkan teman (**Napis, Inuk, Rontog, gumon, Kingkong, Erzu, Moon, rahmad**) tetep buenga wae rek yo. Dan acara tahun baruan tahun depan, aku gak usah di hubungi, aku wis ngerti bakalan di tinggal kok.
15. Teman-teman AOE'rs **mbah ajun**, **arman** (kamu lebih sexy pake rambut panjang, jangan di potong ya), **kecu** (kapan create maneh), **iwan**, **yudha00** (berhenti dari kebiasaan burukmu dek, wanita bukan untuk di obong-obong), **nope** (persahabatan nomer satu ok), **ami**(double nn, penyesalan

datang terahir kawan), **gandring**, **amin rais** mudah2an kutukanmu berakhir dan semua deh.

16. Para *cadangan besi* HMTC arek-arek 2002 **knip**, **vicki**, **huda**, **ciman**, **fauzan02**, **zen02**, **doraemon**. Tanggung jawab kalian bukan hanya pada regenerasi AOE'rs tapi juga pada HMTC dan HMTCp yang tercinta.
17. **Kul-kul atau bajimut** aku belajar banyak pada sosok melankolis selama satu bulan kita "hidup bersama" di jakarta (kecuali kebiasaan mbangkong), **Erwan** komting terima kasih udah jadi penunjuk bagi arek-arek 99, **Mbik**, **Rifqi, Eriyanto, Anna kholil, Adeta, Meme** dan terima kasih yang sebesar-besarnya bagi seluruh keluarga besar C0F atas cinta yang kalian berikan.
18. **Mediana Aryuni** beserta keluarga, terima kasih banyak atas dukunganmu di saat-saat terakhir, tetep semangat ya med!
19. Warga Lab-Prog **Joko**, **Gunna**, **Kendy**, **Rizky**, **Aby** (terima kasih sudah mau jadi dosen pembimbing ke 3), **Jaka** dan Admin laennya, atas nama orang-orang yang menggunakan fasilitas lab, yang belum sempat mengucapkan terima kasih, saya mewakili mereka mengucapkan **terima kasih yang amat sangat besar atas dedikasi kalian**.
20. **Warga Lab AJK**, mas **paimo, kamas** dan laen-laen yang tidak dapat saya sebutkan semua, atas waktu konsultasi yang di berikan.**Warga Lab SI**, **budinug**, **warna agung** dan teman-teman yang ada di sana.
21. **Pak Yudhi, pak narno, pak sholeh, pak koding, pak sugeng** dan semua pegawai administrasi FTIf sukses selalu.

22. Teman-teman Aktivis **KAMI, JMMI, SITC** dan teman-teman aktivis pusat, mantan SC Bakti kampus 2002, **pipit** TS99, **Sigit** GEODESI99, **Tirex** GEODESI00, **Lutfi** MATH00, **Dheyan** STAT00, **Black** KAPAL98, **Teno** MATH98, **kaspo** TFIS00 dan rekan-rekan semua aktivis kampus.
23. Cintaku di bangku kuliah, komputerku tercinta **Pitri**(kucing garong) dan **Pipi**(kucing persia) aku sayang kalian. Ruang **HMTC** TC-221 atas kehangatan dan kasih sayang yang di berikan, Aku tak akan bisa seperti ini tanpa kalian.
24. Teman-teman lain yang tidak dapat penulis sebutkan satu persatu yang turut memberikan kontribusi atas selesainya Tugas Akhir ini, dan orang-orang yang pernah hadir dalam hidupku dan **mewarnai proses kedewasaanku**.
25. Last but not least, terima kasih pada **DIRIKU** sendiri, yang telah mampu menyelesaikan tugas akhir ini, perjalanan masih jauh kawan jangan dulu berbangga diri. Mudah-mudahan kamu mendapat cukup cobaan untuk membuatmu tegar, cukup kasih sayang untuk membuatmu bahagia, cukup harta untuk mengingat yang lemah, dan cukup kekuatan untuk memperjuangkan keadilan Allah.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

BAB I

PENDAHULUAN

Dalam bab ini akan dijelaskan latar belakang, permasalahan yang diangkat, tujuan dan manfaat yang dapat diberikan oleh penulisan Tugas Akhir ini. Batasan masalah, metodologi pembuatan dan sistematika pembahasan Tugas Akhir juga akan dibahas dalam bab ini.

1.1. LATAR BELAKANG

Aplikasi Web atau *Web base aplication*, telah menjadi bagian yang tak terpisahkan lagi dalam kehidupan sehari-hari di era globalisasi ini. Karena aplikasi Web yang dapat memungkinkan jarak semakin sempit dan era informasi yang semakin berkembang, disamping itu jumlah pengguna internet yang semakin hari semakin meningkat, seiring dengan semakin meningkatnya pengguna atau user dari internet tersebut maka semakin meningkat pula masalah-masalah yang ditimbulkan secara teknis misalnya, semakin banyak pengguna internet maka menuntut aplikasi yang kita buat dapat lebih stabil dan aman. Dari alasan terahir (keamanan) aplikasi yang kita buat, seringkali menjadi alasan mengapa banyak user meninggalkan dunia internet, di samping itu masalah keamanan telah menjadi satu hal yang berdampak cukup krusial pada perkembangan dunia internet, timbulnya kejahatan-kejahatan di dunia 'maya'. Dan dampak yang sering merugikan adalah pada dunia bisnis atau e-bussines.

Aplikasi menggunakan PHP dan HTML yang sudah tak dapat di ragukan lagi ke handalannya terutama kestabilan yang menjadi alasan banyak orang lebih memilih PHP sebagai bahasa pemrograman untuk aplikasi web, di samping bahasa yang di gunakan lebih mirip dengan bahasa C, yang memungkinkan dapat memudahkan pengguna untuk eksodus ke bahasa PHP. Di samping itu PHP engine dapat dengan mudah di dapatkan secara free atau gratis, dan source code yang di buka secara bebas dapat memungkinkan orang-orang memodifikasi atau melakukan penyempurnaan pada engine tersebut.

Oleh sebab itu proses modifikasi engine yang memungkinkan suatu aplikasi dapat berjalan dengan tingkat kestabilan serta security yang lebih handal dapat dibuat dan di ciptakan.

1.2. TUJUAN TUGAS AKHIR

Tujuan pembuatan Tugas Akhir ini adalah untuk membuat sebuah tool pembantu / software pembantu untuk membuat script PHP dan HTML lebih aman dengan menggunakan algoritma tar dan kemudian hasil tar script PHP dan HTML tersebut di *encrypt* dan dengan memodifikasi engine Apache sehingga browser yang request ke server engine akan mengeksekusi file tar yang di *encrypt* tersebut, dan setelah file hasil *encrypt* tersebut di eksekusi oleh Apache maka hasil eksekusi akan di zip menggunakan algoritma gunzip(gz) dengan menggunakan mod_gzip.

1.3. PERMASALAHAN

Permasalahan yang diangkat dalam Tugas Akhir ini adalah:

1. Bagaimana merancang dan membuat suatu aplikasi yang dapat menghasilkan suatu aplikasi PHP dan HTML yang lebih aman.
2. Bagaimana server membaca URL pada browser yang di inputkan client.
3. Bagaimana mengirimkan hasil eksekusi file PHP dan HTML pada server Apache ke Client menjadi lebih cepat / menghemat bandwith.
4. Bagaimana menciptakan suatu aplikasi yang mempunyai perbandingan yang sesuai antara keamanan dan kecepatan.

1.4. BATASAN PERMASALAHAN

Aplikasi Case Tool yang akan dirancang dalam Tugas Akhir ini memiliki batasan-batasan sebagai berikut:

1. Aplikasi yang dihasilkan dari tugas akhir ini adalah aplikasi yang berbasis web.
2. Aplikasi yang dibangun dapat memberikan suatu teknik pengamanan script pada aplikasi web yang menggunakan script PHP dan HTML.
3. Aplikasi yang dibangun berupa suatu module apache server untuk mengeksekusi script PHP dan HTML yang telah archieve dan di *encrypt*.
4. Tidak menangani pengeksekusian Include file pada PHP.
5. Tools yang di gunakan untuk pembuatan aplikasi ini adalah :

   a. Sistem Operasi Linux Mandrake 8.

   b. GCC gnu Linux

   c. Apache Web Server.

1.5. TINJAUAN PUSTAKA

Aplikasi Web dengan menggunakan PHP dan HTML sudah tidak asing lagi, terutama bagi orang-orang yang bergelut dengan aplikasi berbasis web, namun sering kali penggunaan aplikasi berbasis web ini mempunyai sistem pengamanan yang kurang memadai, oleh karena itu pembuatan aplikasi web seringkali mengalami pengerusakan (hack), di karenakan script aplikasi dapat di lihat dan di pelajari, sehingga kekurangan dari sistem dapat di pelajari dan dari titik lemah tersebut para hacker dapat merusak atau mengganggu sistem kita.

1.6. MANFAAT TUGAS AKHIR

Manfaat pertama yang didapat dengan dibuatnya aplikasi *case tool* ini adalah keamanan script yang kita buat sehingga script yang akan di eksekusi oleh server akan lebih aman dan hanya orang yang memilikinya yang dapat melihat dan mengedit script ini, misalnya pada aplikasi web PHP menggunakan web server Apache, maka orang-orang yang berada pada satu engine dengan anda dapat melihat script yang anda miliki, dan apabila dia memiliki niat buruk maka kelemahan-kelemahan pada sistem anda dapat di mamfaatkan untuk merusak sistem anda.

Aplikasi hasil ini juga dapat menghasilkan suatu server engine yang lebih optimal terutama dalam hal pemamfaatan bandwith dimana file hasil eksekusi server apache, akan di kompres menggunakan aplikasi gunzip, dan di kirim ke client sehingga masalah bandwith yang seringkali menjadi masalah dapat di atasi, dan hasilnya akan di ungz di client.

1.7. METODOLOGI TUGAS AKHIR

Metodologi yang digunakan dalam pengerjaan dan penyelesaian Tugas Akhir ini adalah sebagai berikut :

1.7.1. Studi Literatur

Studi literatur yang dilakukan adalah mempelajari konsep dan teknologi yang akan digunakan untuk membangun aplikasi tersebut. Selain itu juga mempelajari software yang sudah ada yang akan dijadikan acuan dalam pembuatan Tugas Akhir. Informasi tersebut diperoleh dengan membaca literatur ataupun jurnal-jurnal yang berhubungan.

1.7.2. Perancangan Perangkat lunak dan desain sistem

Pada tahap ini akan dilakukan analisa dan desain terhadap sistem, berdasarkan hasil yang telah diperoleh dari tahap 1. acle dan lain-lain.

1.7.3. Pengembangan Aplikasi

Pada tahap ini dilakukan perancangan dan pembuatan perangkat lunak dengan menggunakan konsep dan teknologi yang telah didapat pada Metodologi Pembuatan Tugas Akhir pada nomor 1. Dengan GCC Linux dan Apache Linux server.

1.7.4. Ujicoba Dan Evaluasi

Pada tahap ini aplikasi sudah selesai dan akan dievaluasi dengan menggunakan aplikasi web PHP dan HTML yang bervariasi dan diperbaiki demi kelayakan software dan keberhasilan software sesuai dengan tujuannya dibangun.

1.7.5. Revisi Sistem

Beberapa perbaikan serta penyempurnaan terhadap aplikasi yang telah dibuat.

1.7.6. Penulisan Naskah

Penulisan naskah yang meliputi latar belakang sistem, dasar teori dan metode yang digunakan dalam Tugas Akhir ini.

Disusun buku sebagai dokumentasi dari pelaksanaan Tugas Akhir. Dokumen ini juga diharapkan dapat berguna bagi orang lain yang memiliki keinginan untuk mengembangkan sistem tersebut lebih lanjut.

1.8 SISTEMATIKA PENULISAN

Buku Tugas Akhir ini dibagi dalam beberapa bab dengan sistematika penyusunan sebagai berikut :

BAB I Pendahuluan

Membahas juga teknologi-teknologi yang mendukung aplikasi *case tool* berjalan di internet dan latar belakang serta manfaat dari dari *case tool* yang dibuat.

BAB II Dasar Teori

Menjelaskan teknologi-teknologi dalam pengembangan aplikasi internet serperti Apache Web server, tools-tools GCC serta aplikasi linux yang menunjang dan algoritma tar.

BAB III Perancangan Perangkat Lunak

Perencanaan dan perancangan algoritma, struktur aplikasi, diagram alur perangkat lunak, serta langkah-langkah pembuatan dan proses pembuatan.

BAB IV Implementasi Perangkat Lunak

Mengimplementasikan rancangan perangkat lunak.

BAB V Uji Coba dan Evaluasi

Uji coba dan evaluasi terhadap sistem perangkat lunak yang dibuat serta penyempurnaan aplikasi jika masih terdapat banyak kekurangan.

BAB VI Penutup

Kesimpulan dan saran untuk pengembangan aplikasi lebih lanjut.

BAB II

DASAR TEORI

BAB II

DASAR TEORI

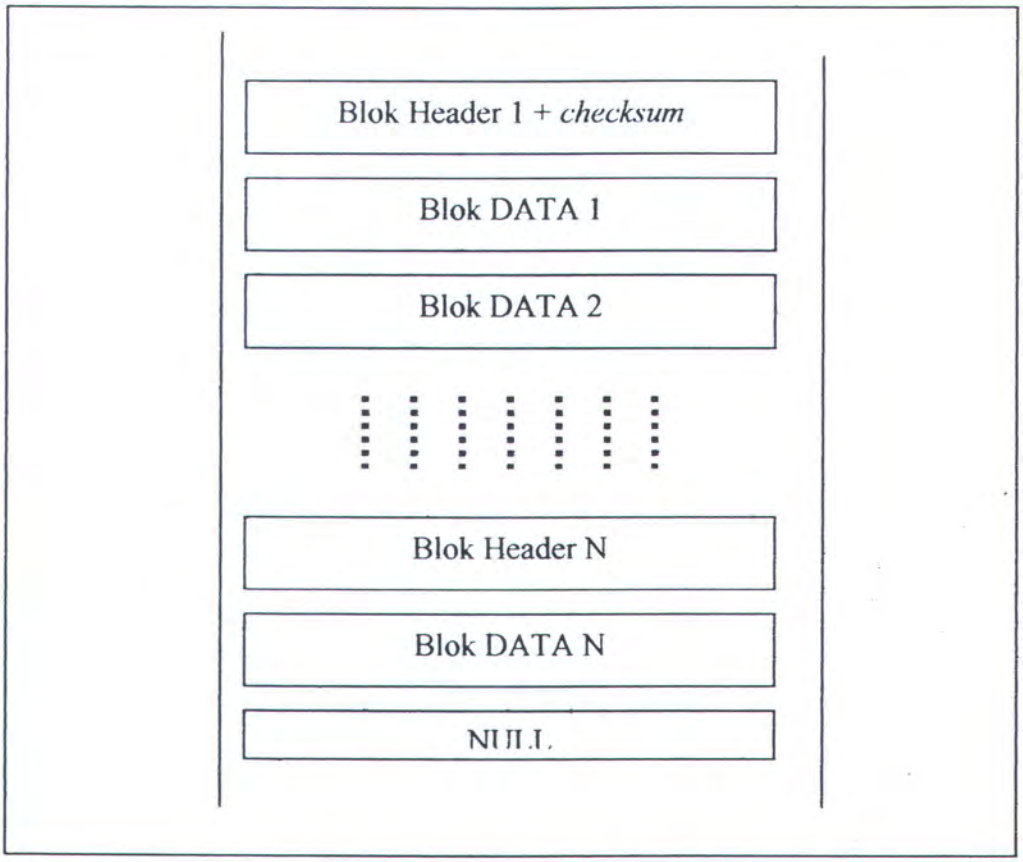
2.1 Tar Archive File

Pada awalnya, tar archive digunakan untuk menyimpan file pada tape magnetik. Nama `tar' berasal dari kegunaan ini, dimana mewakili tape archiver. Di samping nama kegunaannya, tar dapat mengarahkan output-nya ke alat yang tersedia, file-file, atau program lainnya (dengan penggunaan pipe).

Program tar biasa digunakan untuk membuat dan memanipulasi tar archives. Archive adalah sebuah file yang berisikan lebih dari satu file didalamnya, dan masih menyimpan informasi-informasi seperti nama file, *owner(s)* file, and dan informasi lainnya. (Sebagai tambahan, archive merekam hak akses, user dan grup, ukuran dalam bytes, dan waktu modifikasi terakhir. Beberapa archive juga merekam nama-nama file di dalam tiap-tiap direktori archived, sama seperti halnya file lainnya dan informasi direktori). Tar dapat digunakan untuk membuat archive baru di dalam direktori yang dispesifikasi.

Format file tar, mengidentifikasi file tar tidak hanya berdasarkan ekstensi filenya. Setiap header file TAR juga menyimpan *checksum* di dalamnya. maka, dengan membaca 512 bytes pertama dari TAR file, untuk memeriksa apakah file tersebut adalah file TAR atau tidak, yaitu dengan cara membuat *checksum* dan membandingkan dengan *checksum* yang ada didalam file.

Ukuran file sebesar 512 bytes, pada header menyimpan name, size, tanggal terakhir file di modifikasi, hak akses dan yang lainnya. Panjangnya penamaan hanya 100 karakter (termasuk pathnya).



Gambar 1.1 Format file TAR

Masing-Masing file tar adalah suatu urutan 512 byte masing-masing. Masing-Masing blok adalah yang bisa jadi header blok atau blok data. Blok header dikenali oleh adanya string unik "ustar" pada bidang yang unik. Blok berikut memegang data dari file. Data terakhir yang memisahkan suatu file adalah nol di mampatkan ke 512 bytes, walaupun banyak program yang mengekstrak file archive harus mengecek ukuran field dalam headernya dan

memotongnya pada ukuran yang sesuai sub-directories disimpan di dalam archive, mereka berada pada header tunggal tanpa blok data yang berikut. Sub-Directories dapat dikenali oleh kehadiran ascii ' 5' di sana. sub-directories disimpan sebagai file reguler, field nama memberi alur kepada file sehubungan dengan root dari arsip. Menguraikanlah nama file satu-satunya cara untuk menentukan sub-directory yang (mana) suatu file berada.

Untuk berbagai alasan yang mungkin bertalian dengan pembatasan tigapuluh tahun perangkat keras tua dan kebutuhan untuk menyimpan hal-hal yang penting tetap compatible, semua angka-angka disimpan sebagai string, dalam octal. Sebagai contoh, jika ukuran file adalah 1024, . ukuran yang di catat " 00000002000". Semua nomor, jumlah disandikan dalam cara ini, dengan suatu terminator null dalam tempat terakhir dan nol lapisan jika dibutuhkan. Ini meliputi uid, gid (user dan group id), dan ukuran field.

Model fieldnya juga disimpan dalam octal, yang mana menempatkan schema ini menjadi mudah. Mengumpamakan field dinamai , mode[7] memegang suatu terminator nulll. owner, group, dan public permission disimpan di dalam mode[4], mode[5], dan mode[6] berturut-turut, mengikuti pada umumnya chmod nilai semantik . Suid, guid, dan bit di dalam mode[3], dan bit mode[2] yang tinggi adalah bit direktori.

File TAR dapat menyimpan secara bergantian header dan file data. Header disimpan sebesar 512 byte, di ikuti dengan nol (zero) atau beberapa blok lagi dengan ukuran masing-masing sebesar 512 bytes (berisi file data yang tidak terkompres). Jika file archive tersebut mempunyai ukuran (dalam byte) yang tidak

dapat di bagi dengan 512 byte, maka byte sisanya akan di letakkan pada blok data berikutnya (blok terakhir). File TAR yang mempunyai ukuran sebesar 513 byte, maka di butuhkan 512 byte untuk header, dan 513 byte untuk file data, dan 511 byte tambahan sehingga file TAR tersebut dapat di kalikan dengan 512 byte. Tar di desain untuk penyimpanan tape yang dapat di akses hanya tiap blok dalam satu waktu. 512 byte (setengah kilobyte) adalah ukuran standart untuk satu blok.

Header file tar terdiri dari berbagai macam informasi dan di letakkan dalam bentuk struct,

```
Layout of tar header blocks
struct posix_header
{
  char name[100];
  char mode[8];
  char uid[8];
  char gid[8];
  char size[12];
  char mtime[12];
  char chksum[8];
  char typeflag;
  char linkname[100];
  char magic[6];
  char version[2];
  char uname[32];
  char gname[32];
  char devmajor[8];
  char devminor[8];
  char prefix[155];
};
```

Gambar 2.2 Struct Header File TAR

File-file di dalam sebuah archive disebut dengan member. Istilah file digunakan untuk menunjukkan hanya file-file yang dapat diakses secara normal (yakni ls, cat, dan sebagainya), dan istilah member untuk menunjukkan hanya pada member dari sebuah archive. Demikian pula, sebuah nama file adalah nama dari sebuah file, dimana terletak di filesystem, dan nama member adalah nama dari sebuah member archive di dalam archive.

Istilah extraction menunjukkan proses penyalinan sebuah member archive member (ataupun multiple members) menjadi sebuah file di dalam filesystem. Proses meng-ekstrak semua member dari sebuah archive disebut dengan *extracting* archive. Sedangkan istilah unpack dapat digunakan untui menunjukkan *extraction* dari beberapa atau semua member dari sebuah archive. *Extracting* archive tidak akan merusak struktur dari archive, namun sama seperti membuat sebuah archive tidak akan merusak salinan dari file-file yang ada di luar archive. Selain itu dapat juga dengan mencatat daftar member di dalam archive yang diberikan (dimana hal ini sering dianggap sebagai hasil cetakan member-member tersebut menjadi output standar, atau command line), atau menambahkan member-member ke pre-existing archive. Semua operasi-operasi tersebut dapat dilakukan dengan menggunakan tar.

Program tar menyediakan kemampuan untuk membuat tar archives, seperti halnya berbagai macam manipulasi lainnya. Sebagai contoh, tar dapat digunakan pada archieve yang telah dibuat sebelumnya untuk meng-ekstrak file, untuk menyimpan file tambahan, atau untuk meng-*update* atau membuat daftar file yang telah disimpan.

Tar archives dapt digunakan di berbagai cara, antara lain misalnya: *storage*, *backup*, and transportasi.

Storage

Pada umumnya tar archive digunakan untuk menyimpan file-file yang berkaitan dengan transfer file di atas suatu jaringan. Sebagai contoh, proyek GNU mendistribusikan berkas perangkat lunaknya menjadi tar archives, sehingga semua file-file yang berkaitan dengan program tertentu (atau sekumpulan program-program terkait) dapat ditransfer sebagai sebuah unit yang tunggal.

Sebuah pita magnetik dapat menyimpan beberapa file secara berurutan. Namun, pita magnetik tersebut tidak memiliki nama untuk file-file tersebut; pita magnetik hanya mengetahui posisi relatif pada pita tersebut. Satu cara untuk menyimpan beberapa file pada satu pita dan menyimpan namanya adalah dengan membuat sebuah tar archive. Bahkan pada saat mekanisme transfer dasar dapat menyimpan pelacakan nama-nama, sebagaimana yang dapat dilakukan oleh FTP, gangguan penanganan multiple files, direktori, dan berbagai multiple links membuat tar archives menjadi bermanfaat.

Archive files juga digunakan untuk penyimpanan dalam jangka waktu yang lama. Sehingga dapat disimpulkan bahwa tar dapat digunakan dalam semua dimensi, bahkan waktu.

Backup

Oleh karena archive yang dibuat oleh tar mampu menyimpan informasi file dan struktur direktori, tar biasanya digunakan untuk melakukan backups disk yang menyeluruh. Suatu backup meletakkan sekumpulan file (yang mungkin

digunakan oleh banyak para pemakai dan proyek) ke dalam suatu disk atau suatu tape. Hal ini menjaga kerusakan informasi dalam file itu . GNU tar mempunyai fitur khusus yang mmungkinkan untuk membuat dumps yang meyeluruh dari semua file di dalam suatu filesystem.

Transportasi

Archive dapat dibuat pada satu buah sistem, mentransfernya ke sistem yang lain dan meng-ekstrak isinya ke sistem yang lain tersebut. Hal ini memungkinkan untuk transportasi sekumpulan file-file dari satu sistem ke sistem lainnya.

2.2 Apache Web Server

Web server atau lebih tepatnya world wide web server adalah server internet yang mampu melayani koneksi transfer data dalam protocol HTTP(hypertext transfer protocol). Web server saat ini merupakan inti dari server-server internet selain e-mail server, ftp, dan news server. Hal ini dapat di maklumi karena web server telah di rancang untuk dapat melayani beragam jenis data, mulai dari text, hypertext, gambar (image), suara, gambar tiga dimensi, plug-in dan sebagainya. Keunggulan ini membuat web dapat di terima tidak saja di kalangan institusi universitas, melainkan hamper semua perusahaan komersial saat ini telah memiliki sebuah atau beberapa web server sekaligus di internet.

Web server pada umumnya melayani data dalam bentuk file HTML (hypertext markup language). Dari file ini kemudian dapat di kaitkan ke file

HTML lainnya, ke file gambar, ke file suara, dan segala jenis file komputer yang hendak di publikasikan di Internet.

Web server bukan hanya dapat melayani file-file yang ada di dunia Internet. Web server juga dapat di kombinasikan dengan dunia mobile wireless Internet. Dengan menggabungkan web server dengan sebuah WAP (wireless access protokol) gateway, maka jadilah web server ini sebuah WAP server yang siap melayani akses mobile Internet pada handphone-handphone yang memiliki fitur WAP. Dalam konteks ini, web server tidak lagi melayani data file HTML, tetapi file-file WML (wireless markup language).

Apache merupakan web server yang paling banyak di gunakan di internet saat ini, hal ini di sebabkan oleh beberapa faktor seperti kecepatan, performansi, dan tentu saja harga yang gratis, untuk melihat perbandingan Apache dengan web server lainnya dapat di lihat URL http://webcompare.Internet.com/chart.html atau URL http://www.netcraft.com/Survey/.

Untuk menggunakan Apache web server, terlebih dahulu kita harus menjalani proses instalasi, dan tahap-tahap proses instalasi berbeda, tergantung sistem operasi yang akan digunakan, setelah proses instalasi, sebaikknya meluangkan waktu untuk membaca dokumentasi yang tersedia. Masukkan alamat http://www.apache.org

2.3 Apache DSO

2.3.1 Latar Belakang

Pada turunan Unix modern ada suatu mekanisme menarik yang pada umumnya disebut dinamis linking / loading dari Dynamic Shared Objects(DSO) yang menyediakan suatu cara untuk membangun suatu potongan kode program dalam suatu format khusus untuk pemuatan program tersebut pada run-time ke dalam ruang alamat dari suatu program executable.

Pemuatan ini pada umumnya bisa dilakukan dengan dua cara: Secara otomatis oleh suatu program sistem bernama ld.so manakala suatu program executable dimulai atau secara manual dari dalam pelaksanaan program melalui interface sistem yang terencana ke Unix loader melalui perintah sistem dlopen()/dlsym().

Pada cara yang pertama DSO pada umumnya disebut shared librabries atau DSO libraries dan diberinama libfoo.so atau libfoo.so.1.2. Mereka terletak pada suatu direktori sistem (biasanya / usr/lib) dan penghubungan pada program yang executable dibentuk pada build-time dengan menetapkan - lfoo pada perintah penghubung.

Hard-codes library ini mengacu ke file program yang executable sehingga pada start-time Unix loader bisa menemukan libfoo.so di / usr/lib, di hard-coded path melalui linker-options seperti -R atau di path yang telah diatur melalui environtment variabel LD_LIBRARY_PATH. Library itu kemudian memecahkan (yang belum terpecahkan) simbol-simbol di dalam program yang executable yang terdapat dalam DSO itu.

Simbol-simbol dalam program yang executable pada umumnya tidak disesuaikan oleh DSO (sebab DSO adalah reusable library dari kode yang bersifat umum) dan karenanya tidak ada pemecahan lebih lanjut yang harus dilaksanakan. Program yang executable tidak harus bertindak sendiri untuk menggunakan simbol dari DSO sebab pemecahan yang lengkap dilaksanakan oleh Unix loader. (Sebenarnya, kode untuk memasang ld.so adalah bagian dari run-time startup kode yang yang terhubung ke tiap-tiap program executable yang terbatas non-statis). Keuntungan dynamic loading dari kode library umum sudah jelas: kode library hanya perlu disimpan sekali, dalam sistem library seperti libc.so, menghemat ruang disk untuk setiap program.

Pada cara yang kedua DSO pada umumnya disebut shared objects atau DSO files dan dapat dinamai dengan ekstensi yang beruba-ubah (walaupun nama yang resmi adalah foo.so). File-file ini pada umumnya bertempat di dalam suatu direktori yang telah ditentukan oleh program dan tidak secara otomatis terhubung pada program yang executable di mana mereka digunakan.

Sebagai gantinya program yang executable secara manual memuat DSO pada run-time ke dalam alamatnya melalui dlopen(). Pada waktu ini tidak ada pemecahan simbol dari DSO untuk program executable dilaksanakan. Tetapi sebagai gantinya Unix loader secara otomatis memecahkan (yang belum terpecahkan) simbol-simbol di dalam DSO dari satuan simbol yang diekspor oleh program yang executable dan telah memuat DSO libraries(terutama semua simbol dari libc.so yang ada dimana mana). Dari sini DSO mendapatkan isi simbol-

yang berhubungan dengan modul. Handler dapat melakukan salah satu dari tiga hal berikut:

1. Menangani request dan mengindikasikan bahwa request telah dilakukan, dengan mengembalikan konstanta OK.
2. Menolak untuk menangani request, dengan mengembalikan konstanta integer DECLINED. Pada kasus ini, server berjalan seakan-akan tidak ada request..
3. Memberi sinyal error, dengan menampilkan pesan error HTTP dan mengakhiri proses penanganan, dan me-log kejadiannya.

Kebanyakan tahap diakhiri oleh modul yang pertama yang menangani tahap tersebut; untuk *logging*, ` fixups', dan pemeriksaan non-access authentication, semua handlers selalu berjalan (menghalangi suatu kesalahan). Tahap respon adalah unik di dalam modul tersebut, dimana dapat mendeklarasikan beberapa handlers untuk hal tersebut, dengan menggunakan suatu dispatch table yang berpedoman pada jenis MIME dari obyek yang diminta. Modul-modul mungkin mendeklarasikan suatu handler response-phase, dimana dapat menangani tiap permintaan, dengan memberikan sebuah key */* (yaitu, suatu spesifikasi jenis wildcard MIME). Bagaimanapun, wildcard handlers hanya dilibatkan jika server telah mencoba dan tidak berhasil untuk menemukan suatu respon handler yang lebih spesifik untuk jenis MIME obyek yang diminta (meski obyek tersebut tidak ada, ataupun request ditolak).

Handlers sendiri adalah fungsi-fungsi dari satu argumen (sebuah struktur request_rec.), dimana mengembalikan nilai integer, seperti yang telah disebutkan sebelumnya.

2.4.1 Modul

Pada sub-bab berikut akan dijelaskan struktur dari sebuah modul. Salah satunya adalah modul CGI, dimana menangani script-script CGI dan ScriptAlias config file command. Modul CGI lebih kompleks dibandingkan dengan kebanyakan modul.

Untuk menangani script-script CGI, modul mendeklarasikan sebuah response handler dari script-script tersebut. Oleh karena ScriptAlias, modul juga menangani tahap name translation (untuk mengenali ScriptAliased URIs), tahap type-checking (tiap ScriptAliased request digolongkan sebagai script CGI).

Modul harus me-*maintain* beberapa dari tiap informasi server (virtual), yakni ScriptAliases; sehingga struktur modul berisi pointers untuk sebuah fungsi yang membuat struktur tersebut, dan untuk yang lainnya dimana mengkombinasikan dua dari struktur tersebut (jika main server dan virtual server memiliki ScriptAliases yang dideklarasikan).

Modul ini berisi kode untuk menangani perintah Scriptalias untuk modul itu sendiri. Modul tertentu ini hanya mendeklarasikan satu perintah, tetapi bisa lebih, maka modul mempunyai tabel perintah dimana mendeklarasikan perintah-perintah modul tersebut, dan menggambarkan dimana modul tersebut diijinkan, dan bagaimana dilibatkan.

Catatan akhir pada jenis argumen yang terdeklarasikan dari perintah-perintah berikut: pool adalah sebuah pointer dari struktur resource pool; dimana digunakan oleh server untuk menyimpan pelacakan memori yang telah dialokasi, file-file yang dibuka, serta untuk melayani request tertentu, atau untuk menangani proses konfigurasi itu sendiri. Sehingga pada saat request telah selesai (atau, untuk configuration pool, pada saat server di-*restart*), memori dapat dibebaskan, dan file-file ditutup, tanpa seorangpun harus menulis kode eksplisit untuk melacaknya. Demikian pula struktur cmd_parms terdiri dari berbagai informasi tentang file config yang sedang dibaca dan informasi status yang lainnya, dimana terkadang kegunaan dari fungsi yang memroses perintah config-file (seperti ScriptAlias).

2.4.2 Bagaimana Handler Bekerja

Hal terpenting yang ditangani adalah struktur request_rec. Struktur tersebut menggambarkan permintaan (request) tertentu, dimana yang telah dibuat untuk server, dalam memenuhi kepentingan klien. Dalam kebanyakan kasus, masing-masing koneksi ke klien menghasilkah hanya satu struktur request_rec.

2.5 PHP

PHP dahulunya merupakan proyek pribadi dari Rasmus Lerdorf (dengan dikeluarkannya PHP versi 1) yang digunakan untuk membuat home page pribadinya. Versi pertama ini berupa kumpulan script PERL. Untuk versi keduanya, Rasmus menulis ulang script-script PERL tersebut menggunakan bahasa C, kemudian menambahkan fasilitas untuk Form HTML dan koneksi

MySQL. Adapun PHP didapat dari singkatan Personal Home Pages. Setelah mengalami perkembangan oleh suatu kelompok open source(termasuk Rasmus) maka mulai versi 3 nya, PHP telah menampakkan keunggulannya sebagai salah satu bahasa server scripting yang handal. Melalui perkembangan yang pesat ini banyak fasilitas yang ditambahkan dan oleh kelompok ini PHP disebut sebagai "PHP: Hypertext Preprocessor". Sintak yang digunakan berasal dari bahasa C , Java maupun Perl. Sampai tulisan ini dibuat versi PHP yang terbaru adalah versi 4.1.1. Untuk release terbaru dari PHP dapat anda lihat pada web site http://www.php.net .

PHP merupakan bahasa script yang digunakan untuk membuat halaman web yang dinamis. Dinamis berarti halaman yang akan ditampilkan dibuat saat halaman itu diminta oleh client. Mekanisme ini menyebabkan informasi yang diterima client selalu yang terbaru. Semua script PHP dieksekusi pada server dimana script tersebut dijalankan. Oleh karena itu, spesifikasi server lebih berpengaruh pada eksekusi dari script php daripada spesifikasi client. Namun tetap diperhatikan bahwa halaman web yang dihasilkan tentunya harus dapat dibuka oleh browser pada client. Dalam hal ini versi dari html yang digunakan harus didukung oleh browser client.

PHP merupakan script yang menyatu dengan HTML dan berada pada server (server side HTML embedded scripting). Dengan PHP kita dapat membuat beragam aplikasi berbasis web, mulai dari halaman web yang sederhana sampai aplikasi komplek yang membutuhkan koneksi ke database.

Sampai saat ini telah banyak database yang telah didukung oleh PHP dan kemungkinan akan terus bertambah. Selain itu PHP juga mendukung koneksi dengan protokol IMAP, SNMP, NNTP dan POP3.

Untuk dapat menjalankan script php pada web site, ada beberapa hal yang perlu kita tambahkan . Pertama tentunya kita harus mempunyai sebuah web server yang mengatur atau memberikan tempat untuk mengeksekusi script php. Web server ini dinstall pada komputer server kita. Saat ini php dapat dijalankan pada berbagai macam web server seperti pws, iis, xitami maupun apache. Kemudian hal kedua yang perlu kita miliki adalah php parser. PHP parser adalah program yang digunakan untuk menterjemahkan(intepreter) code script dan kemudian mengeksekusinya. PHP parser dapat berupa program yang dijalankan pada suatu shell/DOS prompt yang biasanya berupa program yang telah terkompilasi yaitu php.exe. Selain itu PHP parser bisa juga berupa modul-modul yang diload oleh web server.

PHP juga termasuk dalam HTML-embedded , artinya code php dapat kita sisipkan pada sebuah halaman HTML. Untuk mengetahui bahwa baris - baris HTML merupakan suatu script php maka digunakan pasangan tag. Tag yang dapat digunakan untuk menyatakan script php adalah :

```
<? ... ?>
<% ... %>
<?php ... ?>
```

Gambar 2.3 Tag php

Untuk server yang menggunakan sistem operasi Linux dapat menggunakan Apache (www.apache.org) sebagai web servernya. Sedangkan untuk server yang menggunakan sistem operasi berbasis Windows di sarankan menggunakan Personal Web Server (yang merupakan produk Microsot, karena dapat lebih kompatibel dan dapat di gunakan untuk ASP). Web server Xitami (www.imatix.com) dapat digunakan apabila Anda menggunakan Windows NT.

2.6 Mod_Gzip

Masalah lambatnya koneksi internet tentu merugikan terutama bagi pihak pengguna, di antaranya bagi banyak perusahaan-perusahaan yang menyelenggarakan layanan produk barang maupun jasa gratis ataupun komersial melalui internet. Yaitu dengan teknologi kompresi halaman web.

Kompresi web page atau halaman web bukanlah teknologi baru akan tetapi perkembangannya semakin pesat akhir-akhir ini, karena perkembangan teknologi web jaringan yang padat sehingga menuntut aplikasi yang berbasis web untuk lebih cepat dan stabil.

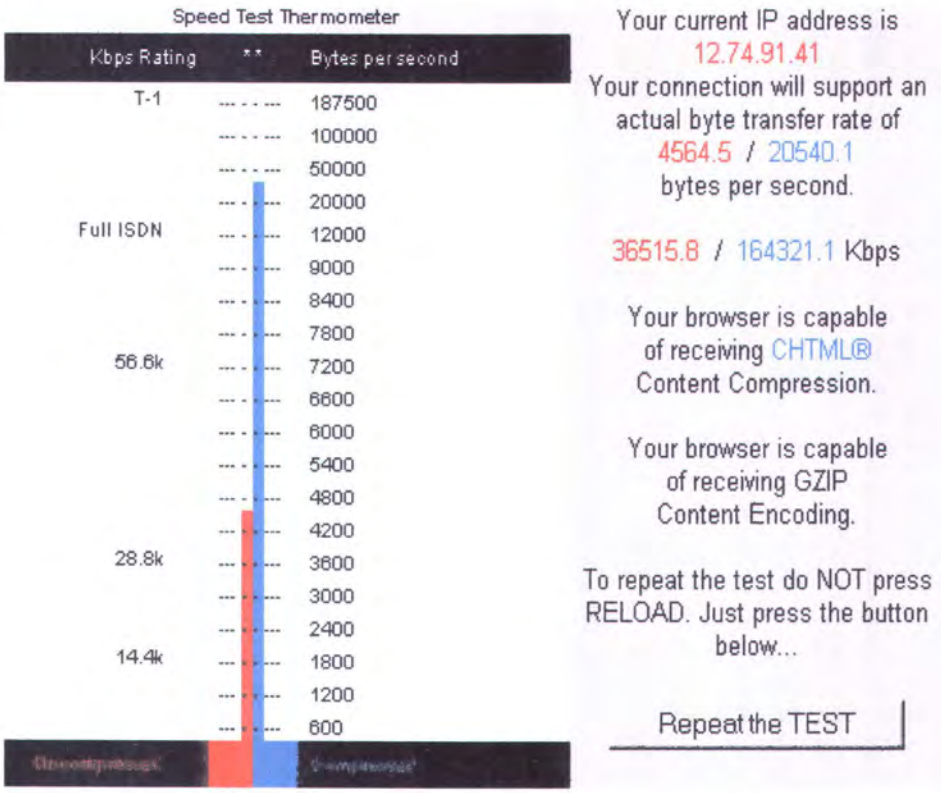
Ide di balik metodologi GZIP adalah sederhana, yaitu mengompres file yang akan di transmisikan ke client dan mengirimkan versi data terkompres tersebut ke client, hal ini di rasa lebih baik dari pada mengirimkan file asli yang ukurannya lebih besar, karena dapat menghemat bandwith pada jaringan. Besar file terkompresi ini tergantung pada file asal atau awal, biasanya mampu mengompres hingga 20% sampai 50% dari file awal.

Pada Apache, hal ini dapat di capai menggunakan Content Negotiation, yang mampu menghasilkan dua set terpisah dari file HTML yang dapat di generate, satu untuk clients yang dapat meng-handle GZIP-encoding, dan satu lagi untuk yang tidak dapat. Solusi ini mengirimkan file gzip-encoded kepada clients yang mengerti, akan tetapi juga berguna untuk kompresi halaman yang di generate secara dynamic.

Misal pada halaman-halaman yang di hasilkan oleh Server-Side Includes (SSI), PHP, dan generasi metode halaman dynamic yang lain. Dan dapat juga di gunakan untuk mengompres Cascading Stylesheets (CSS) serta file plain text.

Kompresi standar yang di gunakan saat ini adalah GZIP. Pada aplikasi client tidak perlu lagi di lakukan pemodifikasian, akan tetapi model kompresi ini hanya di dukung oleh browser yang mengenali protokol HTTP versi 1.1, jadi proses modifikasi hanya di perlukan bagi browser yang tidak mendukung HTTP versi 1.1, akan tetapi semua browser telah memiliki dukungan terhadap protokol HTTP/1.1 termasuk Internet Explorer versi 4.0 keatas, Netscape, Opera, dan Lynx Text Browser, jadi secara garis besar bukan client yang perlu melakukan proses pemodifikasian, akan tetapi server yang perlu pemodifikasian.

Apache yang saat ini menguasai pasar Web server dengan kurang lebih 60% dari seluruh pengguna aplikasi server di dunia menggunakan mod_gzip sebagai implementasi dari kompresi webnya. Proses intalasi mod_gzip pada umumnya dengan menggunakan APXS (Apache Extension ToolS) dengan implementasi DSO (Dynamik shared Object)



Gambar 2.4 Transmisi Tanpa dan Menggunakan HTTP kompresi

Gambar di atas adalah perbandingan penggunaan mod_gzip dan tidak menggunakan mod_gzip, tampak terjadi perubahan yang cukup sinifikan bahkan dapat mencapai 50% lebih cepat, tergantung jenis file yang dikirim.

2.7 Linux

Sistem operasi linux saat ini sudah tidak asing lagi bagi pengguna internet dan komunitas mahasiswa yang memiliki hobbi untuk mencoba software baru. Secara teknis dan singkat dapat dikatakan bahwa linux adalah suatu sistem operasi multi user dan multi tasking, yang dapat berjalan di berbagai platform termasuk prosesor intel 386 maupun yang lebih tinggi. Sistem operasi ini

simbol di dalam program yang executable seolah-olah telah terhubung secara statis sejak awal.

Akhirnya, untuk mengambil keuntungan dari DSO'S, API dari program yang executable harus memecahkan lambang tertentu dari DSO melalui dlsym() untuk penggunaan kemudian di dalam tabel pengiriman dan lain lain. Dengan kata lain: program yang executable harus memecahkan tiap-tiap simbol itu secara manual agar mampu menggunakannya. Keuntungan dari mekanisme seperti itu adalah komponen-komponen program yang opsional tidak perlu dimuat (sehingga tidak menghabiskan memori) sampai mereka diperlukan oleh program. Manakala diperlukan, komponen program ini dapat dimuat secara dinamis untuk memperluas fungsionalitas dasar program.

Walaupun mekanisme DSO ini terdengar lugas ada sedikitnya satu langkah sulit di sini: Pemecahan simbol-simbol dari program yang executable untuk DSO ketika menggunakan suatu DSO untuk memperluas suatu program (cara yang kedua). Mengapa? Sebab "pemecahan kebalikan" simbol-simbol DSO dari program executable berlawanan dengan desain library(di mana library tidak punya pengetahuan tentang program yang menggunakannya) dan tidak tersedia di semua platform maupun yang distandardisasi. Dalam prakteknya simbol-simbol global dari program yang executable sering tiidak ter export ulang dan tidak tersedia untuk digunakan di suatu DSO. Menemukan suatu cara untuk memaksa penghubung(linker) untuk mengekspor semua simbol-simbol global adalah masalah utama yang harus dipecahkan ketika menggunakan DSO untuk pengembangan suatu program pada run-time.

Windows dan Netware menyediakan fasilitas serupa, walaupun penerapannya sedikit berbeda dibanding uraian tentang DSO Unix sepanjang dokumen ini. Khususnya, DSO modules (NLM'S dan DLL's, berturut-turut) dibangun dengan cara yang berbeda dibanding milik Unix. Dokumen ini tidak bermaksud untuk menyelidiki topik membangun DSO modul pada platform-platform ini. Deskripsi mengenai mod_so dan konfigurasinya, bagaimanapun, serupa.

2.3.2 Pemakaian Praktis

Pendekatan melalui shared library adalah khusus, sebab untuk itulah mekanisme DSO dirancang, karenanya itu digunakan untuk hampir semua jenis library yang disediakan oleh sistem operasi. Pada sisi lain menggunakan shared object untuk mengembangkan suatu program tidak digunakan oleh banyak program.

Mulai dari 1998 hanya ada sedikit paket software yang tersedia yang menggunakan mekanisme DSO untuk benar-benar mengembangkan fungsionalitas software2 itu pada saat run-time: Perl 5 (dengan mekanisme XS nya dan Dynaloader modul), Netscape Server, dan lain lain. Mulai versi 1.3, Apache bergabung, sebab Apache telah menggunakan suatu konsep modular untuk mmperluas kemampuan nya dan secara internal menggunakan suatu pendekatan dispatch-list-based untuk menghubungkan modul eksternal ke dalam inti fungsionalitas Apache. Maka, Apache benar-benar ditakdirkan untuk menggunakan DSO untuk memuat modul-modulnya pada saat run-time.

Mulai Apache 1.3, sistem konfigurasi mendukung dua fitur opsional untuk mengambil keuntungan dari pendekatan DSO modular: kompilasi inti program Apache ke dalam suatu library DSO pemakaina bersama dan kompilasi modul-modul Apache ke dalam file-file DSO untuk pemuatan secara eksplisit pada run-time.

2.3.3 Implementasi

DSO mendukung untuk pemuatan modul-modul Apache secara individual berdasar pada suatu modul bernama mod_so.c yang harus secara statis di-compile ke dalam inti Apache. Itu adalah satu-satunya modul di samping http_core.c yang tidak bisa diletakkan dalam DSO itu sendiri (bootstrapping!). Pada kenyataannya semua modul Apache yang terdistribusi bisa ditempatkan ke dalam suatu DSO builder untuk modul-modul itu melalui pilihan configure --enable-shared (lihat INSTALL file di top direktori) atau dengan mengubah perintah Addmodule di dalam src/Configuration ke dalam perintah Sharedmodule (lihat file src/INSTALL). Setelah suatu modul di-compile ke dalam suatu DSO bernama mod_foo.so bisa digunakan perintah Loadmodule mod_so dalam file httpd.conf untuk memuat modul ini pada server startup atau restart.

Untuk menyederhanakan pembuatan file-file DSO untuk modul-modul Apache (terutama untuk modul-modul pihak ketiga) ada sebuah program pendukung baru bernama apxs (APACHE eXtenSion)yang tersedia. Program itu dapat digunakan untuk membangun DSO mendasarkan modul di luar Apache source tree. Gagasannya sederhana: Ketika menginstall Apache, prosedur make

install pada configure menginstall file C header Apache dan menempatkan kompiler yang platform-dependent serta flag-flag penghubung untuk membangun file-file DSO ke dalam apxs program. Dari sini user dapat menggunakan apxs untuk mengcompile source modul-modul Apache tanpa distribusi source tree Apachedan tanpa harus untuk memainkan platform-dependent compiler dan penghubung flag mendukung DSO.

Untuk menempatkan inti program Apache yang lengkap ke dalam suatu library DSO (yan] hanya memerlukan beberapa dari platform yang didukung untuk memaksa linker mengekspor simbol inti apache-- suatu prasyarat untuk DSO modularisasi) aturan SHARED_CORE itu harus diaktifkan lewat pilihan configure --enable-rule=SHARED_CORE (lihat file INSTALL) atau dengan mengubah perintah Rule dalam file Configuration ke Rule SHARED_CORE=YES (lihat file src/INSTALL). Kode inti Apache kemudian ditempatkan dalam suatu library DSO bernama libhttpd.so. Sebab orang tidak bisa menghubungkan suatu DSO melawan terhadap library statis pada semua platform, suatu program executable tambahan bernama libhttpd.ep diciptakan yang mana kedua-duanya mengikat kode statis ini dan menyediakan suatu potongan untuk fungsi main(). Akhirnya httpd program executable itu sendiri digantikan oleh suatu kode yang bootstrapping yang secara otomatis memastikan Unix loader bisa mengisi dan memulai libhttpd.ep dengan menyediakan LD_LIBRARY_PATH ke libhttpd.so.

2.3.4 Keuntungan Dan Kerugian

Fitur-fituer berdasar DSO pada Apache 1.3 di atas mempunyai keuntungan berikut :

1. Paket server lebih fleksibel pada run-time sebab proses server sebenarnya dapat dirakit pada run-time lewat perintah konfigurasi Loadmodule httpd.conf sebagai ganti perintah Configuration Addmodule pada build-time. Sebagai contoh lewat dari sini kita bisa menjalankan instance server yang berbeda (versi standard& SSL, versi minimalistic&diperkuat [mod_perl, PHP3], dll.) hanya dengan satu instalasi Apache.
2. Paket server dapat dengan mudah diperluas dengan modul-modul pihak ketiga bahkan setelah instalasi selesai dilakukan. Ini merupakan suatu manfaat besar untuk vendor packager maintainer yang dapat menciptakan suatu inti package Apache package tambahan yang berisi ekstensi seperti PHP3, mod_perl, mod_fastcgi, dan lain lain.
3. Lebih mudah membuat prototipe Apache sebab dengan pasangan DSO/APXS kedua-duanya dapat dikerjakan di luar Apache source tree dan hanya memerlukan perintah apxs -i diikuti apachectl start untuk menjalankan versi baru modul yang tengah dikembangkan ke dalam server Apache.

Fitur-fituer berdasar DSO pada Apache 1.3 di atas mempunyai kerugian-kerugian berikut :

1. Mekanisme DSO tidak bisa digunakan pada setiap platform sebab tidak semua sistem operasi mendukung pemuatan kode secara dinamis ke dalam alamat dari suatu program.

2. Server kira-kira 20% lebih lambat pada waktu startup karena pemecahan simbol membebani hal-hal yang harus dilakukan oleh Unix loader sekarang.
3. Server kira-kira 5% lebih lambat pada waktu eksekusi pada beberapa platform sebab posisi kode independent (PIC) kadang-kadang trik perakitan yang rumit untuk pengalamatan relatif yang tidak secepat pada pengalamatan absolut.
4. Modul-modul DSO modul tidak bisa dihubungkan melawan DSO-based library yang lain (ld - lfoo) pada semua platform (contohnya platform a.out-based pada umumnya tidak menyediakan kemampuan ini sedangkan platform ELF-BASED menyediakan) DSO mekanisme tidak bisa digunakan untuk semua jenis modul. Atau dengan kata lain, modul-modul yang dikompile sebagai file-file DSO terbatas hanya menggunakan simbol-simbol dari inti Apache, dari library C(libc) dan semua library statis atau dinamis lain yang digunakan oleh inti Apache, atau dari arsip library statis (libfoo.a) yang berisi kode dengan posisi independen. Satu-Satunya kesempatan untuk menggunakan kode lain ialah dengan memastikan inti Apache itu sendiri telah berisi acuan ke kode itu, memuat kode itu lewat dlopen() atau mengaktifkan aturan SHARED_CHAIN ketika membangun Apache saat platform yang digunkan mendukun penghubungan file-file DSO melawan library-library DSO.

Pada beberapa platform (banyak sistem SVR4) tidak ada cara untuk memaksa linker untuk mengekspor semua simbol-simbol global untuk digunakan pada DSO ketika menghubungkan Apache httpd program yang executable. Tapi tanpa kejelasan dari inti simbol-simbol Apache tidak ada modul Apache standar

yang bisa digunakan sebagai suatu DSO. Satu-Satunya kesempatan di sini ialah menggunakan fitur SHARED_CORE sebab dengan ini simbol-simbol global terpaksa diekspor. Konsekwensinya, script src/Configure Apache secara otomatis mengeluarkan fitur SHARED_CORE pada platform ini manakala fitur-fitur DSO digunakan dalam file Configuration atau pada baris perintah configure.

2.4 Apache API

Konsep dasar API dan bagaimana perkembangannya, apache membagi request handling dalam beberapa bagian, kurang lebih hampir sama dengan cara kerja Netscape server API, pembagian handlingnya di antaranya :

1. URI yang berfungsi sebagai penerjemah input.
2. Auth ID checking.
3. Auth access checking.
4. Access checking lainnya dari auth.
5. Menentukan jenis MIME dari object yang diminta
6. `Fixups' --- there aren't any of these yet, but the phase is intended as a hook for possible extensions like SetEnv, which don't really fit well elsewhere.
7. Mengirimkan respon kembali ke klien.
8. Me-logging request

Tahap-tahap tersebut ditangani dengan melihat tiap-tiap rangkaian modul, melihat jika masing-masing diantaranya mempunyai suatu handler untuk tahap,

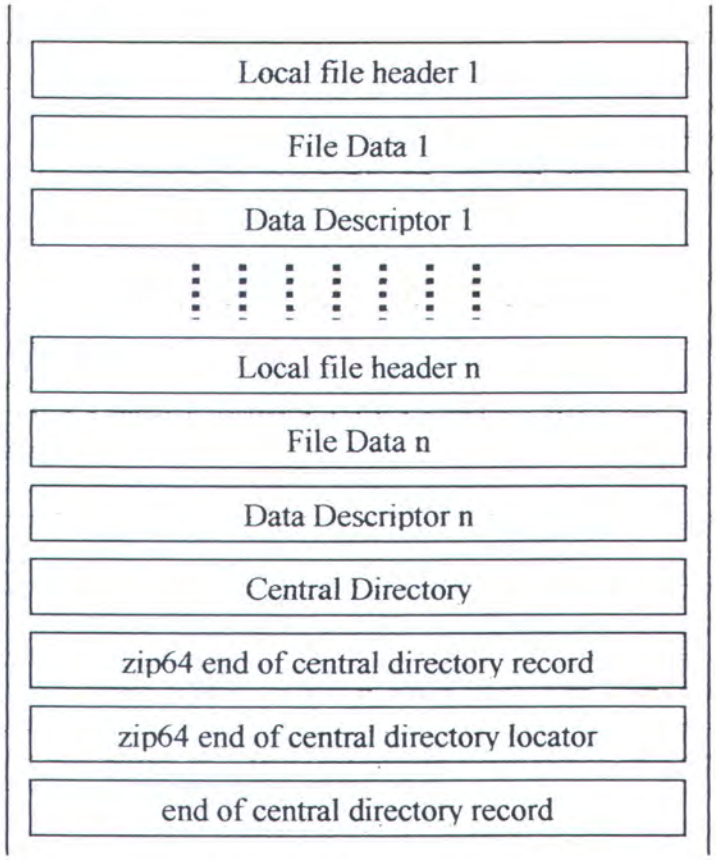
mengimplementasikan standard POSIX. Linux dapat berinteroperasi secara baik dengan sistem operasi yang lain termasuk Apple, Microsoft dan Novell.

Nama linux sendiri di turunkan dari pencipta awalnya Linus Torvalds, yang sebetulnya mengacu pada suatu kumpulan software lengkap yang bersama-sama dengan kernel menyusun suatu sistem operasi yang lengkap.

Lingkungan sistem operasi ini mencakup ratusan program, termasuk *compiler*, interpreter, editor dan utilitas. Perangkat Bantu yang mendukung konektifitas, Ethernet, SLIP dan PPP, dan interoperabilitas. Produk perangkat lunak yang handal (reliable), termasuk versi pengembangan terakhir. Kelompok pengembang yang tersebar di seluruh dunia yang telah bekerja dan menjadikan Linux portable ke suatu platform baru, begitu juga mendukung komunitas pengguna yang memiliki beragam kebutuhan dan juga pengguna dapat turut serta bertindak sebagai tim pengembang sendiri.

2.8 Format File ZIP

File ZIP yang sudah di kenal luas dan digunakan untuk menggabungkan berbagai macam file, terutama untuk membackup data. ZIP adalah suatu aplikasi untuk mengkompres dan meng-archive file-file sehingga dapat lebih kompatibel dan menghemat space.



Gambar 2.5 Format file ZIP

ZIP trailer block adalah format file zip yang banyak di gunakan, yang di tulis secara sequential. file yang sedang ditambahkan mendapatkan suatu header file lokal dan datanya. Manakala semua file ditulis kemudian suatu direktori pusat ditulis dan direktori sentral ini dapat di bagi-bagi dalam beberapa disk. Dan setiap disk mempunyai blok descriptor yang berisi pointer dari awal sentral directory.

descriptor ini selalu di tulis diakhir oleh karena itu biasa di sebut "ZIP File Trailer Block".

ZIP Trailer selalu berada pada akhir dari file zip dan dia mempunyai panjang yang sudah di tentukan, dan nilai magic four-byte berada pada awal block. hal ini yang mempermudah untuk mendeteksi file zip tapi pada kenyataannya tidak semudah itu, kita dapat menambah comment text pada zip archive setelah Trailer block. hal ini biasa di gunakan saat ini, akan tetapi hal ini mengharuskan zip reader dapat mencari Trailer block mulai dari akhir file dan mencari Trailer magic inilah kegunaan fungsi internal __zip_find_disk_trailer.

Struktur yang memberikan nilai balik yang di sebut zzip_disk_trailer dalam library source code, dan nilai yang di butuhkan ada dua yaitu u_rootseek dan u_rootsize. nilai pertama ini dapat di gunakan untuk lseek ketempat dimana sentral directory berada dan nilai yang kedua memberitahu besar ukuran byte dari sentral direktori.

BAB III

PERANCANGAN PERANGKAT LUNAK

BAB III

PERANCANGAN PERANGKAT LUNAK

Bab ini membahas tentang perancangan perangkat lunak, untuk pengamanan script PHP dan HTML, dengan menggunakan algoritma TAR, menggunakan algoritma enkripsi penyisipan karakter, dimana script yang akan dieksekusi nantinya berupa file .TAR yang telah terenkripsi dan apache akan dapat mengeksekusi file tersebut kemudian file dieksekusi oleh apache, dan sebelum file dikirim ke client, terlebih dulu hasil eksekusi dikompres oleh module tambahan yang akan ditambahkan pada apache yaitu module gunzip(Mod_gzip) dan dikirim ke client, dan diclient file tersebut diunkompres dengan sendirinya apabila client menggunakan browser minimal mendukung HTTP 1.1.

3.1 Deskripsi Sistem

Client mengakses menggunakan browser dengan menginputkan alamat dari file yang akan diakses, dan request(inputan) dari client tersebut diterima oleh apache dan inputannya ditangani oleh handler dan dari hasil tersebut handler akan menentukan perlakuan terhadap file yang direquest oleh user nantinya misalnya file dengan ekstensi .php, .html atau yang lainnya, maka pada Apache web server file tersebut dieksekusi sesuai dengan module untuk mengeksekusi file tersebut, dan setelah file tersebut dieksekusi oleh Apache kemudian file hasil eksekusi (yang berupa file HTML) tersebut sebelum dikirim pada client melalui jaringan

akan di kompres terlebih dahulu oleh module gunzip dan setelah file tersebut sampai pada client maka file html tadi di exstrac oleh browser.

Cara kerja dan proses aluran data pada Apache web server ini adalah proses alur yang standart dari Apache web server, dan pada aplikasi yang di buat ini akan memodifikasi alur proses serta menambahkan module pada Apache web server dengan Mod_Gzip yaitu module untuk kompresi data sebelum proses pengiriman data ke client.

3.2 Spesifikasi Sistem

Spesifikasi perangkat lunak yang di kembangkan dalam sistem ini di jelaskan sebagai berikut :

1. Pembuatan File TAR yang akan di akses, pembuatan file tar ini di maksudkan untuk menggabungkan file-file php dan html, untuk mempermudah proses enkripsinya, setelah file tar tersebut jadi, maka proses selanjutnya adalah proses enkripsi. Kemudian file tar ini diletakkan pada server untuk di akses.
2. Pembuatan Module Untuk mengakses File TAR, pembuatan module TAR ini, di dalamnya terdapat proses-proses, yang di antaranya adalah proses men-dekript file TAR tersebut dan kemudian proses meng-Untar file yang di minta dan kemudian mengekseskusi file yang telah di Untar tersebut. Kemudian file yang telah di ekseskusi tersebut di kirimkan ke client yang me-request file tersebut. Jadi pada module ini terdapat 4 langkah proses yaitu, men-decrypt, meng-Untar, mengekseskusi dan mengirimkan.

3. Penambahan Module Mod_Gzip, pada proses atau langkah penambahan module gzip ini, yang di lakukan adalah proses instalasi dan setting pada file httpd.conf langkah-langkah lebih lanjutnya di jelaskan pada bab selanjutnya.

3.3 Kebutuhan Sistem

Sistem atau perangkat lunak yang dikembangkan dapat digunakan pada sistem operasi linux, dengan berbagai macam distro. Asalkan mempunyai kompiler gcc dan Untuk bisa menjalankan perangkat lunak ini dengan baik dibutuhkan komputer dengan memori fisik minimal sebesar 64 Mb. Pada komputer dengan memori fisik kurang dari itu, perangkat lunak ini berjalan lambat. Output pada perangkat lunak ini berupa berupa output biasa pada browser.

3.4 Desain Sistem

Pada sub bab ini akan dijelaskan mengenai desain perangkat lunak untuk rekayasa Apache web server sehingga dapat menghasilkan aplikasi yang cukup handal dalam hal keamanan dan kecepatannya, yang memenuhi spesifikasi dan kebutuhan sistem seperti yang telah dijelaskan pada sub bab sebelumnya. Desain rekayasa perangkat lunak ini meliputi desain proses, desain struktur data, desain algoritma yang digunakan. Desain proses menjelaskan tentang proses transaksi yang merupakan alur data dari sistem, sedangkan desain Data berisikan tentang gambaran serta penjelasan inputan data yang ada serta hasil keluaran atau output data dari aplikasi, dan desain algorithma adalah desain dan proses serta langkah-langkah dalam penyusunan struktur sistem yang di buat.

3.4.1 Desain Proses

Desain proses digunakan untuk mengetahui proses apa saja yang berlangsung pada sistem. Untuk desain proses dari perangkat lunak ini digunakan Diagram Aliran Data (DAD). Pembuatan DAD tersebut menggunakan Tool Power Designer versi 6.0.

DAD akan menunjukkan hubungan antara proses satu dengan yang lain, entitas–entitas atau orang–orang, serta data masukan dan data keluaran yang terlibat dalam proses–proses tersebut.

3.4.1.1 Proses Meng-encrypt file TAR

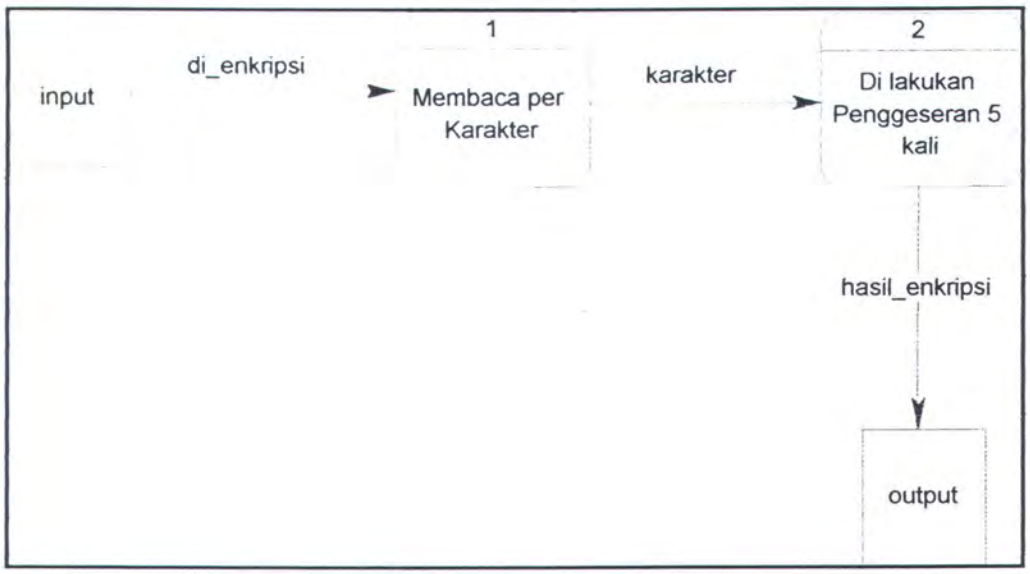
Desain untuk proses enkripsi file TAR dapat di lihat seperti berikut, di buat dua macam encrypt yaitu dengan algorithma penggesaran karakter dan yang kedua adalah dengan meng-XOR karakter dengan angka.

Penggeseran karakter

Gambar 3.1 DAD level 0 enkripsi file

Terlihat pada gambar di atas bahwa proses enkripsi file TAR, di mana prosesnya memasukkan inputan berupa file TAR dan outputnya adalah file TAR juga.

Gambar di bawah ini adalah gambar DAD level 1 dari proses enkripsi, di mana pada gambar terlihat proses untuk mengenkripsi file TAR, yaitu dengan cara penggeseran karakter.



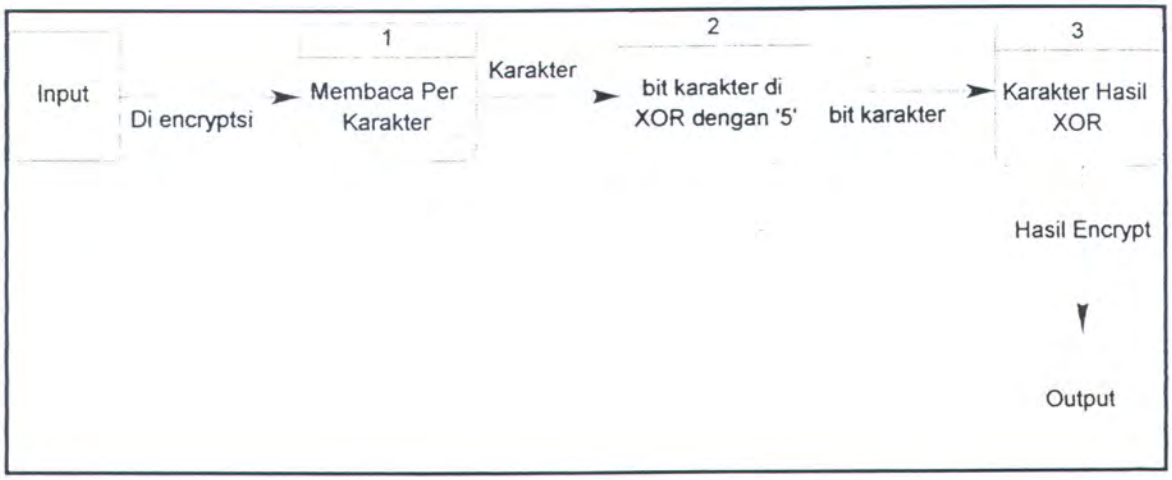
Gambar 3.2 DAD level 1 Proses enkripsi

Tampak pada gambar diatas bahwa proses enkripsi di lakukan dengan membaca karakter isi file TAR dan menggesernya kemudian menuliskan pada file output.

XOR Karakter Pada File

Pada proses encryptsi ini di lakukan XOR bit karakter yang di baca dengan bit karakter lain, dalam hal ini dengan angka '5' sehingga file setelah di encrypt menjadi tidak berpola.

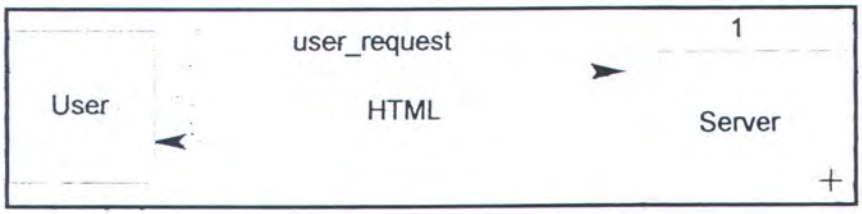
Gambar 3.3 DAD level 0 enkripsi file dengan XOR



Gambar 3.4 DAD level 1 Proses enkripsi XOR

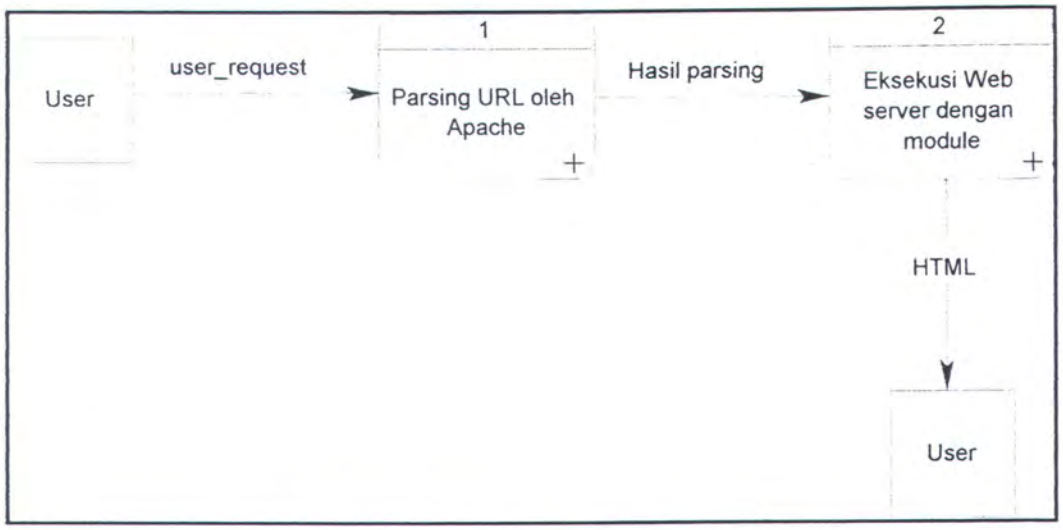
3.4.1.2 Proses Pada Module TAR

Proses-proses yang terjadi pada module TAR di antaranya adalah parsing input dari URL, proses decrypt file TAR, proses ekstrak file yang di request dan proses pengiriman file hasil eksekusi ke client.



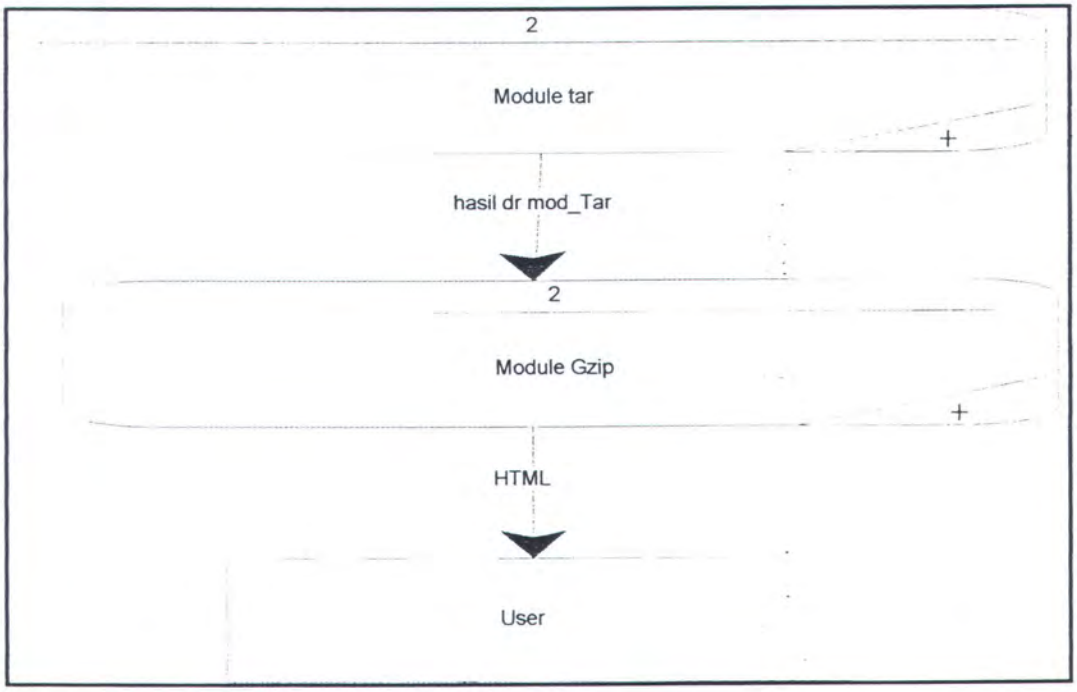
Gambar 3.5 DAD level 0 Module TAR

Gambar diatas adalah gambar DAD level 0 pada proses awal jika client me-request pada server dan server mengirimkan request berupa file HTML pada client.



Gambar 3.6 DAD level 1 Module TAR

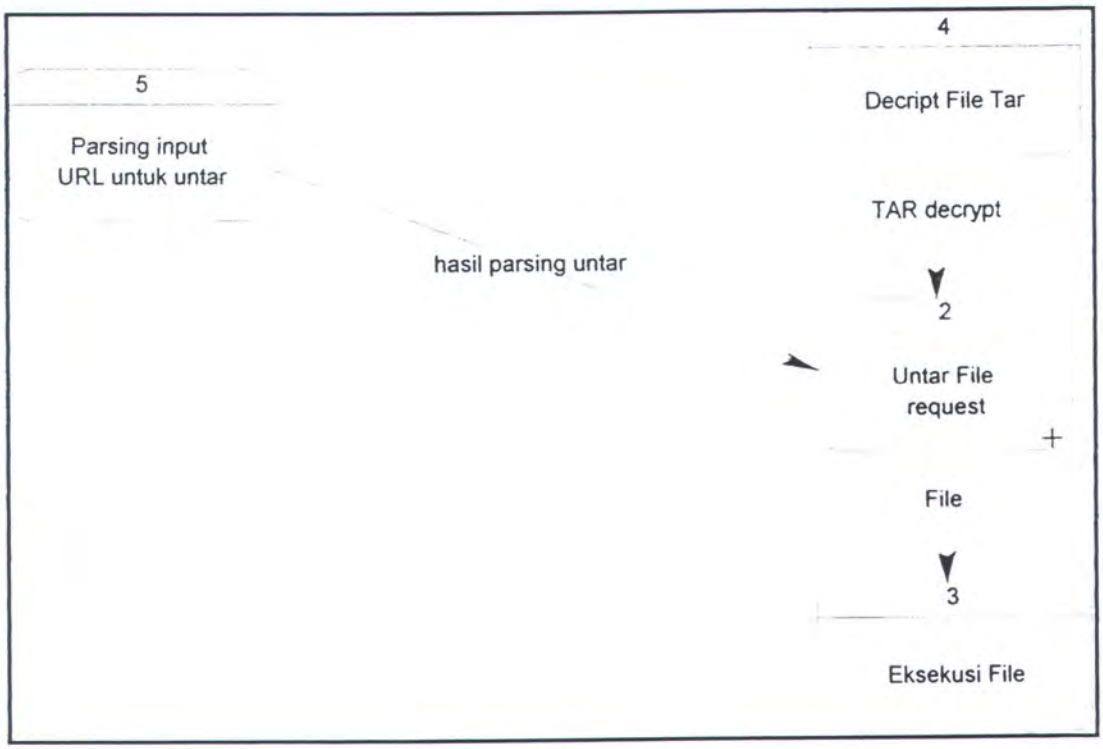
Pada gambar diatas terlihat bahwa proses yang terjadi didalam server adalah terjadinya proses parsing dari URL yang dalam hal ini di tangani oleh handler dan proses selanjutnya adalah eksekusi dengan module yang sesuai dengan ekstensi filenya.



Gambar 3.7 DAD Level 2 Module TAR

Proses pada module yang sesuai ini, yaitu tepatnya adalah module TAR, sehingga apabila handler mengenali ekstensi file .tar maka handler akan memanggil module TAR untuk menangani file yang ada pada URL, dan kelihatan pada gambar diatas ada sebuah proses module gzip sebelum file request sampai pada client.

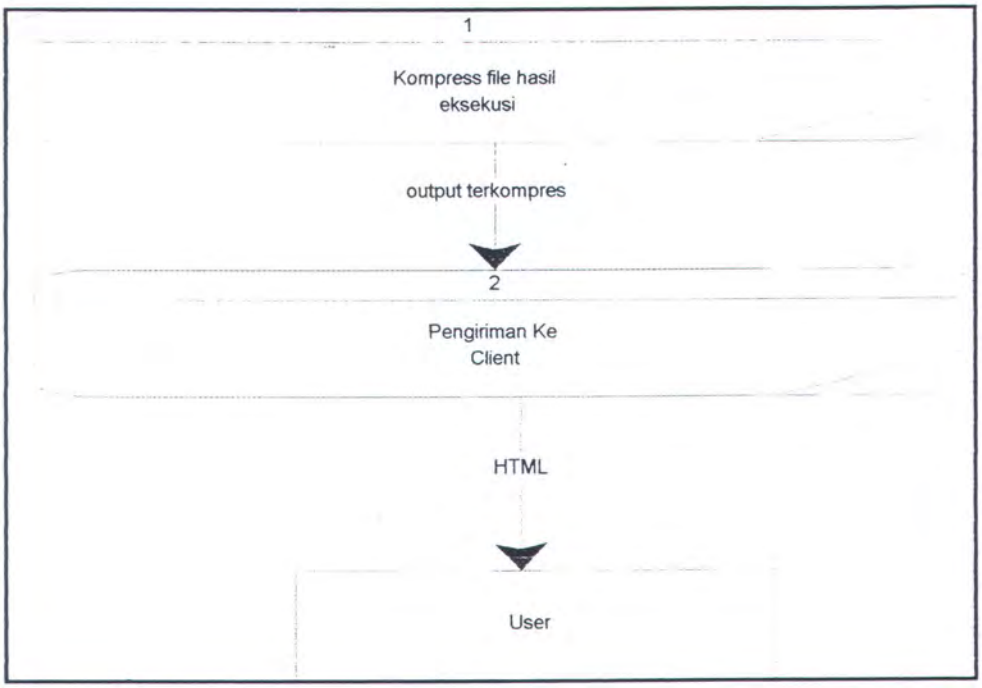
Gambar dibawah terlihat bahwa proses yang terjadi di dalam module TAR yaitu berupa parsing input, decrypt, untar file serta pengiriman file hasil eksekusi kepada client yang me-request file tersebut.



Gambar 3.8 DAD Level 3 Module TAR

3.4.1.3 Proses Module GZIP

Pada gambar di bawah ini tampak proses didalam module gzip, di mana terjadi kompresi pada file yang akan di kirim ke client. Dan proses untuk meng-ekstrak file yang di kirim terjadi di client.



Gambar 3.9 Proses pada module GZIP

3.4.2 Desain Data

Data yang digunakan untuk implementasi perangkat lunak ini dibagi menjadi tiga bagian utama, yaitu data masukan, data yang digunakan selama proses serta data keluaran.

3.4.2.1 Data Masukan

Data masukan adalah data yang di masukkan oleh user atau client pada browser, dimana masukannya berupa alamat server serta request file yang akan di ekstrak beserta path dari file yang akan di ekstrak. Sistem penulisannya adalah :

```
\localhost\nama_file.tar\nama_file_yg_direquest
```

Gambar 3.10 Inputan client pada browser

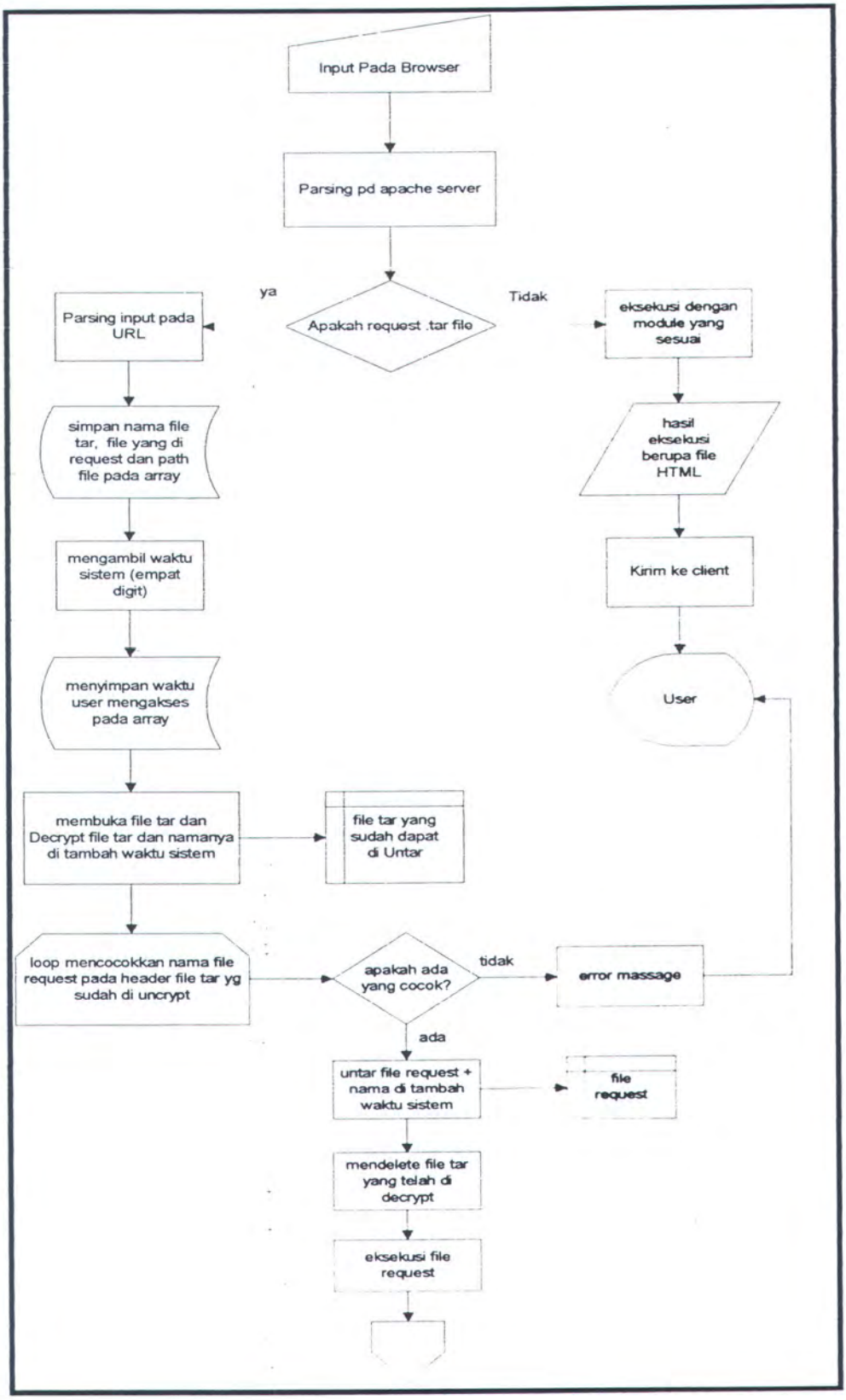
3.4.2.2 Data Selama Proses

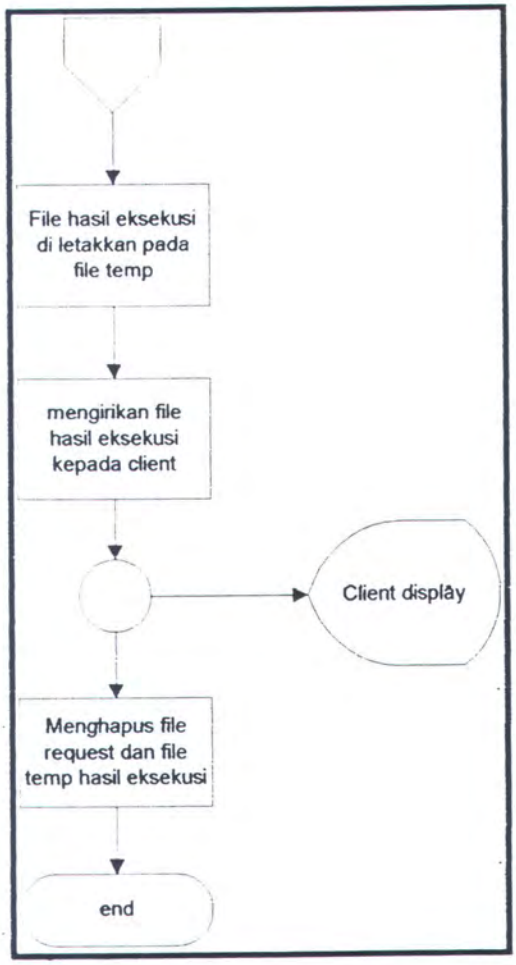
Data yang di butuhkan selama proses pada module ini adalah data nama file tar, data file yang di request oleh client, serta path letak data tersebut. Dan dalam proses tersebut data lain yang di butuhkan adalah data waktu client request ke server, hal ini di gunakan untuk memberikan penandaan atau penamaan pada file yang telah di uncrypt dan file yang telah di ekstrak. Jadi semisal file tar yang di *encrypt* bernama file.tar saat client merequest ke server maka di ambil waktu sistem sebanyak 4 (empat) digit sehingga file tar yang telah di uncrypt menjadi file.tar+empat_digit_waktu+sistem misal waktu sistem adalah 1234 maka file tar sekarang file.tar1234.

3.4.2.3 Data Keluaran

Data keluaran adalah data yang dihasilkan dari proses-proses yang terdapat dalam sistem. Data yang keluar dari sistem dapat berupa sebuah file html, atau file gambar(jpg, gif dan lain-lain) yang kemudian di kirimkan ke browser client.

3.4.3 Desain Algoritma





Gambar 3.11 Flowchart sistem

Dari flowchart diatas dapat di lihat alur proses dari file ketika di request oleh client sampai pada proses ekseskusi dan di kirimkan ke client.

BAB IV

IMPLEMENTASI PERANGKAT LUNAK

BAB IV

IMPLEMENTASI PERANGKAT LUNAK

Dalam bab ini akan dibahas proses pembuatan sistem yang telah dirancang pada bab sebelumnya. Pembuatan perangkat lunak atau aplikasi berbasis web ini dibagi menjadi 4 yaitu :

1. Setting pada file konfigurasi Apache yaitu pada file httpd.conf.
2. Pembuatan perangkat lunak untuk meng-*encrypt* dan mengamankan file tar.
3. Pembuatan perangkat lunak berupa module pada apache web server untuk mengakses file tar.
4. Pemasangan module gzip untuk Apache server.
5. Me-restart Apache web server.

4.1 Setting file konfigurasi Apache Web Server

Hal yang paling penting atau yang paling utama tentang konsep module dalam apache web server adalah bagaimana inputan melalui browser dikenali sebagai apa serta bagaimana penanganannya untuk inputan yang dimasukkan melalui browser ini. Sebagai gambarannya, kita memasukkan sebuah inputan pada browser seperti : http://localhost/~coba/tes.php maka pada Apache server akan memparsing inputan dengan ekstensi .php jika ada maka selanjutnya apa yang akan dilakukan oleh server tersebut, sehingga apabila kita hendak memodifikasi misalnya membuat apache dapat membaca ekstensi yang lain .tar misalnya.

Untuk mengubah prilaku Apache server dalam menghandle file yang mempunyai ekstensi .TAR maka perlu di lakukan setting pada Apache server, tepatnya pada file httpd.conf yaitu menambahkan sintaks,

```
AddType application/x-httpd-enc .tar
```

Gambar 4.1 Setting pada file httpd.conf

Sehingga apabila client melakukan request pada browser dan requestnya mempunyai ekstensi .tar maka handler akan memanggil module yang akan mengekseskusi URL request.

4.2 Pembuatan file encrypt

4.2.1 Dengan Algorithma Caesar

File-file yang akan di akses yang nantinya di letakkan di web server adalah file dalam format TAR yang telah di *encrypt* dan di modifikasi sehingga file tar tersebut tidak dapat di akses.

Langkah pertama adalah membuat file TAR, dengan menggunakan sintaks seperti biasa, akan tetapi pembuatan tersebut di lakukan pada folder letak file-file yang akan di TAR berada, hal ini di lakukan untuk mempermudah pembuatan file TAR tersebut. Sintaks untuk membuat tar file tersebut adalah :

```
$ > tar -cvf nama_file.tar *
```

Gambar 4.2 Sintaks membuat file TAR

Kemudian setelah file tersebut menjadi file TAR maka langkah selanjutnya adalah melakukan enkripsi. Algorithma untuk meng-encrypt file tersebut dengan menggunakan algorithma Caesar cipher, yaitu dengan mengubah karakter-karakter yang ada pada file. Algoritma Caesar cipher ini pada intinya adalah menggeser letak alphabet dengan alphabet yang lain, dapat di jelaskan seperti dibawah ini (jika si geser 3 kali) :

Plain Alphabet a b c d e f g h i j k l m n o p q r s t u v w x y z

Cipher Alphabet D E F G H I J K L M N O P Q R S T U V W X Y Z A B C

Jika, Plaintext : Hello how are you.

Maka, Ciphertext : KHOOR KRZ DUH BRX.

Algoritma ini akan meng-*encrypt* A-Z dan a-z, akan tetapi karakter yang lain hanya akan di lewatkan, tidak di *encrypt* juga.

Pembuatan enkripsi file TAR yaitu dengan cara menyalin isi file input dan memindahkan isi file tersebut ke file output, akan tetapi sebelumnya di lakukan proses penukaran karakter dari file input kemudian di lakukan proses penulisan pada file output. Sintaks untuk membuka file input adalah seperti berikut :

```
if ((in=fopen(argv[1],"r")) == NULL){
      fprintf(stderr,"Error - cannot
      open %s \n\n",argv[2]);exit(2);
}
```

Gambar 4.3 Fungsi membuka file input

Apabila file input tidak dapat di buka maka akan menampilkan pesan error dan apabila pembukaan file input tersebut berhasil maka langkah selanjudnya

adalah memanggil fungsi encode, yaitu fungsi untuk meng-encrypt karakter pada file input.

```
encode(in, stdout, shift);
```

Gambar 4.4 Memanggil fungsi menggeser karakter

File input tersebut di baca satu persatu, yaitu karakter demi karakter, kemudian dilakukan pengecekan untuk karakter yang di baca dan apabila karakter yang di baca adalah karakter alphabet maka di lakukan penggeseran sebanyak 5 kali, dan apabila karakter yang di baca adalah karakter selain itu maka karakter langsung di tulis pada file output. Fungsi untuk menggeser alphabet tersebut adalah sebagai berikut :

```
if((inF = open(argv[1], O_RDONLY)) == -1) {
      perror("open");
      exit(-1);
}
if((ouF = open(argv[2], O_WRONLY | O_CREAT |
O_EXCL, 0666)) == -1) {
      perror("open");
      exit(-1);
}
while((bytes = read(inF, line, sizeof(line))) > 0){
                  line[0] = line[0] - 5;
                  write(ouF, line, bytes);
}
close(inF);
close(ouF);
```

Gambar 4.5 Fungsi untuk menggeser karakter

Dapat di lihat pada fungsi ini, dimana pembacaan tiap karakter dalam file di lakukan pengecekan dengan kondisi konsi dan syarat-syarat yang ada, sehingga apabila ada syarat yang terpenuhi maka akan di geser sesuai dengan inputan penggeseran, dalam hal ini adalah 5 geseran sehingga karakter A di gantikan dengan karakter F dan demikian seterusnya. Dan setelah dilakukan penggeseran tersebut karakter yang sudah di geser di tulis pada file output yang sudah di buat.

Source program tersebut kemudian di konpile dengan menggunakan GCC Linux, yaitu dengan menggunakan sintak :

```
$ > gcc -o file_output file_input
```

Gambar 4.6 Mengompilasi Program Pada GCC

Setelah proses kompilasi, jika terdapat kesalahan pada program, maka akan terdapat pesan kesalahan dan letak kesalahan tersebut, jika program telah benar maka pada direktori yang sama dengan program terdapat file output tadi. Dan langkah berikutnya adalah meng-encrypt file TAR, sintak untuk meng-encrypt file TAR adalah :

```
$>./file_output nama_file.tar file_out.tar
```

Gambar 4.7 Meng-encrypt file TAR

Setelah seluruh proses pada pembuatan file TAR selesai maka langkah berikutnya adalah meletakkan file tersebut pada folder public_html, yaitu dengan dengan cara mengkopinya.

4.2.2 Dengan Algorithma XOR karakter

Proses yang terjadi pada meng-encrypt file TAR dengan menggunakan XOR, prosesnya sama, baik dalam pembacaan karakter file TAR dan penulisannya, akan tetapi yang berbeda adalah proses dalam meng-encrypt karakter yang di baca dan kemudian di tuliskan pada file output.

Proses encryptnya berupa merubah bit atau meng-XOR bit pada karakter yang di baca dengan karakter lain, dimana bit yang sama (bernilai 1) akan menghasilkan nilai '0', kemudian hasil XOR tersebut di tulis pada file output.

```
if((inF = open(argv[1], O_RDONLY)) == -1) {
      perror("open");
      exit(-1);
}
if((ouF = open(argv[2], O_WRONLY | O_CREAT |
O_EXCL, 0666)) == -1) {
      perror("open");
      exit(-1);
}
while((bytes = read(inF, line, sizeof(line))) > 0){
                  line[0] = line[0] ^ 5;
                  write(ouF, line, bytes);
}
close(inF);
close(ouF);
```

Gambar 4.8 Fungsi meng-XOR

4.3 Pembuatan Module TAR

Bagian yang paling utama dalam pembuatan tugas akhir ini adalah pembuatan module TAR, proses-proses yang terjadi pada module TAR adalah pembacaan input user (parsing input), men-decrypt file TAR, meng-untar file request, mengeksekusi file request, dan mengirimkan file hasil eksekusi.

4.3.1 Membaca Inputan User

Proses awal yang terjadi pada module TAR ini adalah proses parsing dari inputan user di browser (pada URL), proses parsing ini akan memisahkan nama file TAR serta nama file request dan path filenya.

Proses ini di perlukan untuk proses selanjutnya, dan proses ini di perlukan juga untuk mengecek apakah file request tersebut ada atau tidak didalam file TAR. Jika file request tersebut ada maka proses selanjutnya akan di lakukan, akan tetapi jika file tersebut tidak ada maka akan di tampilkan pesan kesalahan pada browser.

```
for(count=0; count < strlen(input); count++){
      if(input[count]=='/') indkator++;
      if(indkator == 2 && input[count] !='/'){
          namatar[cnttar] = input[count]; cnttar++;
      }
      if(indkator >= 3){
          if(strlen(input)-count != 1)
            tempURI[cntURI] = input[count+1];
          cntURI++;
      }
}
for(count=0; count < strlen(input); count++){
      if(input[count]=='/') ind2++;
      if(ind2 == indkator && input[count] != '/'){
            filespec[cntspec] = input[count];cntspec++;
            filespec[cntspec]='\0';
      }
}
```

Gambar 4.9 Program Parsing input

Hasil Proses parsing di letakkan pada array char dan hasil parsing ini di gunakan untuk proses selanjutnya.

4.3.2 Men-decrypt File TAR

Proses selanjudnya setelah memparsing inputan dari browser selanjutnya adalah men-decrypt file tar. Pada proses ini ada hal-hal yang perlu di perhatikan, diantaranya adalah file TAR ini akan di akses lebih dari satu user, sehingga di perlukan penanda dalam setiap file yang akan di eksekusi, oleh karena itu di perlukan sebuah cara lagi untuk penamaan pada file TAR yang di akses ini. Yaitu dengan cara menambahkan nama file TAR tersebut dengan waktu sistem,

prosesnya adalah setiap ada client yang me-request file pada server maka di lakukan pencatatan waktu sistem, kemudian nama file output TAR yang telah di decrypt di beri tambahan waktu sistem, semisal nama file TAR adalah FILE.tar dan waktu sistem adalah 2345 maka nama file output setelah di decrypt adalah FILE.tar2345 penambahan ini dapat menyebabkan file TAR yang akan di akses menjadi unik.

```
void ambil3dgt(char *S1,char *S2,int M,int P){
        int I,J,N;
        for(N=0;S2[N];N++);
        if(M>=0 && M<=N-1){
          for(I=M,J=0;(J<P && S2[I]);I++,J++)
            S1[J]=S2[I];
          S1[P]='\0';
        }
        else
          S1[0]='\0';
      }
```

Gambar 4.10 Fungsi mengambil waktu sistem

Dari gambar diatas telihat bahwa terdapat tiga parameter dalam memanggil fungsi tersebut, dalam memanggil fungsi tersebut di perlukan parameter-parameter lain di antaranya dapat di lihat sebagai berikut :

```
        time_t t;
        t = time(NULL);
        sprintf(temptime,"%ld",t);
        ambil3dgt(TIME,temptime,6,4);
```

Gambar 4.11 Untuk memanggil fungsi Time

Tampak pada gambar diatas di perlukan empat parameter yaitu TIME, temptime, 6 dan 4, parameter terakhir adalah besarnya waktu sistem yang diambil, dalam detik.

Setelah mengambil waktu sistem dan meletakkannya pada variabel maka langkah selanjutnya adalah men-decrypt file TAR, pada fungsi ini proses yang terjadi adalah seperti halnya pada proses untuk meng-encrypt file TAR akan tetapi ada penambahan proses, yaitu proses menamaan file TAR setelah di decrypt.

Proses awal adalah membaca file TAR yang di encrypt kemudian di buat file output yang sebelumnya proses penamaannya di tambah waktu sistem, kemudian file TAR yang di baca di tulis pada file TAR baru, dan terlebih dahulu isi filenya di geser terlebih dahulu.

Setelah file TAR tersebut di decrypt maka file tersebut dapat di akses oleh fungsi lainnya untuk meng-untar file yang di request.

```
strcat(namatarblm,namatar);//tar yang masih di encrypt
    strcat(namataracc,namatar);//nama tar yang deencrypt
    strcat(namataracc,TIME);
    if((inF=open(namatarblm,O_RDONLY))==-1){
      perror("open");
      exit(-1);
    }
    if((ouF=open(namataracc,O_WRONLY|O_CREAT|O_EXCL,
     0666))==-1){
      perror("open");
      exit(-1);
    }
    while((bytes = read(inF,line,sizeof(line)))>0){
      line[0] = line[0] - 5;
      write(ouF,line,bytes);
    }

    close(inF);
    close(ouF);

    strcpy(buang3,filespec);//nama file yang di
ekstrak(just name)
  if((in=fopen(namataracc,"rb"))==0){
            return 1;
            }
```

Gambar 4.12 Fungsi untuk men-decrypt file TAR pada penggeseran karakter

```
strcat(namatarblm,namatar);//tar yang masih di encrypt
    strcat(namataracc,namatar);//nama tar yang deencrypt
    strcat(namataracc,TIME);
    if((inF=open(namatarblm,O_RDONLY))==-1){
      perror("open");
      exit(-1);
    }
    if((ouF=open(namataracc,O_WRONLY|O_CREAT|O_EXCL,
     0666))==-1){
      perror("open");
      exit(-1);
    }
    while((bytes = read(inF,line,sizeof(line)))>0){
     line[0] = line[0] ^ 5;
      write(ouF,line,bytes);
    }

    close(inF);
    close(ouF);

    strcpy(buang3,filespec);//nama file yang di
ekstrak(just name)
  if((in=fopen(namataracc,"rb"))==0){
          return 1;
          }
```

Gambar 4.13 Fungsi untuk men-decrypt file TAR pada XOR karakter

4.3.3 Meng-Untar File Request

Setelah proses decrypt file TAR selesai, maka file TAR yang ada sekarang dapat di akses seperti biasa, maka langkah selanjutnya adalah mengekstrak file yang di request oleh client, proses mengekstrak ini memerlukan inputan dari proses lain, yaitu proses parsing input karena untuk mengekstrak di butuhkan nama file yang di request.

```
int istarheader(unsigned char *hdr,long csum,char
askflag){
   for(cnt=0;cnt<=147;cnt++) newsum+=*(hdr+cnt);
   for(cnt=156;cnt<512;cnt++) newsum+=*(hdr+cnt);
      newsum+=256;
      if(newsum==csum) return 1;
      else {
        if(askflag)return 0;
        else return 0;
      }
}
```

Gambar 4.14 Fungsi mengecek header file TAR

```
int extract_tar(void){
        char* buffer;
        int cnt;
        if((buffer=malloc(UNC_OUTBUFSIZ))==0){
                return -1;
                }
        while(!feof(in)){

        cnt=fread(buffer,1,UNC_OUTBUFSIZ,in);
                untarstream(buffer,cnt);
        }
        return 0;
}
```

Gambar 4.15 Fungsi extract TAR

Pada fungsi di atas proses yang terjadi adalah membuka file TAR dan kemudian di baca header filenya, lalu proses selanjudnya di lanjutkan dengan proses pada fungsi untarstream.

```
int untarstream(unsigned char data[], unsigned long
datasize){
  datapos=0;
  for(;;){
      if(position==-1) {
            do{datapos+=512;
               if(datapos>datasize) return 0;
      }while(!istarheader((char*)&header,csum,1));
if (header.name[0] == 0)break;
      strcpy(KML,header.name);
      position=0; skip=0; strcpy(dosname,header.name);
if (header.flags=='5'||header.name[strlen(header.name)-
1]=='/'){if (header.name[strlen(header.name)-1] == '/')
         header.name[strlen(header.name)-1] = 0;
         skip=1; /* Directory ??? */}
         else if(banding()==1){salah=1;
               skip=openfile(header.name,dosname);}
            else skip=1;
      }//end if atas
      if ((size-position) <= (datasize-datapos)){
            dsize = size-position;}
      else dsize = (datasize-datapos);
      if(!skip){
      if(write(outfile,data+datapos,dsize)<dsize)error++;
                  c_break();}}
      position += dsize;
      datapos += dsize;
      if(size==position) {
            if(!skip){close(outfile);}
            position=-1;
if(datapos % 512) datapos=( (datapos / 512) + 1) * 512;
            printf("OK   \n");}
if(datasize==datapos)
            return 0;}return 0;}
```

Gambar 4.16 Fungsi Untar Stream

4.3.4 Mengeksekusi File Request

Setelah file yang di request tersebut selesai di ekstrak maka proses selanjutnya adalah mengeksekusi file yang selesai di request tersebut. Proses pengeksekusian ini di perlukan untuk file PHP saja, karena file yang lain tidak perlu di eksekusi, hanya langsung di kirim ke client.

Fungsi untuk mengeksekusi file PHP ini dapat di lihat seperti di bawah ini:

```
toPHP()
{
    strcpy(namacopy,"php ");
    strcat(namacopy,nametes);
    strcat(namacopy," > ");
    strcat(namacopy,path);
    strcat(namacopy,"aku.mbo");
    system(namacopy);
    strcpy(buang2,"cp ");
    strcat(buang2,path);
    strcat(buang2,"aku.mbo ");
    strcat(buang2,buang1);
    system(buang2);
    system("rm -f /home/kemalsyah/public_html/aku.mbo");
}
```

Gambar 4.17 Fungsi untuk mengeksekusi file PHP

Tampak pada fungsi diatas bahwa setelah file PHP di eksekusi maka file hasil eksekusinya di letakkan pada file lain atau file temporarry, dan kemudian file PHP hasil eksekusi tersebut di kopi pada file yang akan di kirimkan ke client.

4.3.5 Mengirimkan File Eksekusi

```
static int hello_handler(request_rec *r){
          strcpy(input,r->uri);
          take_input();
          if(strstr(filespec,".php") != 0){
            r->content_type = "text/html";
            ap_send_http_header(r);
            if (!r->header_only){
                extract_tar();
                toPHP();
                f = ap_pfopen(r->pool,buangl,"r");
                while(fgets(line,sizeof(line),f) != NULL){
                  ap_rputs(line,r);}ap_pfclose(r->pool,f);
            }}
          if(strstr(filespec,".html") != 0 ){
            r->content_type = "text/html";
            ap_send_http_header(r);
            if (!r->header_only){extract_tar();
                f = ap_pfopen(r->pool,buangl,"r");
                while(fgets(line,sizeof(line),f) != NULL){
                  ap_rputs(line,r);}ap_pfclose(r->pool,f);}  }
          else if(strstr(filespec,".gif") != 0 ) {
            r->content_type = "image/jpg";
            ap_send_http_header(r);
            if (!r->header_only){
                extract_tar();
              if((inF = open(buangl/*tesgbr1*/,O_RDONLY))==-1){
                  perror("open");exit(-1);
                }
              while((bytes = read(inF,line,sizeof(line))) > 0){
                  ap_bwrite(r->connection->client,line,bytes);
                } } } return OK;}
```

Gambar 4.18 Fungsi mengirimkan file hasil eksekusi

Langkah berikutnya setelah proses ekstrak selesai dan pengeksekusian file PHP selesai maka proses selanjutnya adalah proses mengirimkan file request ke client. Pada fungsi ini terdapat statement kondisi yang bermacam-macam hal ini di maksudkan untuk memperlakukan file yang ada dengan perlakuan yang sesuai.

Misalnya file gambar, yang berekstensi .JPG maka file ini langsung di kirim ke client tanpa harus di proses lagi. Begitu juga dengan file html atau file lain yang tidak membutuhkan proses pengolahan lagi. Langsung di kirim ke client, seperti tampak pada fungsi di atas bahwa file-file di proses berdasarkan ekstensi filenya, ada yang langsung di kirim ke client dan ada yang masih harus di proses lagi.

4.4 Instalasi Module Gzip

Setelah seluruh rangkaian proses pembuatan dan peletakkan file TAR, serta penambahan module dan kompilasi module TAR selesai maka langkah berikutnya adalah menambahkan module gzip, langkah-langkah proses instalasi adalah :

1. Login sebagai Administrator/root.
2. Membuat folder atu direktori /home/modgziptmp

   mkdir modgziptmp

   dan kemudian merubah kepemilikan direktori dengan

   chmod 777 modgziptmp
3. Mengambil source file mod_gzip 1.3.26a.1a dan meletakkan pada direktori /usr/local/src

   cd usr local src

wget http: telia.dl.sourceforge.net sourceforge mod_gzip-1.3.26.1a.tgz

4. Meng-extract file dan merubah ke direktori.

   tar xfz mod_gzip-1.3.26.1a.tgz

   cd mod_gzip-1.3.26.1a

5. Kompile/install mod_gzip dengan mengetikkan 2 commands (path dari apxs dapat juga berbeda tergantung servernya).

   make APXS=/usr/local/apache/bin/apxs

   make install APXS=/usr/local/apache/bin/apxs

Setelah berhasil melakukan proses instalasi dengan benar maka pada file httpd.conf perlu di lakukan penyetingan untuk konfigurasi-konfigurasi untuk mod_gzip.

```
mod_gzip_item_exclude        file        \.js$
mod_gzip_item_exclude        mime        ^text/css$
mod_gzip_item_include        file        \.html$
mod_gzip_item_include        file        \.shtml$
mod_gzip_item_include        file        \.php$
mod_gzip_item_include        mime        ^text/html$
mod_gzip_item_include        file        \.txt$
mod_gzip_item_include        mime        ^text/plain$
mod_gzip_item_include        file        \.css$
mod_gzip_item_include        mime        ^text/css$
```

Gambar 4.19 Setting pada httpd.conf untuk mod_gzip

Sedangkan untuk mengkonfigure server untuk meng-GZIP-encode file PDF di perlukan setting sebagai berikut :

```
mod_gzip_item_include     file      \.pdf$
mod_gzip_item_include     mime     ^application/pdf$
```

Gambar 4.20 Setting untuk file PDF

Module ini juga dapat bekerja secara baik baik pada Mozilla maupun Opera, karena aplikasi ini dapat mendecode content GZIP-encode sebelum melemparkannya pada PDF reader (kebanyakan orang menggunakan Adobe Acrobat Reader).

4.5 Menjalankan Apache Web Server

Setelah seluruh proses instalasi dan setting selesai, maka untuk mencoba dan menjalankan aplikasi yang di buat maka terlebih dahulu Apache web server harus di hidupkan atau di start.

Untuk menjalankan server tersebut menggunakan sintak :

```
$> /usr/local/apache/bin/apachectl start
```

Gambar 4.21 Sintak untuk start Apache

Apabila ada pesan kesalahan dan Apache server tidak dapat di start maka yang perlu di periksa adalah file konfigurasi yaitu pada httpd.conf, lihat bagian mana yang menyebabkan eror, kemudian di cari pada file konfigurasi apache.

BAB V

UJI COBA DAN EVALUASI PERANGKAT LUNAK

BAB V

UJI COBA DAN EVALUASI PERANGKAT LUNAK

Dalam bab ini dibahas mengenai uji coba aplikasi penambahan module apache web server yang telah dirancang dan dibuat. Uji coba dilakukan untuk mengetahui apakah aplikasi dapat berjalan sebagaimana mestinya dengan lingkungan uji coba yang telah ditentukan serta dilakukan sesuai dengan skenario uji coba. Uji coba di lakukan pada file yang bervariasi, baik ukuran, jenis serta operasi yang di lakukan oleh file, keterangan tentang file yang akan di akses dapat di lihat pada tabel dibawah ini.

Tabel 5.1 Keterangan File

FILE	KETERANGAN
1	File php berukuran berukuran 1 KB, melakukan fungsi sederhana
2	File php berukuran berukuran 1 KB, melakukan fungsi sederhana
3	File php berukuran berukuran 1 KB, melakukan fungsi sederhana dan me-request file gambar(1KB)
4	File php berukuran berukuran 1 KB, melakukan fungsi sederhana
5	File php berukuran berukuran 1 KB, melakukan fungsi query pada database mysql
6	File php berukuran berukuran 1 KB, melakukan uji coba koneksi ke database
7	File HTML biasa yang tidak me-request file gambar(37 KB)
8	File HTML biasa yang tidak me-request file gambar(20 KB)
9	File HTML(1 KB) yang me-request file gambar sebesar 39 kb
10	File HTML(1 KB) yang me-request beberapa file gambar sebesar 32.3 kb
11	Merequest file gambar secara langsung sebesar 39 KB

5.1 Lingkungan Uji Coba Ideal

Proses uji coba yang akan dilakukan terbagi dalam dua kondisi, yaitu uji coba pertama adalah uji coba keamanan, dan yang kedua adalah uji coba untuk kecepatan acces. Uji coba keamanan, akan di bandingkan file yang telah di lakukan proses security dengan file biasa pada module apache standart.

Sedangkan uji coba kedua akan di bandingkan kecepatan akses untuk proses menggunakan apache web server tanpa ada module tambahan(apache yang standart), dan pada kondisi percobaan yang kedua adalah apache web server dengan menambahkan module yang dibuat menggunakan algoritma tar dan *encript* file.

Pada ujicoba dengan menggunakan web server apache tanpa menggunakan penambahan module tar yang di buat, dan tanpa penambahan module gzip, dimana nanti akan dilakukan pencatatan waktu sejak request user sampai proses permintaan user selesai.

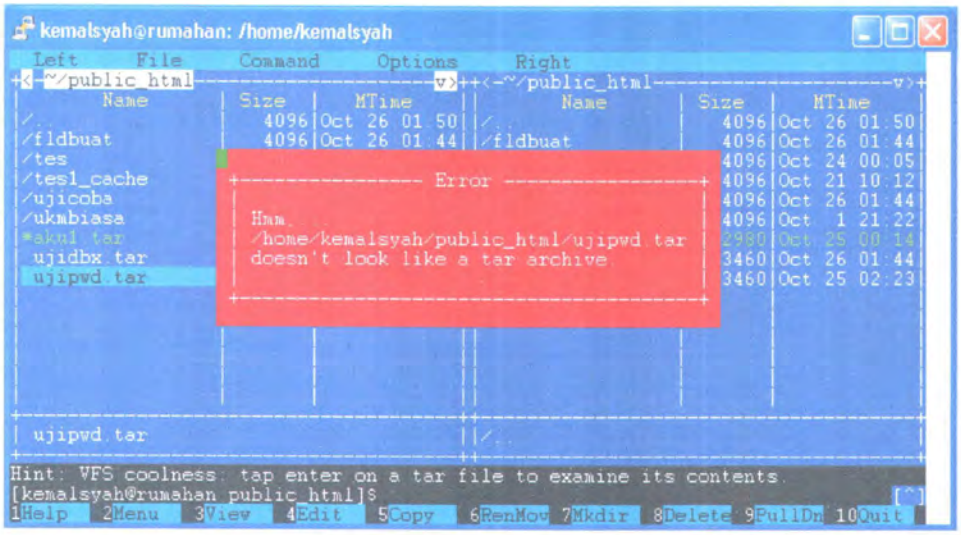
Pada proses akses menggunakan apache biasa, dilakukan request file yang umum digunakan pada aplikasi web yaitu HTML, PHP, file gambar (.JPG, .GIF).

Pencatatan waktu di ulang sebanyak 5 kali. Dan percobaan dilakukan dengan 11 halaman halaman web, yang diantaranya berisi file php, html dan gambar.

5.1.1 Uji Coba Keamanan

Pada sub bab ini, dibandingkan keamanan antara file standart dan file yang menggunakan aplikasi yang dibuat :

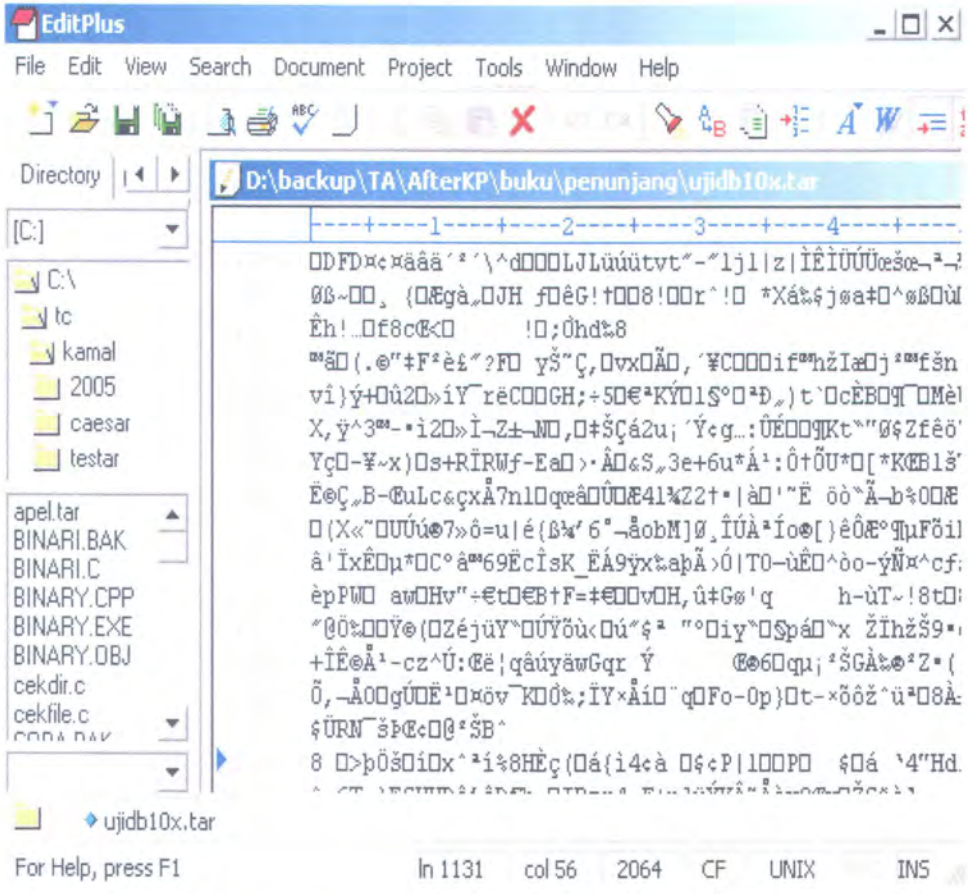
Gambar 5.1 Source Code Web Tanpa *encrypt*



Gambar 5.2 Menggunakan File yang di-*encrypt*

Tampak pada gambar di atas, dimana pada aplikasi biasa source dapat dilihat dengan mudah, akan tetapi pada file yang telah diarchive menggunakan file tersebut maka tidak dapat diakses.

Apabila file yang telah di-tar dan di-*encrypt* tersebut dibuka dengan program-program editor, misalnya saja EditPlus, maka file tersebut akan tampak, seperti gabungan file text dan file biner yang acak, dan akan membingungkan orang yang tidak mempunyai hak akses ke sana, gambar file tar yang di-*encrypt* dilihat dengan menggunakan teks editor.



Gambar 5.3 File encrypt yang dibuka dengan EditPlus

5.1.1.1 Kecepatan Dengan Algoritma Caesar

Pada sub bab ini akan di lakukan perbandingan kecepatan proses pengaksesan file TAR yang di encript dengan algoritma caesar. Dan pecobaan ini di lakukan dengan file-file standart yang berukuran kecil.

Tabel 5.2 Waktu Pengaksesan dengan Algoritma Caesar

Nama File	Waktu					Total
1	313	328	343	305	311	**1600**
2	205	221	215	200	232	**1073**
3	500	453	445	476	475	**2349**
4	185	205	210	192	190	**982**
5	325	308	313	320	312	**1578**
6	330	340	314	321	319	**1624**
7	330	340	302	305	313	**1590**
8	510	514	505	498	512	**2539**
9	310	297	299	297	303	**1506**
10	610	605	610	609	612	**3046**
11	202	206	199	204	195	**1006**

Tabel 5.3 Rata-rata waktu akses dalam mili second

FILE	1	2	3	4	5	6	7	8	9	10	11
RATA-RATA	320	215	470	196	316	325	318	508	301	609	201

5.1.1.2 Kecepatan Dengan Algoritma XOR

Berikut ini akan di lakukan pengukuran kecepatan akses jika menggunakan algorithma encripsi XOR pada module TAR.

Tabel 5.4 Waktu Pengaksesan dengan Algoritma XOR

Nama File	Waktu					Total
1	310	330	342	305	300	**1587**
2	200	220	213	203	230	**1066**
3	500	450	445	475	473	**2343**
4	184	205	205	190	190	**974**
5	325	305	310	320	312	**1572**
6	330	340	304	310	305	**1589**
7	330	340	300	298	313	**1581**
8	510	514	505	498	512	**2539**
9	310	287	299	297	300	**1493**
10	610	605	600	608	605	**3028**
11	205	200	196	204	195	**1000**

Tabel 5.5 Rata-rata waktu akses dalam mili second

FILE	1	2	3	4	5	6	7	8	9	10	11
RATA-RATA	317	213	469	195	314	317	316	507	299	605	200

5.1.2 Uji Coba Kecepatan Akses Apache

Berikut ini adalah hasil perhitungan waktu dari apache web server tanpa menggunakan module tambahan, yang di hitung adalah acces time, yaitu pencatatan seluruh proses dari awal request sampai akhir selesai proses, waktu dalam ukuran milli second.

Tabel 5.6 Waktu Pengaksesan dengan Apache standart

Nama File	Waktu					Total
1	210	220	235	200	197	**1062**
2	100	120	100	105	135	**560**
3	400	350	345	375	370	**1840**
4	84	100	105	90	90	**469**
5	225	200	210	220	210	**1065**
6	230	240	205	200	200	**1075**
7	230	240	200	198	210	**1078**
8	405	400	400	398	400	**2003**
9	220	185	198	197	200	**1000**
10	510	505	500	505	500	**2520**
11	100	100	95	100	95	**490**

Tabel 5.7 Rata-rata waktu akses dalam mili second

FILE	1	2	3	4	5	6	7	8	9	10	11
RATA-RATA	212	112	368	94	213	215	216	401	200	504	98

5.1.3 Uji Coba Kecepatan Akses Apache Dengan Module TAR

Pada bagian ini akan di lakukan uji coba dengan berbagai sekenario, di antaranya adalah sebagai berikut.

5.1.3.1 Uji coba Pada File Berukuran Kecil

Pada uji coba menggunakan apache web server yang dengan menambah module tar, sekenario percobaannya masih sama dengan sekenario percobaan diatas, file yang di akses juga sama. Penghitungan waktu untuk percobaan menggunakan module tar ini adalah sebagai berikut:

Tabel 5.8 Waktu Pengaksesan dengan Apache yang di tambah module tar

Nama File	Waktu					Total
1	308	320	322	310	302	**1562**
2	200	210	203	202	222	**1037**
3	500	450	445	465	463	**2323**
4	184	197	202	190	190	**963**
5	315	305	310	306	312	**1548**
6	320	315	304	310	305	**1554**
7	310	320	300	298	313	**1541**
8	510	504	505	498	507	**2524**
9	303	287	298	297	300	**1485**
10	602	605	600	595	605	**3007**
11	202	199	196	201	195	**993**

Tabel 5.9 Rata-rata waktu akses dalam mili second

FILE	1	2	3	4	5	6	7	8	9	10	11
RATA-RATA	312	207	465	193	310	311	308	505	297	601	199

5.1.3.2 Uji Coba Pada File Berukuran Besar.

Pada sekenario berikut ini, akan dilakukan pengukuran kecepatan akses dengan menggunakan file-file yang berukuran besar secara berurutan. File-file tersebut akan di ukur kecepatan pengaksesannya dan akan dilakukan perbandingan antar filenya, penghitungannya di lakukan sebanyak kurang lebih 3 kali percobaan. Dan algoritma encripsi yang di gunakan adalah algoritma caesar.

Tabel 5.10 Pengukuran file besar

Besar File	Waktu Pengaksesan		
100 KB	1.382	1.332	1.322
200 KB	2.252	2.34	2.362
300 KB	3.244	3.185	3.334
400 KB	3.893	3.91	3.762
500 KB	4.496	4.507	4.427
600 KB	6.324	6.385	6.402
700 KB	7.203	7.21	7.431
800 KB	8.302	8.251	8.382
900 KB	10.956	10.905	11.06
1 MB	11.624	11.582	11.681
1100 KB	12.731	12.642	12.801
1200 KB	13.199	13.299	13.459
1300 KB	15.112	15.223	15.321
1400 KB	17.354	17.521	17.481
1500 KB	19.111	19.125	19.421
1600 KB	20.69	20.469	19.288
1700 KB	21.311	21.251	21432
1800 KB	22.135	22.31	22.282
1900 KB	23.511	23.522	23.45
2 MB	24.824	24.912	25.11

Dari hasil pengukuran kecepatan akses file besar maka di dapat data seperti di atas, dan rata-rata kecepatan akses adalah sebagai berikut :

Kecepatan pada ukuran file 100 KB	:	1.345
Kecepatan pada ukuran file 200 KB	:	2.318
Kecepatan pada ukuran file 300 KB	:	3.254
Kecepatan pada ukuran file 400 KB	:	3.855
Kecepatan pada ukuran file 500 KB	:	4.477
Kecepatan pada ukuran file 600 KB	:	6.370
Kecepatan pada ukuran file 700 KB	:	7.281
Kecepatan pada ukuran file 800 KB	:	8.312
Kecepatan pada ukuran file 900 KB	:	10.974
Kecepatan pada ukuran file 1 MB	:	11.629
Kecepatan pada ukuran file 1100 KB	:	12.725
Kecepatan pada ukuran file 1200 KB	:	13.319
Kecepatan pada ukuran file 1300 KB	:	15.219
Kecepatan pada ukuran file 1400 KB	:	17.452
Kecepatan pada ukuran file 1500 KB	:	19.219
Kecepatan pada ukuran file 1600 KB	:	20.149
Kecepatan pada ukuran file 1700 KB	:	21.331
Kecepatan pada ukuran file 1800 KB	:	22.242
Kecepatan pada ukuran file 1900 KB	:	23.494
Kecepatan pada ukuran file 2 MB	:	24.949

5.1.4 Uji Coba Dengan Module Gzip

Uji coba perbandingan apache web server dengan menggunakan mod_gzip dan apache tanpa menggunakan mod_gzip, sekenario percobaannya adalah pada file apache web server di letakkan file yang berukuran cukup besar kurang lebih 1.4 MB, dan pada jaringan lokal di ukur kecepatan down load file tersebut.

Uji coba pertama dengan menggunakan apache web server tanpa module g_zip, file di download sebanyak tiga kali, dan uji coba kedua dengan tambahan module g_zip maka di dapat hasil (dalam milli second).

Tabel 5.11 Perbandingan Waktu akses dalam mili second pada module gzip

Kondisi	**Waktu**		
Tanpa mod_gzip	790	603	570
Dengan mod_gzip	620	510	470

Maka rata-rata download file tanpa module gzip *adalah*

(790 + 630 + 570) / 3 = 663.33 milli second

Sedangkan rata-rata download file dengan module gzip *adalah*

(620 + 510 + 470) / 3 = 533,33 milli second

5.1.4.1 Uji Coba Dengan Module TAR dan Module Gzip

Pada sub bab ini akan di lakukan perbandingan kecepatan proses pengaksesan file TAR yang di encript dengan algoritma caesar dan di tambah dengan module Gzip. Dan pecobaan ini di lakukan dengan file-file standart yang berukuran kecil.

Tabel 5.12 Waktu Pengaksesan dengan module TAR dan module Gzip

Nama File	Waktu					Total
1	305	316	319	308	300	**1548**
2	198	207	200	201	218	**1024**
3	487	447	442	463	463	**2302**
4	182	194	201	190	187	**954**
5	312	302	307	304	310	**1535**
6	309	312	303	308	304	**1536**
7	307	318	298	296	312	**1531**
8	507	503	503	493	503	**2509**
9	300	286	296	296	299	**1477**
10	602	601	599	593	603	**2998**
11	201	197	195	200	193	**986**

Tabel 5.13 Rata-rata waktu akses dalam mili second

FILE	1	2	3	4	5	6	7	8	9	10	11
RATA-RATA	310	205	460	191	307	307	306	502	295	600	197

5.1.4.2 Uji Coba Dengan Module TAR Caesar dan Module Gzip

Pada sub bab ini akan di lakukan perbandingan kecepatan proses pengaksesan file TAR yang di encript dengan algoritma caesar dan di tambah dengan module Gzip. Dan pecobaan ini di lakukan dengan file-file standart yang berukuran kecil.

Tabel 5.14 Waktu Pengaksesan dengan module TAR Caesar dan Gzip

Nama File	Waktu					Total
1	311	326	330	305	311	**1583**
2	205	215	215	200	230	**1065**
3	494	453	445	474	473	**2339**
4	183	200	206	192	190	**971**
5	319	308	313	320	312	**1572**
6	315	320	314	321	319	**1589**
7	320	332	302	305	311	**1570**
8	506	500	505	498	511	**2520**
9	302	297	299	297	298	**1493**
10	601	605	602	609	611	**3028**
11	202	202	197	201	195	**997**

Tabel 5.15 Rata-rata waktu akses dalam mili second

FILE	1	2	3	4	5	6	7	8	9	10	11
RATA-RATA	317	213	468	194	314	318	314	504	299	606	199

5.1.4.3 Uji Coba Dengan Module TAR XOR dan Module Gzip

Pada sub bab ini akan di lakukan perbandingan kecepatan proses pengaksesan file TAR yang di encript dengan algoritma XOR. Dan pecobaan ini di lakukan dengan file-file standart yang berukuran kecil.

Tabel 5.16 Waktu Pengaksesan dengan module TAR XOR dan module Gzip

Nama File	Waktu					Total
1	310	321	324	305	300	**1560**
2	200	215	213	203	218	**1049**
3	455	450	445	473	475	**2298**
4	184	202	200	190	190	**966**
5	321	305	310	313	312	**1561**
6	321	320	304	302	305	**1552**
7	324	319	300	296	311	**1550**
8	501	504	505	498	508	**2516**
9	301	287	297	297	299	**1481**
10	601	605	601	606	605	**3018**
11	201	200	196	202	195	**994**

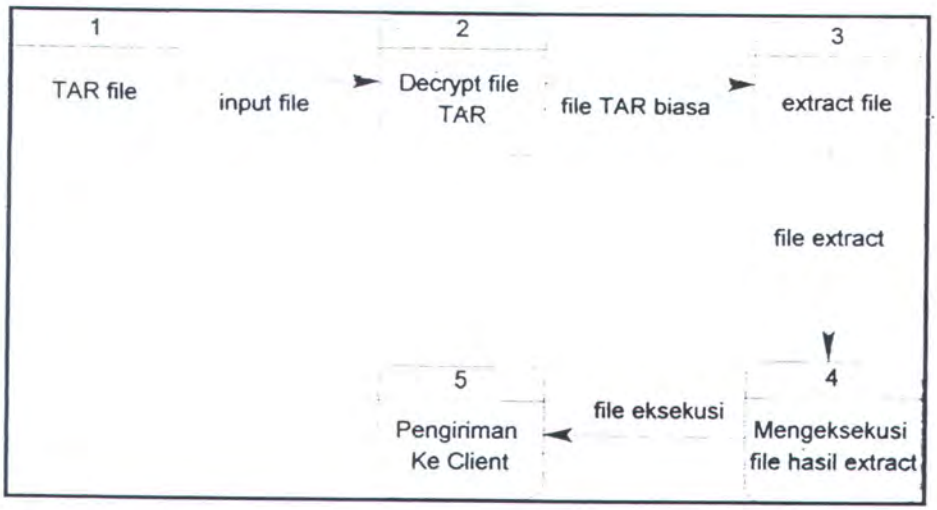
Tabel 5.17 Rata-rata waktu akses dalam mili second

FILE	1	2	3	4	5	6	7	8	9	10	11
RATA-RATA	312	210	460	193	312	310	310	503	296	604	199

5.1.4 Uji Coba Kecepatan Akses Jika Di Encrypt Dulu Baru Di TAR.

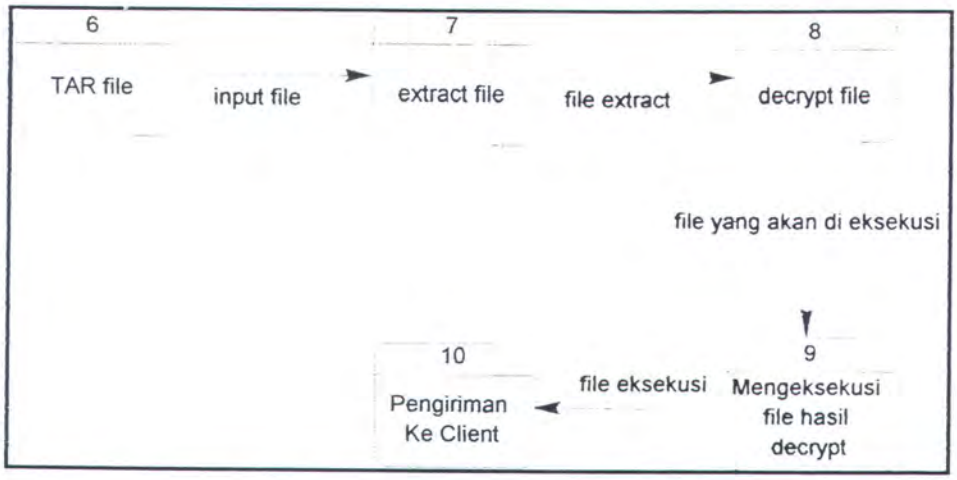
Kecepatan proses akses dari user ke server apabila file TAR yang ada dalam keadaan sudah ter-encrypt, pada bagian ini akan di analisa proses aksesnya dan kecepatannya.

Perbedaan dengan proses yang ada adalah pada proses meng-encrypt file-file yang akan di archieve di mana file-file yang ada di encrypt terlebih dahulu baru kemudian di TAR dan perbedaan yang kedua adalah proses yang ada dalam module TAR, dimana pada module TAR proses men-decrypt file terjadi setelah proses meng-untar file TAR. Dan kemudian file yang di untar tersebut di eksekusi kemudian file hasil eksekusinya di kirim ke client.



Gambar 5.4 Alur TAR pada module tar

Terlihat pada gambar diatas, proses yang terjadi dalam module TAR yaitu proses decrypt file TAR di lakukan terlebih dahulu, kemudian proses selanjudnya adalah proses meng-untar file yang di request oleh client dan proses eksekusi file request kemudian hasil eksekusi tersebut di kirimkan ke client.



Gambar 5.5 Alur Jika di encrypt dulu Kemudian di TAR

Terlihat pada gambar diatas perbedaan yang ada pada gambar sebelumnya adalah proses men-decrypt file terjadi setelah proses mengekstrak file request.

Dan kemudian proses sesudahnya sama dengan proses pada gambar sebelumnya. Kelebihan dari proses encrypt dulu kemudian di TAR dari pada proses TAR dulu kemudian di encrypt adalah jika kita mengencrypt dulu kemudian di TAR maka pada proses men-decrypt file request dapat berjalan lebih cepat karena file yang di decrypt relatif lebih kecil dari pada harus men-decrypt semua file TAR yang mempunyai ukuran lebih besar. Jadi proses meng-encrypt file terlebih dahulu mempunyai kinerja yang lebih baik dari pada membuat file TAR dulu kemudian meng-encrypt.

Kekurangan jika menggunakan proses encrypt dulu kemudian di TAR adalah membutuhkan waktu yang lebih lama dalam meng-encrypt filenya, karena file-file yang akan di archive dalam TAR file harus di encrypt terlebih dahulu semua.

5.2 Lingkungan Uji Coba Tidak Ideal

5.2.1 Uji Coba Kecepatan Keamanan

Pada bagian ini akan di bandingkan kecepatan akses antara algoritma Caesar dan algoritma XOR.

5.2.1.1 Kecepatan Dengan Algoritma Caesar

Tabel 5.18 Waktu Pengaksesan dengan Algoritma Caesar

Nama File	Waktu					Total
1	431	350	400	355	402	**1938**
2	278	263	383	253	300	**1477**
3	535	490	570	585	593	**2773**
4	286	348	538	295	404	**1871**
5	378	430	380	400	335	**1923**
6	374	396	364	385	365	**1884**
7	400	435	338	325	352	**1850**
8	620	564	575	544	587	**2890**
9	448	420	310	380	400	**1958**
10	710	695	659	688	685	**3437**
11	315	363	335	294	390	**1697**

Tabel 5.19 Rata-rata waktu akses dalam mili second

FILE	1	2	3	4	5	6	7	8	9	10	11
RATA-RATA	388	295	555	374	385	377	370	578	391	687	339

5.2.1.2 Kecepatan Dengan Algoritma XOR

Tabel 5.20 Waktu Pengaksesan dengan Algoritma XOR

Nama File	Waktu					Total
1	420	375	488	315	315	**1913**
2	233	303	287	313	335	**1471**
3	505	575	555	585	473	**2693**
4	386	268	308	295	394	**1651**
5	425	310	313	523	315	**1886**
6	434	446	514	415	525	**2334**
7	310	390	328	395	318	**1741**
8	580	614	515	544	597	**2850**
9	348	390	390	410	405	**1943**
10	656	645	808	618	725	**3452**
11	315	260	287	334	308	**1504**

Tabel 5.21 Rata-rata waktu akses dalam mili second

FILE	1	2	3	4	5	6	7	8	9	10	11
RATA-RATA	383	294	539	330	377	467	348	570	389	690	301

5.2.2 Uji Coba Kecepatan Akses Apache

Berikut ini adalah hasil perhitungan waktu dari apache web server tanpa menggunakan module tambahan, yang di hitung adalah acces time, yaitu pencatatan seluruh proses dari awal request sampai akhir selesai proses, waktu dalam ukuran milli second.

Tabel 5.22 Waktu Pengaksesan dengan Apache standart

Nama File	Waktu					Total
1	250	240	215	230	200	**1135**
2	130	125	170	185	195	**805**
3	400	403	410	417	408	**2038**
4	120	108	115	190	132	**665**
5	245	260	280	220	230	**1235**
6	270	245	282	230	262	**1289**
7	270	250	285	300	240	**1345**
8	454	481	487	438	440	**2300**
9	270	297	205	212	251	**1235**
10	580	565	580	545	570	**2840**
11	172	153	127	130	125	**707**

Tabel 5.23 Rata-rata waktu akses dalam mili second

FILE	1	2	3	4	5	6	7	8	9	10	11
RATA-RATA	227	161	408	133	247	258	269	460	247	568	141

5.2.3 Uji Coba Kecepatan Akses Apache Dengan Module TAR

Pada bagian ini akan di lakukan uji coba dengan berbagai sekenario, di antaranya adalah sebagai berikut.

5.2.3.1 Uji coba Pada File Berukuran Kecil

Pada uji coba menggunakan apache web server yang dengan menambah module tar, sekenario percobaannya masih sama dengan sekenario percobaan diatas, file yang di akses juga sama. Penghitungan waktu untuk percobaan menggunakan module tar ini adalah sebagai berikut:

Tabel 5.24 Waktu Pengaksesan dengan Apache yang di tambah module

Nama File	Waktu					Total
1	400	355	448	300	305	**1808**
2	230	300	287	310	320	**1447**
3	495	555	546	565	470	**2631**
4	366	258	300	294	390	**1608**
5	420	310	310	483	315	**1838**
6	430	436	474	410	485	**2235**
7	311	370	325	386	316	**1708**
8	560	592	515	544	537	**2748**
9	338	380	384	400	403	**1905**
10	656	645	708	618	702	**3329**
11	312	240	267	312	308	**1439**

Tabel 5.25 Rata-rata waktu akses dalam mili second

FILE	1	2	3	4	5	6	7	8	9	10	11
RATA-RATA	362	289	526	322	368	447	342	550	381	666	288

5.2.3.2 Uji Coba Pada File Berukuran Besar.

Pada sekenario berikut ini, akan dilakukan pengukuran kecepatan akses dengan menggunakan file-file yang berukuran besar secara berurutan. File-file tersebut akan di ukur kecepatan pengaksesannya dan akan dilakukan perbandingan antar filenya, penghitungannya di lakukan sebanyak kurang lebih 3 kali percobaan.

Tabel 5.26 Pengukuran file besar

Besar File	**Waktu Pengaksesan**		
100 KB	3.382	2.582	3.322
200 KB	3.852	4.834	4.362
300 KB	5.244	3.115	2.334
400 KB	5.896	5.912	4.762
500 KB	6.49	6.51	6.272
600 KB	7.624	7.985	8.42
700 KB	9.204	8.21	9.431
800 KB	11.302	12.251	11.382
900 KB	13.956	14.05	13.06
1 MB	15.624	15.52	14.981
1100 KB	16.731	18.642	17.801
1200 KB	19.199	19.299	18.459
1300 KB	20.22	20.223	20.321
1400 KB	22.354	23.21	21.481
1500 KB	24.11	23.725	24.41
1600 KB	26.69	26.769	25.288
1700 KB	27.611	28.251	27.432
1800 KB	29.15	28.313	28.282
1900 KB	33.511	32.52	31.45
2 MB	34.824	35.912	35.11

Dari hasil pengukuran kecepatan akses file besar maka di dapat data seperti di atas, dan rata-rata kecepatan akses adalah sebagai berikut :

Kecepatan pada ukuran file 100 KB	:	3.095
Kecepatan pada ukuran file 200 KB	:	4.349
Kecepatan pada ukuran file 300 KB	:	3.564
Kecepatan pada ukuran file 400 KB	:	5.523
Kecepatan pada ukuran file 500 KB	:	6.424
Kecepatan pada ukuran file 600 KB	:	8.010
Kecepatan pada ukuran file 700 KB	:	8.948
Kecepatan pada ukuran file 800 KB	:	11.645
Kecepatan pada ukuran file 900 KB	:	13.689
Kecepatan pada ukuran file 1 MB	:	15.375
Kecepatan pada ukuran file 1100 KB	:	17.725
Kecepatan pada ukuran file 1200 KB	:	18.986
Kecepatan pada ukuran file 1300 KB	:	20.255
Kecepatan pada ukuran file 1400 KB	:	22.348
Kecepatan pada ukuran file 1500 KB	:	24.082
Kecepatan pada ukuran file 1600 KB	:	26.249
Kecepatan pada ukuran file 1700 KB	:	27.765
Kecepatan pada ukuran file 1800 KB	:	28.582
Kecepatan pada ukuran file 1900 KB	:	32.494
Kecepatan pada ukuran file 2 MB	:	35.282

5.2.4 Uji Coba Dengan Module Gzip.

Uji coba perbandingan apache web server dengan menggunakan mod_gzip dan apache tampa menggunakan mod_gzip, sekenario percobaannya adalah pada file apache web server di letakkan file yang berukuran cukup besar kurang lebih 1.4 MB, dan pada jaringan lokal di ukur kecepatan down load file tersebut.

Uji coba pertama dengan menggunakan apache web server tanpa module g_zip, file di download sebanyak tiga kali, dan uji coba kedua dengan tambahan module g_zip maka di dapat hasil (dalam milli second)

Tabel 5.27 Perbandingan dengan Mod_gzip dan tanpa Mod_gzip

Kondisi	**Waktu**		
Tanpa mod_gzip	790	630	570
Dengan mod_gzip	620	310	460

Maka rata-rata download file tanpa module gzip *adalah*

(790 + 630 + 470) / 3 = 663.33 milli second

Sedangkan rata-rata download file dengan module gzip *adalah*

(620 + 310 + 460) / 3 = 463,33 milli second

5.2.4.1 Uji Coba Dengan Module TAR dan Module Gzip

Pada sub bab ini akan di lakukan perbandingan kecepatan proses pengaksesan file TAR jika Apache server di install module Gzip . Dan pecobaan ini di lakukan dengan file-file standart yang berukuran kecil.

Tabel 5.28 Waktu Pengaksesan dengan module TAR dan module Gzip

Nama File	Waktu					Total
1	398	355	415	300	305	**1773**
2	230	300	284	310	306	**1430**
3	495	551	546	552	470	**2614**
4	363	258	300	294	376	**1591**
5	415	311	310	463	315	**1814**
6	430	434	444	410	473	**2191**
7	311	362	325	376	316	**1690**
8	560	557	515	544	536	**2712**
9	338	380	374	400	391	**1883**
10	636	645	708	618	670	**3277**
11	312	240	257	302	308	**1419**

Tabel 5.29 Rata-rata waktu akses dalam mili second

FILE	1	2	3	4	5	6	7	8	9	10	11
RATA-RATA	355	286	523	318	363	438	338	542	377	655	284

5.2.4.2 Uji Coba Dengan Module TAR Caesar dan Module Gzip

Pada sub bab ini akan di lakukan perbandingan kecepatan proses pengaksesan module TAR yang di encript dengan algoritma caesar dan module Gzip. Dan pecobaan ini di lakukan dengan file-file standart yang berukuran kecil.

Tabel 5.30 Waktu Pengaksesan dengan module TAR Caesar dan module Gzip

Nama File	Waktu					Total
1	401	340	400	355	402	**1898**
2	278	263	363	253	300	**1457**
3	535	490	570	582	573	**2750**
4	286	348	408	295	404	**1741**
5	378	399	380	400	335	**1892**
6	374	366	364	375	365	**1844**
7	400	411	338	325	352	**1826**
8	520	564	575	544	587	**2790**
9	428	420	310	380	400	**1938**
10	610	695	659	688	685	**3337**
11	315	343	335	294	390	**1677**

Tabel 5.31 Rata-rata waktu akses dalam mili second

FILE	1	2	3	4	5	6	7	8	9	10	11
RATA-RATA	380	291	550	348	378	369	365	558	388	667	335

5.2.4.3 Uji Coba Dengan Module TAR XOR dan Module Gzip

Pada sub bab ini akan di lakukan perbandingan kecepatan proses pengaksesan file TAR yang di encript dengan algoritma XOR. Dan pecobaan ini di lakukan dengan file-file standart yang berukuran kecil.

Tabel 5.32 Waktu Pengaksesan dengan module TAR XOR dan Module Gzip

Nama File	Waktu					Total
1	420	375	468	315	315	**1893**
2	233	303	287	313	312	**1448**
3	505	565	555	535	473	**2633**
4	383	268	308	295	374	**1628**
5	415	310	313	503	315	**1856**
6	434	446	504	415	500	**2299**
7	310	390	328	375	318	**1721**
8	580	601	515	544	577	**2817**
9	348	390	376	391	392	**1897**
10	656	645	708	618	725	**3352**
11	315	260	287	301	290	**1453**

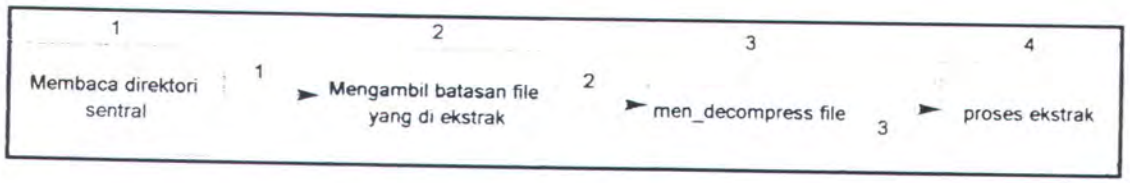
Tabel 5.33 Rata-rata waktu akses dalam mili second

FILE	1	2	3	4	5	6	7	8	9	10	11
RATA-RATA	379	290	527	326	371	460	344	563	379	670	291

5.3 Uji Coba Kecepatan Akses Apache Dengan ZIP File.

Pada bagian ini akan di bahas, apabila file yang akan di akses berupa file zip kemudian file tersebut di encrypt dan diletakkan di pada web server untuk di akses.

Langkah awal untuk mengakses adalah men-decript file ZIP yang telah di encript, kemudian langkah berikutnya adalah meng-ekstrak file ZIP, maka proses atau langkah-langkah yang harus di lakukan adalah membaca directori sentral yang berada di akhir file ZIP, kemudian pada direktori sentral tersebut mengambil informasi-informasi berupa letak atau posisi file yang akan di request, kemudian langkah selanjutnya adalah meng-ekstrak filenya. Akan tetapi, proses sebelumnya adalah men-decompress file, atau prosesnya dapat di lihat pada diagram alur di bawah ini.



Gambar 5.6 Alur Ekstrak ZIP file

Seperti di lihat pada diagram alur di atas, dimana file yang zip yang telah di decript, kemudian di ekstrak, dengan mengambil informasi-informasi yang di perlukan untuk proses ekstraknya, kemudian mendecompres file ZIP tersebut dan proses ekstrak file yang di request selesai.

Jika module yang di buat adalah berupa file ZIP maka proses akan lebih efisien terutama pada proses mengekstrak file request, karena jika menggunakan file TAR proses ekstrak tergantung dari letak file request pada file TAR.

5.4 Evaluasi

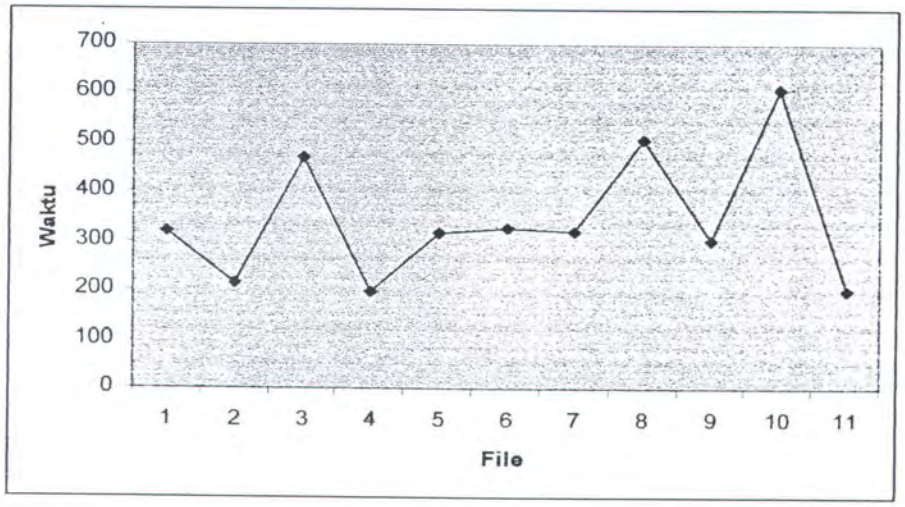
Pada bagian ini akan di lakukan evaluasi pada hasil uji coba yang di lakukan, dan hasilnya kemudian di lakukan perbandingan dan analisanya.

5.4.1 Pada Keadaan Ideal

5.4.1.1 Keamanan Source Dan Kecepatan TAR Yang Di Encript

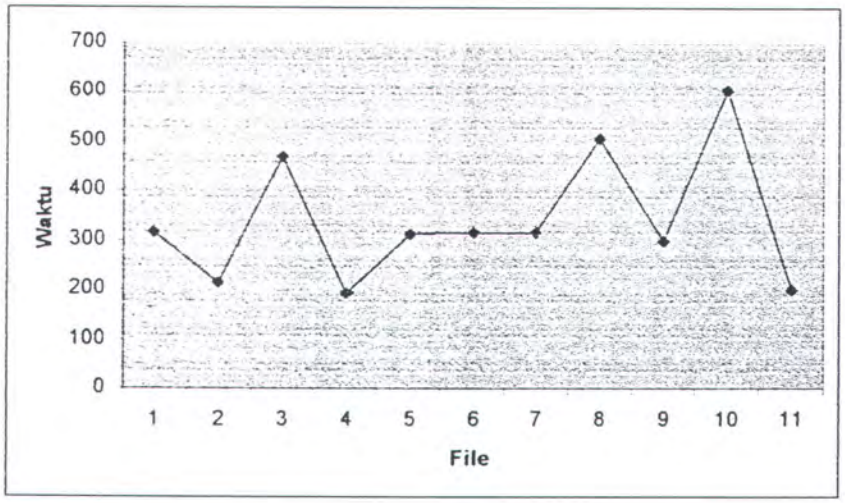
Dengan menggunakan module ini, maka file php dan html akan lebih aman karena file tersebut nantinya di tar dan kemudian di-encrypt sehingga jika tidak mempunyai program untuk men-decrypt maka file tar tersebut tidak dapat di untar dan di lihat source code yang ada didalam file tar tersebut.

5.4.1.1.1 Algoritma Caesar

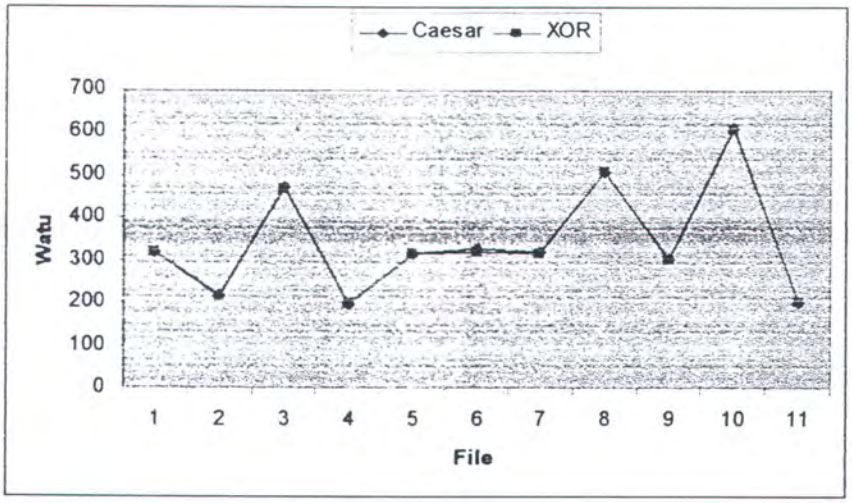


Gambar 5.7 Diagram Kecepatan akses dengan encripsi Caesar

5.4.1.1.2 Algoritma XOR



Gambar 5.8 Diagram Menggunakan Algoritma XOR

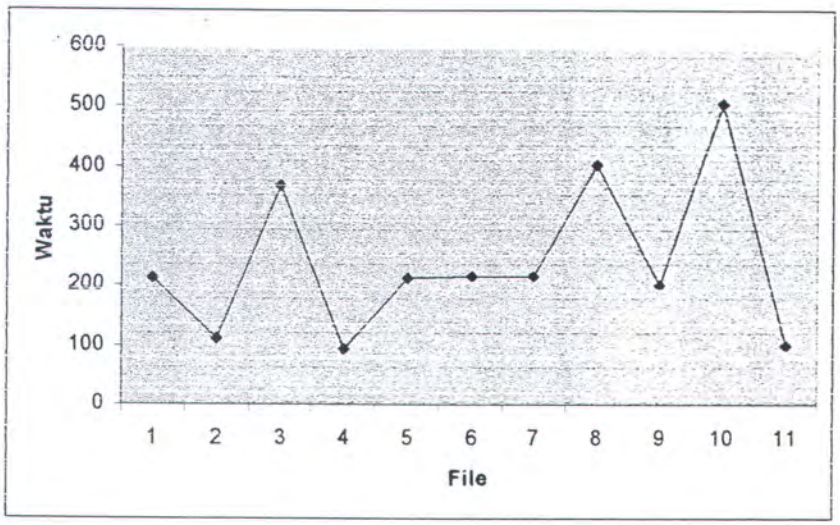


Gambar 5.9 Diagram Perbandingan Algoritma Caesar dan XOR

Seperti terlihat pada perbandingan kecepatan akses file TAR yang di encript menggunakan algoritma Caesar dan XOR, tidak terlihat perbedaan yang signifikan sekali, akan tetapi penggunaan algoritma XOR mempunyai kecenderungan lebih cepat.

Hal ini di karenakan penggunaan variabel pada algoritma Caesar, tidak terlalu banyak, sehingga operasi bit dan operasi aritmatika pada algoritma caesar menghasilkan waktu akses yang hampir tidak berbeda.

5.4.1.2 Kecepatan Pengaksesan Apache standart.



Gambar 5.10 Diagram Kecepatan akses Apache standart

5.4.1.3 Kecepatan Pengaksesan Module TAR.

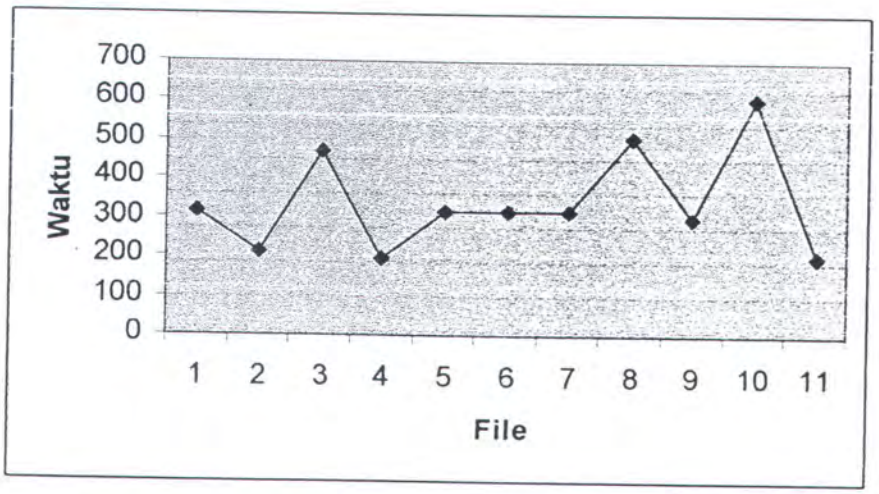
Pada bagian ini akan di bahas analisa kecepatan pengaksesan dengan menggunakan module tar. Dan di coba pada berbagai macam jenis dan ukuran file.

5.4.1.3.1 Pada File Kecil.

Dari perbandingan percobaan pada sub bab sebelumnya tampak perbedaan waktu yang tidak terlalu signifikan pada pengaksesan file text, terutama file yang

tidak melakukan permintaan ulang pada server, untuk lebih jelasnya, akan di bandingkan waktu akses masing-masing halaman dari data yang di hasilkan.

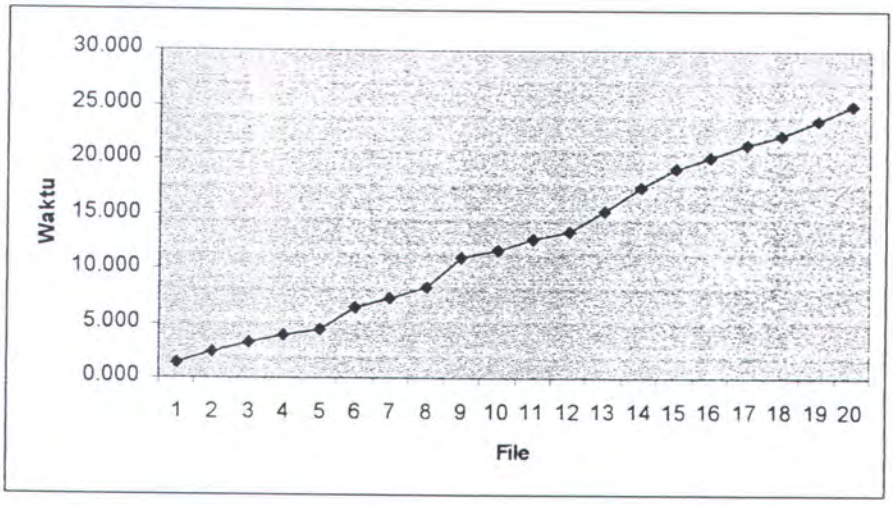
Pada halaman pertama, tampak perbedaan waktu yang tidak terlalu signifikan, hal ini di karenakan file yang di minta ke server berukuran kecil dan hanya file text, sehingga prosesnya lebih cepat. Semakin banyak file image maka akan membuat semakin lambat, hal ini di sebabkan file html akan me-*request* lagi ke server sehingga membutuhkan waktu untuk proses untar file yang di *request* oleh browser dan mengirimkannya ke browser. Hal ini yang akan menyebabkan waktu akhir menjadi lebih lama.Tampak pada file 7 adalah file yang paling lambat pengaksesannya, hal ini di sebabkan karena file tersebut melakukan request gambar pada server, akan tetapi file tersebut tidak ada dalam file tar.



Gambar 5.11 Diagram Perbandingan Kecepatan apache dengan module

5.4.1.3.2 Pada File Besar.

Pada percobaan ini, dapat terlihat perbedaan pada tiap ukuran file yang di coba. Berikut ini adalah gambar-gambar grafik kecepatan akses dari tiap filenya.



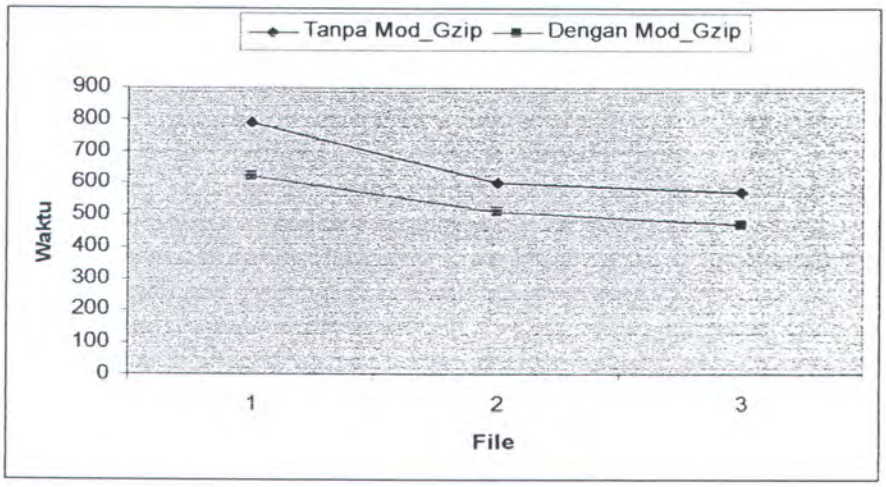
Gambar 5.12 Perbandingan Pengaksesan File Besar

Pada gambar di atas terlihat perbedaan antara waktu akses file satu dengan lainnya. Dan perbedaan ukurannya berbanding lurus dengan kecepatan aksesnya.

Hal ini di sebabkan karena setiap kali mengakses file request terlebih dahulu harus mengencrypt seluruh file TAR dan proses meng-untar file dengan membaca file TAR tersebut tiap bloknya, sehingga apabila file yang akan di decript mempunyai ukuran yang semakin besar maka membutuhkan waktu yang semakin lama. Dan proses membaca file tiap bloknya juga semakin lama, karena tiap kali akan mengekstrak file pada file TAR maka harus membaca blok file TAR, sampai di temukan file yang akan di UNTAR.

5.4.1.4 Menggunakan Module Gzip.

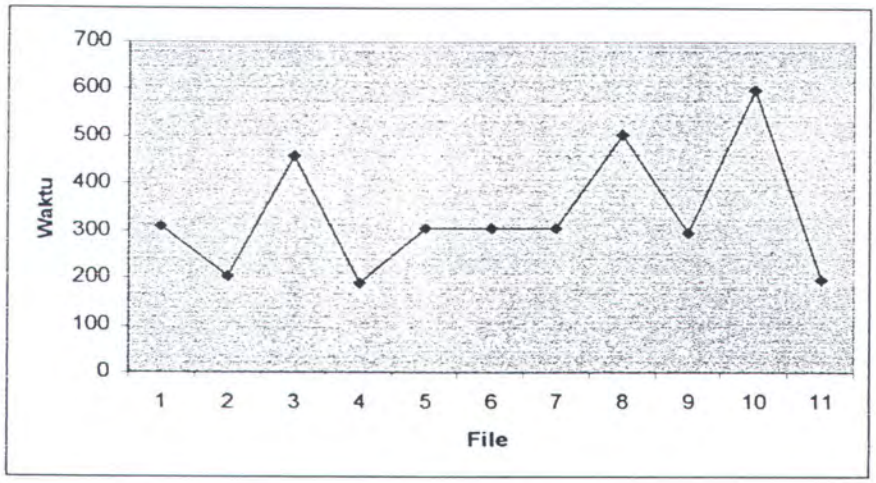
Pada uji coba dengan menggunakan module gzip dan tidak menggunakan module gzip, dapat di lihat hasil percobaannya, dimana jika menggunakan module gzip maka terdapat perbedaan waktu dimana jika menggunakan module gzip maka kecepatan aksesnya lebih cepat dari pada tanpa menggunakan module gzip.



Gambar 5.13 Perbandingan Module Gzip dan tanpa Mod_gzip

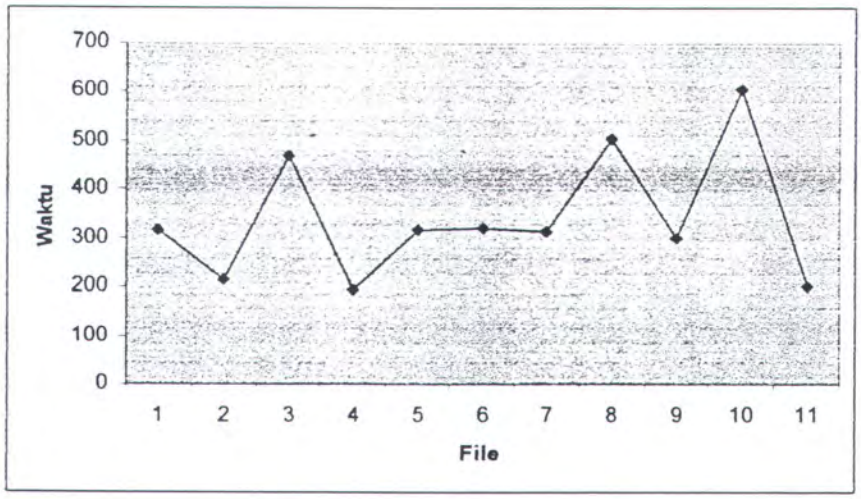
Penggunaan modul ini pada Apache web server dapat mempercepat waktu akses, terutama untuk aplikasi dengan struktur jaringan yang kurang baik, di mana proses transfer file menjadi masalah yang banyak terjadi.

5.4.1.4.1 Dengan Module TAR dan Module Gzip



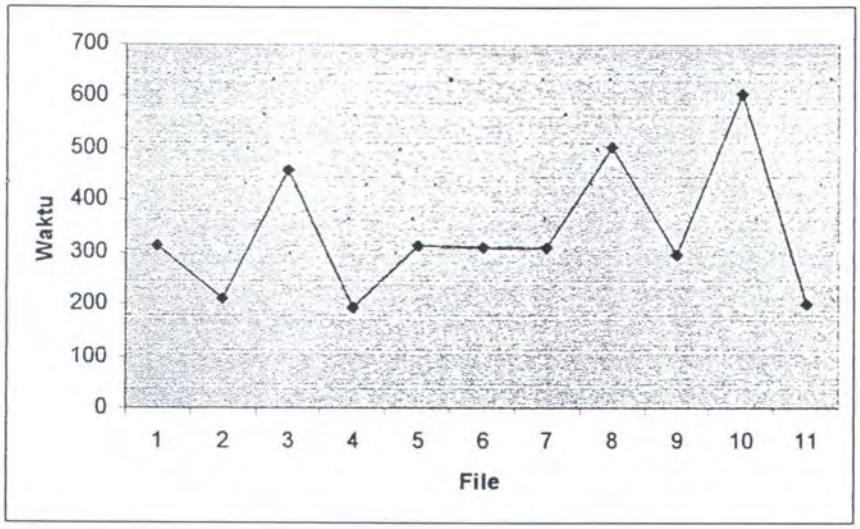
Gambar 5.14 Diagram Waktu akses module TAR dengan Gzip

5.4.1.4.2 Dengan Module TAR Caesar dan Module Gzip



Gambar 5.15 Diagram Waktu akses module TAR Caesar dengan Gzip

5.4.1.4.3 Dengan Module TAR XOR dan Module Gzip



Gambar 5.16 Diagram Waktu akses module TAR XOR dengan Gzip

5.4.1.5 Kecepatan Pengaksesan Di Encrypt Kemudian Di TAR

Kecepatan pengaksesan apabila file-filenya di encrypt dulu kemudian file-filenya di archive dengan format TAR, maka proses pengaksessan dengan menggunakan metode ini lebih cepat karena ukuran file yang akan di decrypt lebih kecil sehingga proses men-decrypt file tersebut lebih cepat.

Akan tetapi akan memakan waktu yang lebih lama dan kurang efisien dalam proses pembuatan file TAR-nya karena file-file yang akan di archive di encrypt satu-satu dan setelah itu di archive dengan format TAR.

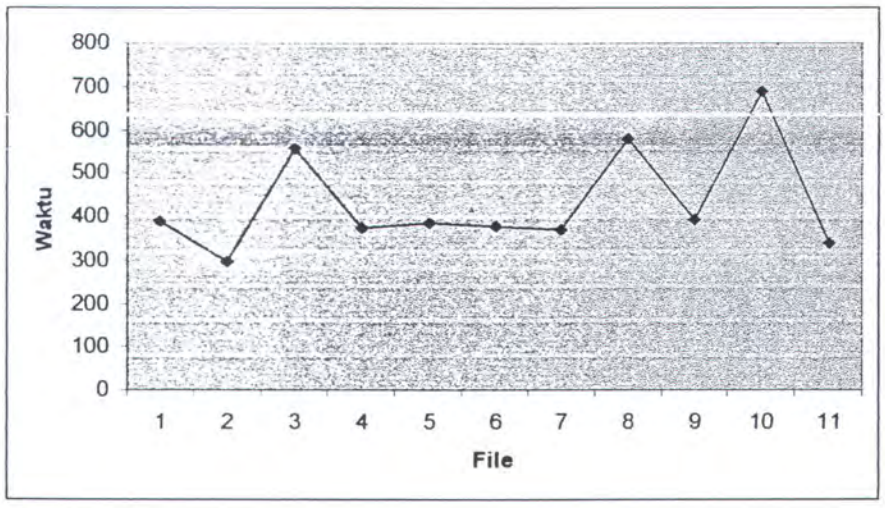
5.4.2 Pada Keadaan Tidak Ideal

Pada bagian ini akan di lakukan percobaan pengaksesan module TAR pada jaringan yang melewati beberapa node. Dimana terdapat faktor dari luar yang akan mempengaruhi performa dari module TAR. Sehingga dapat di ketahui faktor yang lebih mempengaruhi penggunaan module TAR ini.

5.4.2.1 Kecepatan TAR Yang Di Encript

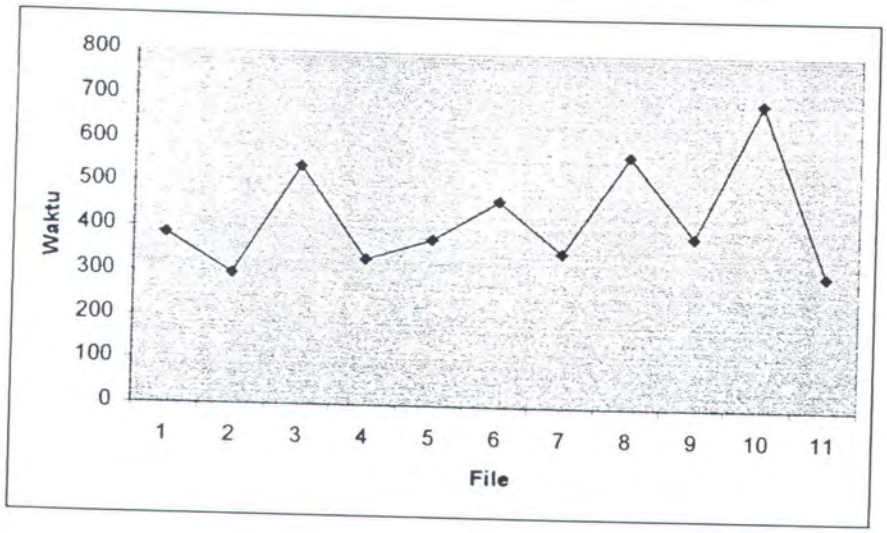
Perbandingkan antara algoritma Caesar dan XOR dalam diagram dari hasil pengukuran kecepatan.

5.4.2.1.1 Algoritma Caesar

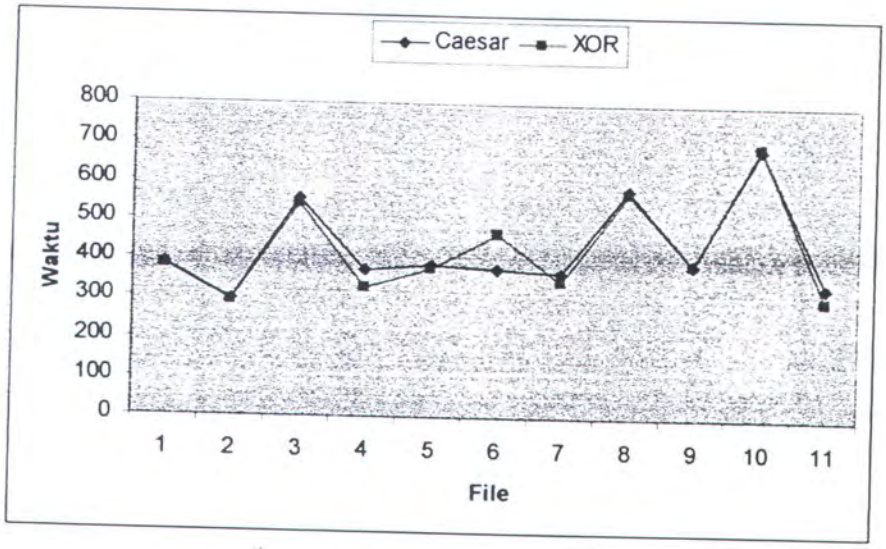


Gambar 5.17 Diagram Algoritma Caesar

5.4.2.1.2 Algoritma XOR



Gambar 5.18 Diagram Algoritma XOR

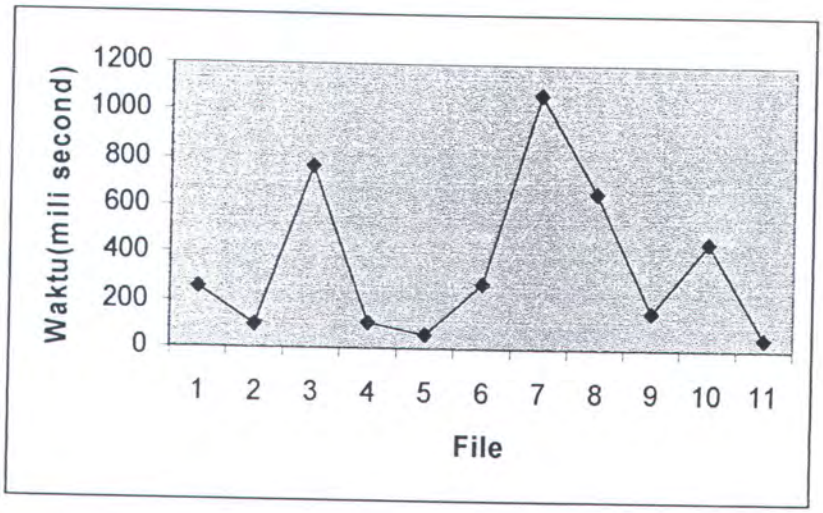


Gambar 5.19 Diagram Perbandingan Algoritma Caesar dan XOR

Jika di lihat dari perbandingan hasil percobaan antara algoritma Caesar dan XOR pada lingkungan tidak ideal, maka hasil uji kecepatan akses yang dilakukan hasilnya tidak sama, atau tidak menunjukkan satu kecenderungan

tertentu. Hal ini di sebabkan adanya faktor luar yang mempengaruhi proses kinerja module TAR.

5.4.2.2 Kecepatan Pengaksesan Apache standart

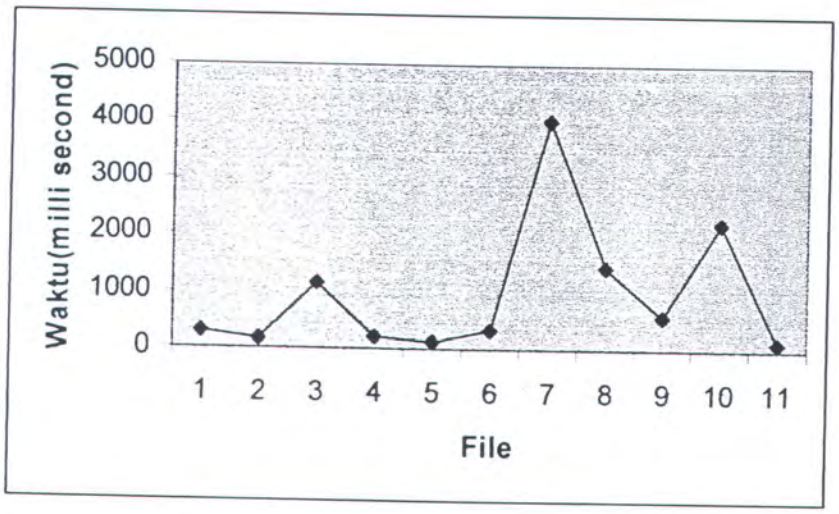


Gambar 5.20 Diagram Perbandingan Kecepatan apache standart

5.4.2.2 Kecepatan Pengaksesan Module TAR.

Pada bagian ini akan di bahas analisa kecepatan pengaksesan dengan menggunakan module tar. Dan di coba pada berbagai macam jenis dan ukuran file.

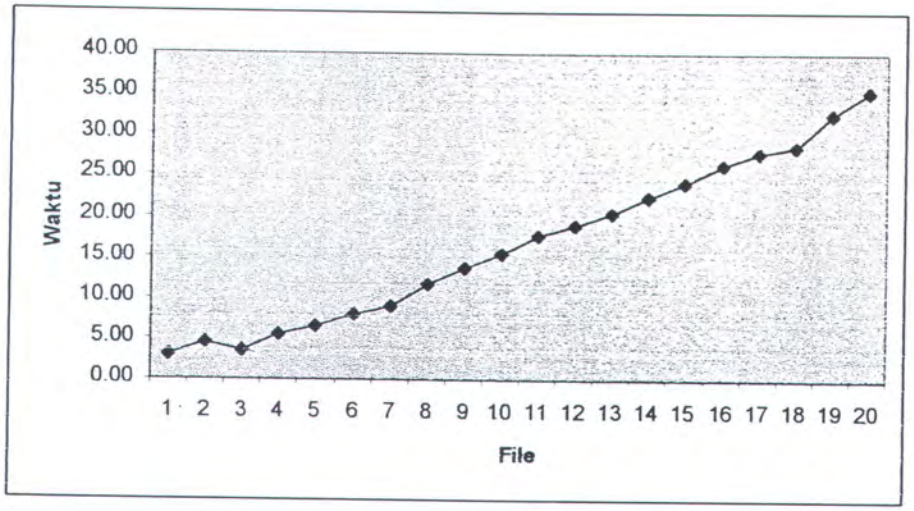
5.4.2.2.1 Pada File Kecil.



Gambar 5.21 Diagram Kecepatan dengan module TAR

Terlihat pada perbandingan waktu akses, dimana terjadi perbedaan yang cukup besar, jika tidak menggunakan module TAR tambahan, hal ini di sebabkan adanya faktor luar yang mempengaruhi proses kecepatan akses.

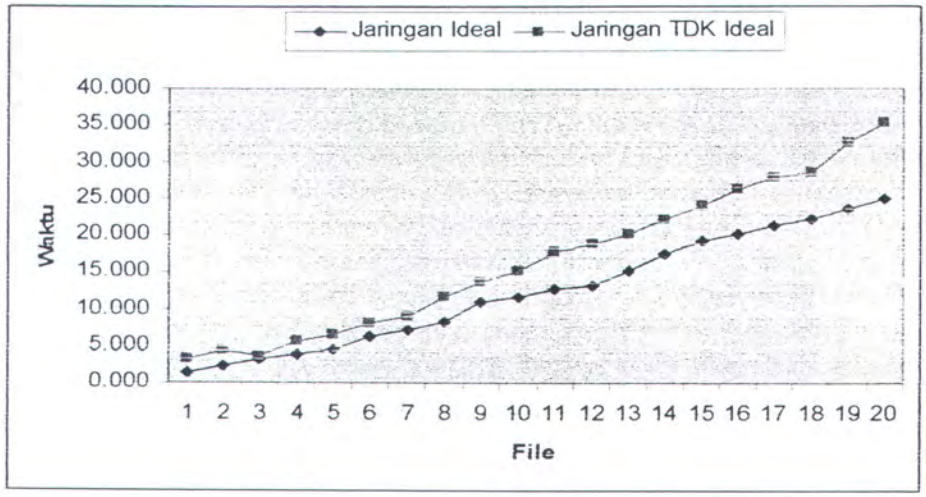
5.4.2.2.2 Pada File Besar.



Gambar 5.22 Akses File Besar Pada Keadaan Tidak Ideal

Terlihat dari diagram diatas, dimana kecepatan akses file secara keseluruhan menunjukkan kecenderungan semakin lama waktu aksesnya jika file yang di akses semakin besar.

Akan tetapi secara keseluruhan, semakin besar ukuran file TAR yang diakses maka semakin membutuhkan waktu untuk mengaksesnya.

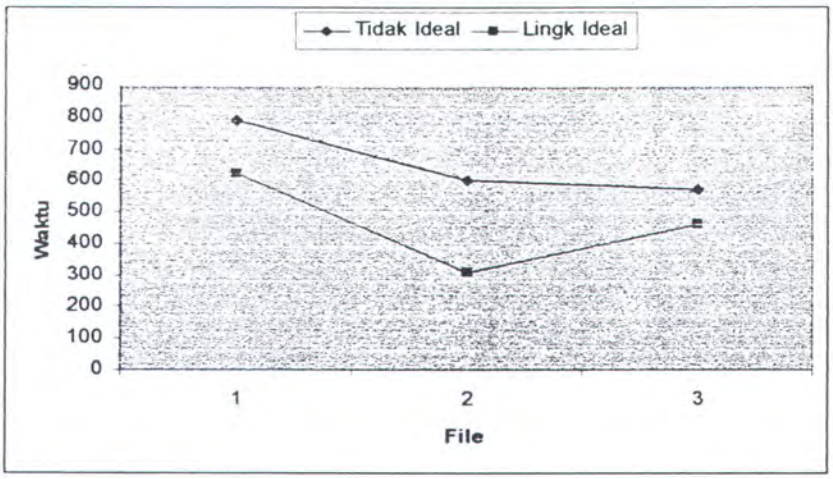


Gambar 5.23 Perbandingan Pada Jaringan Ideal Dan Tidak Ideal

Pada diagram diatas, dimana terlihat perbandingan waktu akses yang antara jaringan ideal dan jaringan tidak ideal. Dimana terlihat jaringan yang tidak ideal membutuhkan waktu akses yang lebih lama dari pada waktu akses pada jaringan ideal.

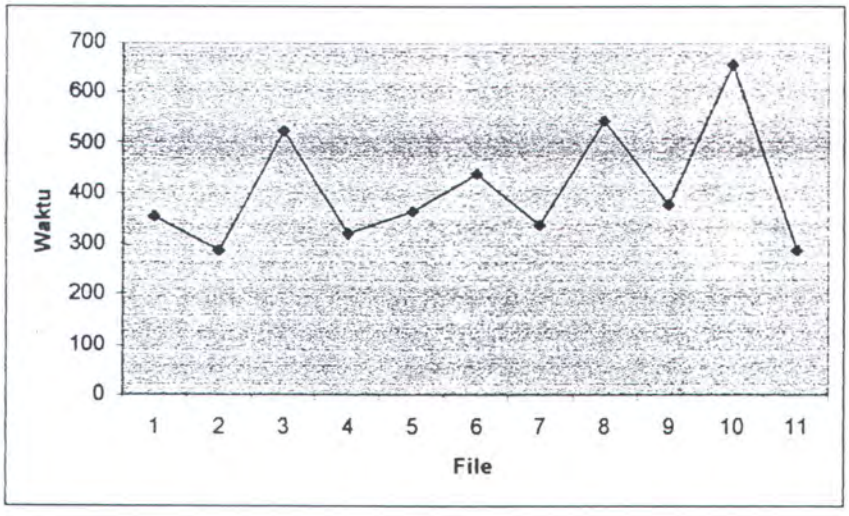
Akan tetapi secara keseluruhan tampak bahwa waktu akses file yang mempunyai ukuran lebih besar akan membutuhkan waktu yang lebih lama juga.

5.4.2.3 Menggunakan Module Gzip.



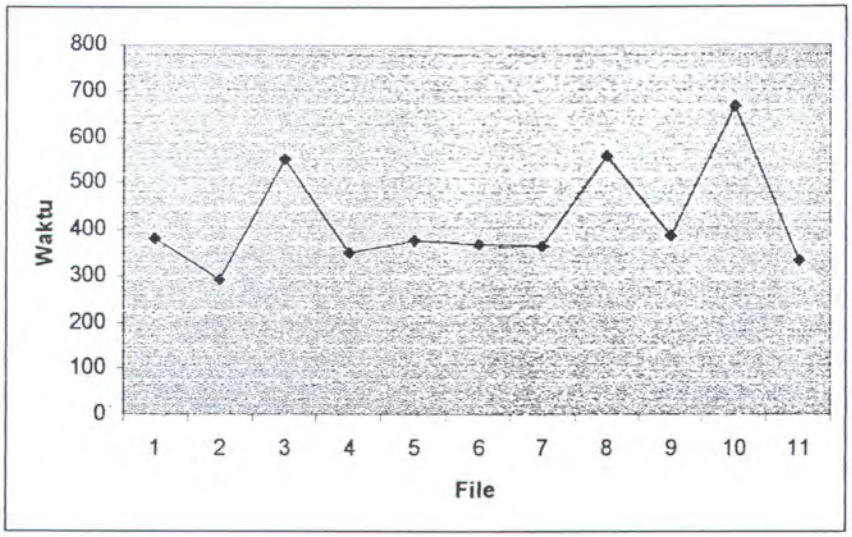
Gambar 5.24 Perbandingan Menggunakan Mod_gzip dan tanpa Mod_gzip

5.4.2.3.1 Dengan Module TAR dan Module Gzip



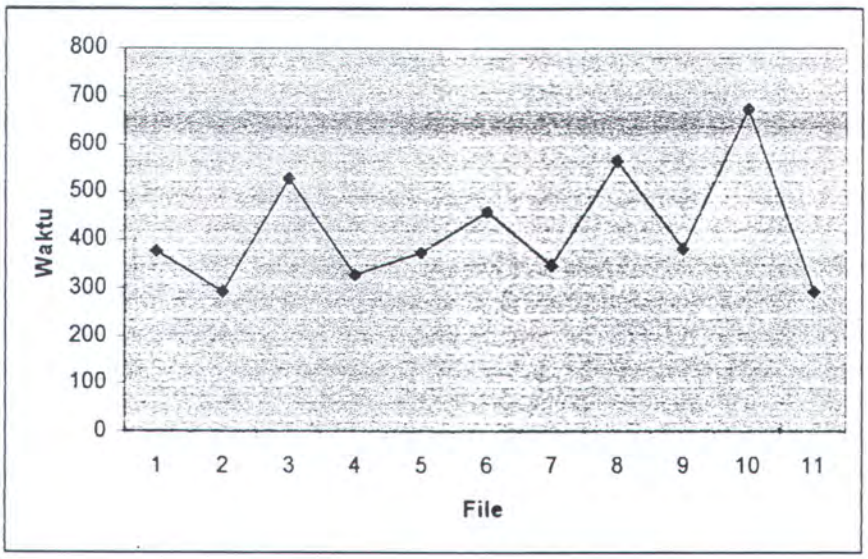
Gambar 5.25 Diagram Waktu akses module TAR dengan Gzip

5.4.2.3.2 Dengan Module TAR Caesar dan Module Gzip



Gambar 5.26 Diagram Waktu akses module TAR Caesar dengan Gzip

5.4.2.3.3 Dengan Module TAR XOR dan Module Gzip



Gambar 5.27 Diagram Waktu akses module TAR XOR dengan Gzip

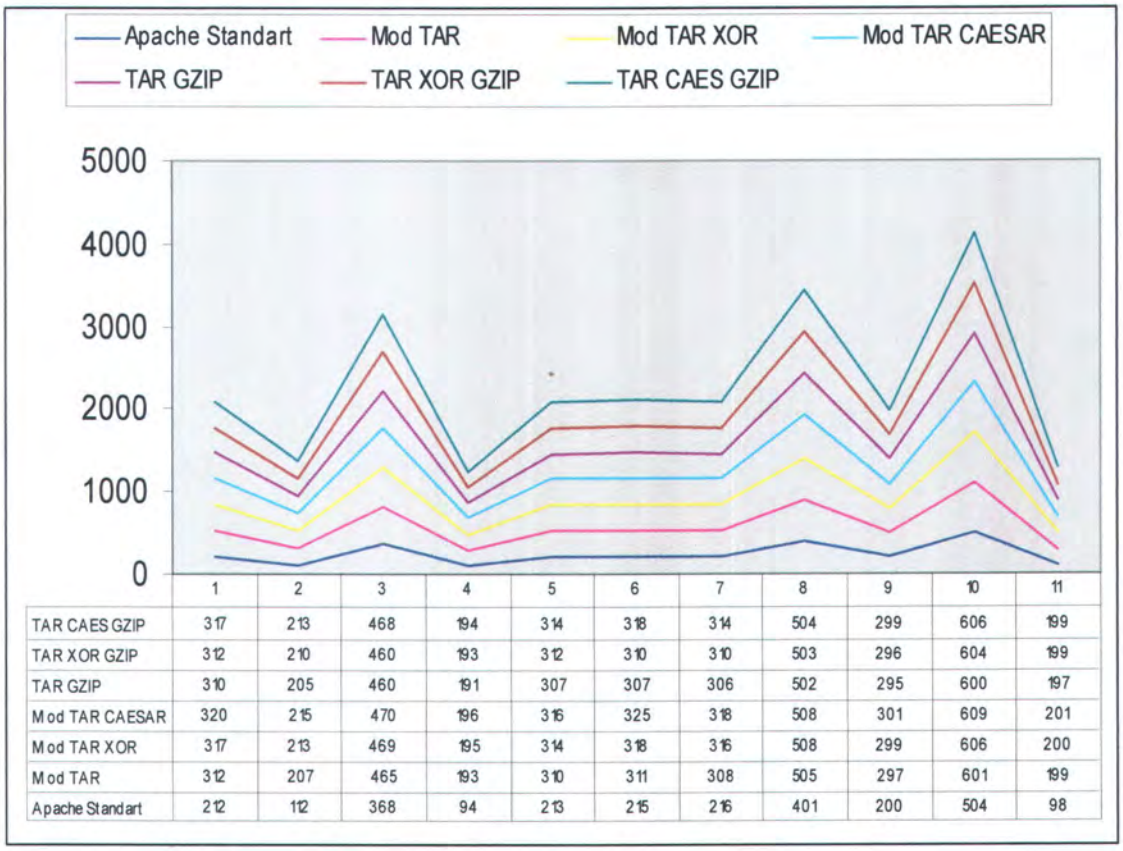
5.4.3 Kecepatan Akses Apache Dengan ZIP File.

Pada module tambahan yang menggunakan format TAR, proses yang cukup memakan waktu adalah pada proses men-decript file TAR dan proses meng-UNTAR file yang di request.

Proses ekstrak file dengan menggunakan ZIP format dapat berjalan lebih efektif karena proses pengaksesan yang terjadi hanya sebatas file yang akan di ekstrak saja. Dalam arti, jika kita ingin mengekstrak satu file dalam file ZIP maka yang akan di baca hanya file yang berada di daerah file yang akan di ekstrak.

Lain halnya dengan menggunakan format TAR, kita harus membaca semua file TAR, jadi apabila file yang akan di ekstrak berada di tengah atau akhir file TAR maka akan membutuhkan waktu yang lebih lama.

5.4.4 Diagram Perbandingan Tiap-tiap Percobaan



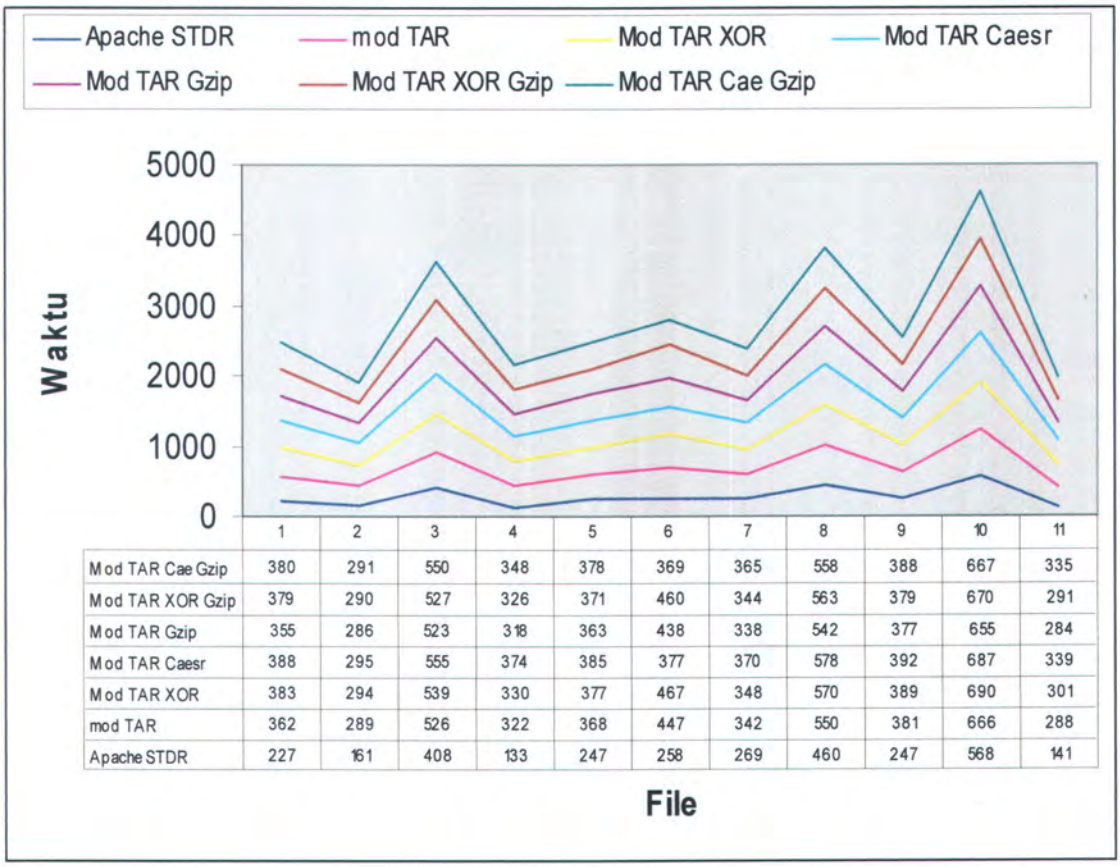
	1	2	3	4	5	6	7	8	9	10	11
TAR CAES GZIP	317	213	468	194	314	318	314	504	299	606	199
TAR XOR GZIP	312	210	460	193	312	310	310	503	296	604	199
TAR GZIP	310	205	460	191	307	307	306	502	295	600	197
Mod TAR CAESAR	320	215	470	196	316	325	318	508	301	609	201
Mod TAR XOR	317	213	469	195	314	318	316	508	299	606	200
Mod TAR	312	207	465	193	310	311	308	505	297	601	199
Apache Standart	212	112	368	94	213	215	216	401	200	504	98

Gambar 5.28 Diagram Perbandingan semua Percobaan Kondisi Ideal

Jika di lihat dari diagram dan tabel perbandingan antara data-data yang ada maka dapat di lihat perbedaan kecepatan akses pada penggunaan module apache yang standart dengan menggunakan module TAR, begitu juga pengaksesan pada module TAR yang di encripsi, mempunyai waktu akses yang lebih lama dari penggunaan module TAR tanpa encript.

Sedangkan dari data yang ada pada waktu akses module TAR yang di encript dengan module TAR yang di encript dengan menambahkan module Gzip,

dapat dilihat bahwa pada penambahan module Gzip, dapat memberikan waktu akses yang lebih cepat dari pada tanpa menggunakan module Gzip.



	1	2	3	4	5	6	7	8	9	10	11
Mod TAR Cae Gzip	380	291	550	348	378	369	365	558	388	667	335
Mod TAR XOR Gzip	379	290	527	326	371	460	344	563	379	670	291
Mod TAR Gzip	355	286	523	318	363	438	338	542	377	655	284
Mod TAR Caesr	388	295	555	374	385	377	370	578	392	687	339
Mod TAR XOR	383	294	539	330	377	467	348	570	389	690	301
mod TAR	362	289	526	322	368	447	342	550	381	666	288
Apache STDR	227	161	408	133	247	258	269	460	247	568	141

Gambar 5.29 Diagram Perbandingan Semua Percobaan Kondisi tidak ideal

BAB VI

PENUTUP

BAB VI

PENUTUP

Pada bab ini berisi, kesimpulan dari pembuatan tugas akhir ini serta berisi saran untuk kemungkinan pengembangan sistem lebih lanjut.

6.1 KESIMPULAN

Berdasarkan pada perancangan dan pembuatan sistem terhadap permasalahan yang diangkat, maka dapat diambil kesimpulan sebagai berikut :

1. Enkripsi file menggunakan algorithma Caesar atau menggunakan algorithma XOR dapat memberikan keamanan yang lebih pada script PHP dan HTML.
2. Dengan membuatan module pada apache, dalam hal ini module TAR, maka server dapat membaca URL yang telah dimodifikasi pada browser yang diinputkan oleh client.
3. Dengan penggunaan module Gzip pada server maka proses transfer file dapat berjalan lebih cepat dan menghemat bandwith karena file yang dikirim terlebih dahulu dikompresi.
4. Penggabungan ketiga module di atas, yaitu algoritma enkripsi, module TAR apache, dan module Gzip, dapat membuat server bekerja lebih efektif, yaitu dengan penambahan waktu yang tidak terlalu lama dapat menghasilkan sistem keamanan yang lebih baik.

6.2 SARAN

Berikut ini adalah saran untuk kemungkinan pengembangan lebih lanjut dari hasil perancangan dan pembuatan prototipe aplikasi sistem dalam tugas akhir ini :

1. Penambahan module untuk aplikasi web dinamis, seperti asp, perl dan lainnya, sehingga module ini dapat digunakan pada berbagai jenis aplikasi tersebut.
2. Pengembangan sistem keamanan, yaitu dengan membuat sistem keamanan menggunakan algoritma yang lain, misalnya algoritma DES atau yang lain.

DAFTAR PUSTAKA

DAFTAR PUSTAKA

1. Ady Wicaksono, *Membangun Jaringan Itu Mudah*, Info komputer Maret, PT. Gramedia, 2003.
2. *Apache manual and documentation*, www.apache.org, 1999.
3. Bose, Ranjan, *Information Theory Coding And Cryptography*, McGraw-Hill, 2003.
4. *DSO apache documentation*, http://zend.com/apidoc/zend.layout.php, 1999.
5. *GNU tar Documentation*, www.delorie.com/gnu/docs/tar/tar_toc.html, 1998.
6. Jesus Castagnetto, Harish Rawat Dkk, *Professional PHP Programing*, Wrox Press, 1999.
7. Kurt Wall, Mark Watson, and Mark Whitis, *Linux Programing Unleashed*, Sams, 1999.
8. *Mod_gzip on Linux server*, http://telia.dl.sourceforge.net, 2000.
9. *PHP Hypertext Manual*, www.php.net, 1999.
10. Reichard, Kevin, *The Linux Internet Server*, Mis Pr, 1997.
11. Tare, Ramkrishna S, *Unix Utilities*, McGraw-Hill, 1988.
12. *The PHP Company*, www.zend.com, 2000.
13. *The Archive Utility for Windows*, http://www.winzip.com, 2001.