Membangun Linux *filesystem* berbasis Steganografi pada *File* MP3

Ahmy Yulrizka
1) Program Studi Teknik Informatika ITB, Bandung 13504115, email: if14115@students.if.itb.ac.id

Abstract – *Makalah ini membahas ide untuk membuat linux filesystem yang berdiri diatas file MP3. Hal ini bertujuan untuk mempermudah penyembunyian suatu file kedalam file dengan format MP3. Dipilih format file MP3 karena file ini adalah file yang sangat umum digunakan untuk meyimpan file audio karena format ini menyimpan ukuran file yang sangat kecil*

Filesystem dalam linux dikenal dengan nama Linux Virtual Filesystem (VFS). Kernel linux sebenarnya yang melakukan operasi – operasi file. Untuk tiap filesystem kernel memiliki tabel operasi yang berisi koleksi handler untuk tiap operasi file. Dengan metode untuk membuat suatu filesystem kita hanya perlu membuat fungsi – fungsi filesystem dan mendaftarkan fungsi tersebut ke kernel. Mendaftarkan fungsi ke kernel harus dalam bentuk modul. Modul ini yang lebih dikenal dengan modul device driver. Modul inilah yang sebenarnya di daftarkan pada kernel.

Makalah ini memusatkan perhatian pada ide membuat suatu kernel module yang berisi kumpulan fungsi operasi filesystem, bukan pada bagaimana menyembunyikan data pada file MP3. Telah banyak metode untuk menyimpan data diataf file MP3. Makalah ini berasumsi bahwa fungsi tersebut telah terimplementasi. Filesystem ini dibuat dengan bahasa pemrograman C.

Kata Kunci*: MP3, steganografi, filesystem, VFS, kenel, device driver*

1. PENDAHULUAN

Steganografi adalah suatu ilmu yang digunakan untuk menyembunyikan atau menyamarkan informasi atau data diatas pesan atau data yang lain. Tujuan steganografi adalah untuk menghilangkan kecurigaan terhadap suatu pesan.

Medium untuk menyembunyikan data sering disebut dengan *cover image. Cover image* tidak terbatas pada gambar saja, melainkan bisa berbentuk *file* teks, file video dan audio. Dalam makalah ini digunakan media file audio untuk menyembunyikan data.

Terdapat banyak sekali format file audio yang telah umum di pakai di dunia komputer. Diantaranya adalah MP3, WAV, WMV, OOG dll. Dari banyak format file tersebut, Format file MP3 merupakan format file audio yang paling banyak digunakan. Format file ini dipilih karena ukuran hasil kompresi yang sangat tinggi. Kualitas suara yang dihasilkan oleh format file MP3 setara dengan suara yang dihasilkan oleh CD.

Filesystem dalam dunia komputer merupakan suatu metode yang digunakan untuk menyimpan dan mengorganisasikan file dan data yang disimpan didalamnya sehingga user akan mudah untuk mencari dan mengaksesnya. Filesystem dapat menggunakan media penyimpanan data atau yang dikenal dengan *storage device* seperti hardisk, CD-ROM, flashdisk dan melibatkan fungsi – fungsi untuk mengorganisasikan lokasi fisik file pada media tersebut. Secara formal filesystem merupakan suatu tipe data abstrak yang diimplementasi untuk penyimpanan, organisasi, manipulasi, navigasi, akses dan pengambilan suatu data.

Linux adalah sistem operasi yang berbasis file. Semua data maupun perangkat keras diabstraksikan sebagai suatu file . Hal tersebut berari dengan menuliskan suatu data ke file, berarti kita bisa menuliskan data ke suatu device tertentu. File device tersebut merupakan file spesial yang biasanya berada pada direktori */dev*. Filesystem dalam linux diimplementasi dengan membuat suatu module pada kernel. Modul tersebut berisi kumpulan fungsi – fungsi yang akan di eksekusi oleh kernel. Sebagai contoh apabila kita ingin menulis suatu ke filesystem X maka kernel

akan mencari fungsi tulis pada modul kernel X.

Ide dari makalah ini sebenarnya mengimplementasi fungsi – fungsi modul kernel yang menuliskan data keatas file MP3 yang telah disimpan pada filesystem lain. Untuk kemudahan penulis menamakan filesystem yang akan dibuat dengan nama **MyP3FS**. Filesystem ini sebenarnya berdiri diatas filesystem yang telah ada. Misalkan filesystem ini berdi diatas filesystem FAT32. MyP3FS merupakan representasi abstrak yang menuliskan datanya pada file – file MP3 yang berada pada filesystem FAT32. Sebagai contoh apabila kita membuat suatu file pada MyP3FS, file tersebut akan disembunyikan pada file mp3 pada filesystem FAT32. Bagi user filesystem MyP3FS hanya dipandang sebagai filesystem biasa. User dapat melakukan berbagai operasi file sebagaimana filesystem biasa.

2. STEGANOGRAFI PADA FILE MP3

Sebelum memulai membahas bagaimana impelmentasi filesystem ini, akan dijelaskan terlebih dahulu bagaimana proses penyimpanan susatu file. Terdapat beberapa metode untuk menyimpan atau menyembunyikan data diatas format file MP3 [1]. diantaranya adalah :

1. Penggantian *Least Significant Byte Pada metode ini, LSB* input pada setiap sampling digantikan dengan data yang akan di kodeka. Dengan merubah bit yang paling kecil maka perubahan kualitas suara akan berubah pula. Namun karena pendengaran manusia tidak sensitif terhadap perubahan suara yang tidak signifikan, makan manusia yang mendengarkanya tidak akan merasakan banyak perubahan.
2. Penyebaran Spektrum.
   Dengan metode penyebaran spektrum pesan dikodekan dan disebar ke setiap spektrum frekuensi yang memungkinkan.
3. Teknik Echo
   Teknik ini menyamarkan pesan kedalam sinyal yang membentuk *echo*. Kemudian pesan disembunyikan dengan menvariasikan tiga parameter dalam echo : besar amplitudo awal, tingkat penuruan atenuasi dan offset

Pada prinsipnya, ketiga metode diatas bekerja dengan cara mengeksploitasi kelemahan pendengaran manusia dalam membedakan suara dengan perbedaan frekuensi yang sedikit.

Makalah ini lebih menitik beratkan bagaimana membangun suatu filesystem baru dengan memanfaatkan metode – metode peyembunyian data diatas. Untuk membuat suatu implementasi fungsi penyembunyian data pada file MP3 di kernel linux tidak akan dibahas.

3. IMPLEMENTASI FILESYSTEM

Linux adalah sistem operasi yang berbasiskan file. Semua hal dalam linux dalam bentuk file. Terdapat dua jenis file dalam sistem operasi linux, *Character file* dan *Block file.* Perbedaanya adalah character file menyimpan data dalam satuan character atau 1 byte. Sedangkan Block file menuliskan data dalam satuan block. Semua device dalam linux direpresentasikan dengan file. Baik *CDROM, Hardisk, Paralelport, Network Card* dll direpresentasikan dengan file. File – file tersebut disebut dengan special file dan terletak pada direktori */dev/*

Linux Virtual Filesystem (VFS) merupakan abstraksi yang digunakan kernel untuk melakukan organisasi filesystem. Untuk setiap penambahan, pengubahan, pengaksesan file, semuanya ditangani oleh kernel. Kernel melakukan operasi spesifik file dengan cara memanggil suatu fungsi modul filesystem yang bersangkutan. Oleh karena itu tiap filesystem yang berbeda-beda hanya mendaftarkan fungsi *(handler)* spesifik untuk filesystem-nya saja, kernel yang akan mengeksekusi fungsi modul yang sesuai untuk filesystem tersebut.

Pada saat ini filesystem ini diasumsikan bahwa tempat menyimpan data merupakan satu file MP3 yang berukuran sangat besar. Hal ini bertujuan untuk menghindari kerumitan penyimpanan data pada lebih dari satu file MP3.

3.1 Struktur data

Dalam menyimpan suatu informasi dalam filesystem, kernel linux memiliki beberapa struktur data yaitu[2][3] :

1. **File System Type (*struct_file_system_type*)**

Definisinya dapat dilihat pada *include/linux/fs.h*. Struktur data ini menyimpan informasi filesystem pada kernel. Informasi yang disimpan adalah nama filesystem dan fungsi handle *get_sb* dan *kill_sb*. *get_sb* panggil pada saat filesystem di mount dan *kill_sb* dipanggil pada saat filesystem di umount. Melakukan mount pada filesystem berarti melekatkan filesystem kedalam strutur filesystem dari sistem operasi atau yang dikenal dengan root "/". Artinya apabila kita melakukan mount terhadap filesystem maka kita bisa mengakses filesystem tersebut disuatu tempat dibawah struktur root. Misalkan memount */dev/cdrom0* (cd rom drive) ke direktori */media/cdrom* maka file pada cdrom tersebut dapat dibaca pada direktori /media/cdrom. Umount adalah proses untuk melepaskan filesystem dari root.

2. **Super Block (*struct super_block*)**
   Struktur ini menyimpan informasi filesystem yang sedang di mount. Didalamnya terdapat tabel operasi (s_ops) dan root dentry(s_root). Pada saat filesystem di mount. Fungsi get_sb akan mencari objek superblock ini.
3. **Super Block Operations (*struct super_operation*)**
   sturktur data ini berisi fungsi untuk memanipulasi inode.
4. **Inode (struct *inode)***
   objek inode merupakan representasi *low level* kernel terhadap suatu file. Pada inode terdapat 2 tabel operasi. *i_op* dan i_fop. Masing masing menyimpan fungsi handler untuk operasi inode dan file. Fungsi ini dipanggil oleh kernel pada saat kernel berusaha untuk me-*resolve* suatu path.
5. **Inode Operations (*struct inode_operation)***
   struktur ini merupakan tabel operasi yang tiap elemenya menunjuk *pointer* pada fungsi untuk menangani beberapa operasi seperti membuat direktori (*mkdir*), mencari inode(*lookup)* dll.
6. **Address Space Operations** (struct address_space_operation)
   Merupakan tabel operasi untuk melakukan organisasi address space
7. **Dentry (struct dentry)**
   Kernel menggunakan dentries untuk merepresetasikan struktur filesystem. Dentry menunjuk pada objek inode. Dentry yang menyimpan informasi struktur file.
8. **File (struct file)**
   File object digunakan untuk menyimpan informasi proses terhadap suatu file yang sedang digunakan.

3. Impelementasi Fungsi

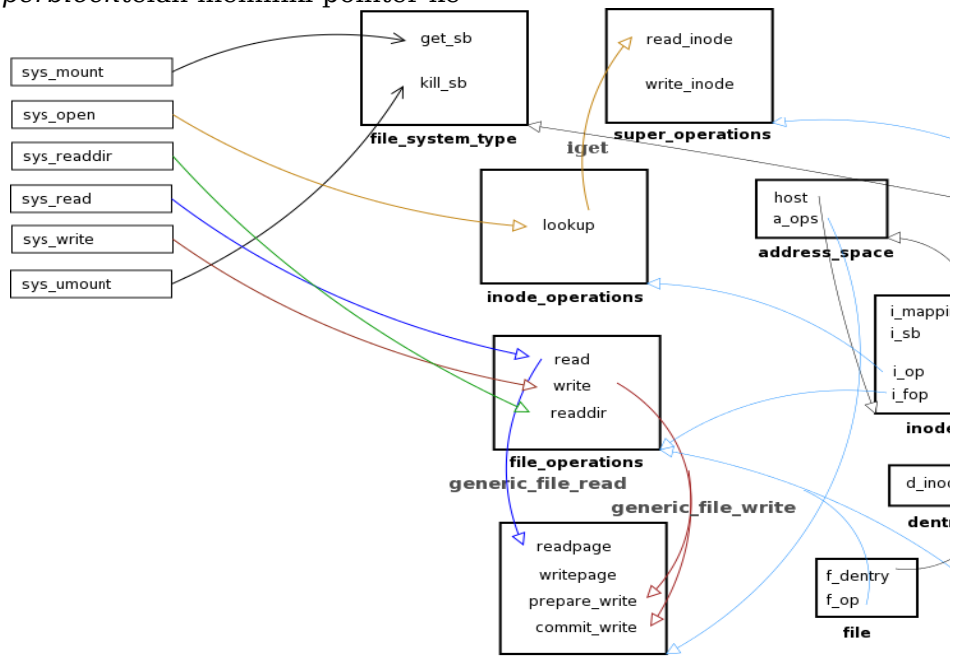
Fungsi yang akan diimplementasi dapat dilihat pada gambar 1. Fungsi tersebut diimplementasi sbb :

1. **init module**
   fungsi ini akan dipanggil pada saat modul melakukan inisialisasi. Yang dilakukan pada modul ini adalah melakukan *registrasi* terhadap filesystem ini. Dilakukan dengan membuat struktur data sbb :

```
static struct file_system_type
myp3fs = {
  name : "myp3fs",
  get_sb: myp3fs_get_sb,
  kill_sb: myp3fs_kill_sb,
  owner : THIS_MODULE
};
int init_module(void) {
  int err;
  err =
register_filesystem(&myp3fs);
  return err;
}
```

   fungsi ini akan melakukan pendaftaran filesystem baru pada kernel dengan tipe filesystem myp3fs.
2. **myp3fs_get_sb**
   fungsi ini dipanggil pada saat sistem di mount. Fungsi ini akan membaca *superblock* yang disimpan pada awal file mp3. Dalam fungsi ini *root inode* akan diinisialisasi dan *dentry*nya di set pada *field* s_root.
3. **myp3fs_kill_sb**
   fungsi ini akan dipanggil pada saat filesystem akan di umount
4. **super_operation.read_inode**
   fungsi ini digunakan untuk membaca inode. Pada saat

filesystem telah di mount, maka *superblock*telah memiliki pointer ke dentry yang menyimpan inode.



Gambar 1: Funasi yang diimplementasi dan struktur datanya[2]

5. **super_operations.write_inode**
   fungsi ini akan dipangil apabila perlu untuk menuliskan inode yang telah berubah ke file mp3
6. **inode_operations.lookup**
   fungsi ini akan dipanggil apabila kernel sedang berusaha untuk *meresolve* suatu path. Fungsi ini akan menelusuri *dentry* dari s_root pada *superblock*.
7. **file_operation.readdir**
   Fungsi ini akan dipanggil pada saat kernel membaca isi suatu direktori. Fungsi ini yang sebenarnya akan dipanggil pada saat user mengetik perintah *ls*. Hierarki filesystem akan dibaca berdasarkan *dentry* yang disimpan.
8. **file_operation.read**
   fungsi ini yang akan dipanggil oleh kernel pada saat kernel mendapat *system call read* untuk suatu file. Filesystem ini akan mencari file tersebut berdasarkan path. Data pada file disimpan pada inode. Isi dari inode dapat dipandang sebagai suatu page yang di representasikan dengan *address_space_object*.
9. **address_space_operation.readpage**
   seperti yang dikatakan sebelumnya bahwa isi file atau isi inode direpresentasikan sebagai page. Operasi pada page adalah yang sebenarnya dipanggil pada saat mengakses page. Operasi readpage akan dipanggil apabila kernel mendapat *system call* untuk membaca isi file.
10. **file_operation.write**
    sama seperti pada file_operation.read, fungsi ini yang akan dipanggil saat menulis suatu file.
11. **address_space_operation.writepage**
    fungsi ini akan dipanggil apabila terdapat perubahan pada page. Karena isi inode diakses oleh fungsi dari *address_space_operation* maka fungsi ini akan dipanggil pada perubahan isi suatu file.
12. **address_space_operations.commit_write.**
    Fungsi ini akan dipanggil apabila kernel mendapat *system call* untuk merubah isi suatu file. Fungsi ini akam menuliskan inode pada file MP3 berdasarkan path dari file tersebut.
13. **cleanup_module**
    fungsi ini akan dipanggil pada saat module dihilangkan dari kernel. Fungsi ini akan melakukan proses *unregister* filesystem ini yang di registrasikan pada saat fungsi init

module dipanggil

3. HASIL DAN PEMBAHASAN

Untuk mensimulasikan bagaiman cara filesystem bekerja, maka dibuat ilustrasi sebagai berikut.

Apabila filesystem ini telah berhasil terimplementasi maka pengguna terlebih dahulu harus melakukan kompilasi terhadap modul device driver myp3fs ini. Proese kompilasi akan menghasilkan suatu *kernel object file* yang memiliki ekstensi *.ko* . Untuk mendaftarkan modul tersebut pada kernel, pengguna memasukan perintah

 # insmod myp3fs.ko

perintah ini akan memanggil fungsi init_module pada modul myp3fs dan melakukan registrasi filesystem pada kernel.

Setelah melakukan registrasi modul pada kernel maka filesystem ini sudah bisa digunakan. Seperti yang telah dijelaskan sebelumnya bahwa untuk mengurangi kerumintan filesystem ini hanya bisa berjalan diatas 1 file MP3 saja. Semua data akan disembunyikan pada file tersebut.

Untuk memulai melakukan mount myp3fs filesystem diatas satu file mp3 digunakan perintah

mount -t myp3fs [path file mp3] [mount point]

contoh

mount -t myp3fs /media/iPod/test.mp3 /mnt/myp3fs

perintah ini akan melakukan mounting sebuah device baru yang memiliki filesystem MYP3FS dan akan di tempelkan ke bawah direktori /mnt/myp3fs. Perintah ini akan membaca superblock yang disimpan pada awal file mp3 tersebut. Karena data yang disembunyikan merupakan suatu blok. Fungsi **get_sb** akan mengkonstruksi superblock yang disembunyikan pada file mp3 tsb.

Apabila pengguna berpindah ke direktori */mnt/myp3fs* maka ia sebenarnya meakses filesystem diatas file mp3 /media/iPod/test.mp3. Module akan melakukan navigasi direktori dengan membaca dentry rootnya

Misalkan ia membuat suatu file bernama “sembunyi.txt” pada direktori **/mnt/myp3fs** maka file tersebut akan secara langsung disembunyikan pada file **/media/iPod/test.mp3**.

Pengguna teresbut dapat melihat isi root direktori dengan memasukan perintah

ls /mnt/myp3fs

perintah tersebut akan menghasilkan sbb :

. .. sembunyi.txt

sebenarnya perintah tersebut melakukan **system_call sys_readdir** yang akan memanggil fungsi file_operations_readdir

Misalkan pengguna ingin membaca isi dari file sembunyi.txt, ia akan memasukan perintah

cat /mnt/myp3fs/sembunyi.txt

modul myp3fs akan mencari indoe dari files sembunyi.txt dengan fungsi *inode_operation.lookup* setelah menemukan inode dari file tersebut kernel akan memanggil fungsi *super_operation.read_inode* dan membaca file tersebut memanfaatkan fungsi *address_operations.readpage*

begitu juga pada saat user ingin menuliskan isi file tersebut. Ia dapat memasukan perintah

echo “ini pesan rahasia” > /mnt/myp3fs/sembunyi.txt

maka modul filesystem myp3fs akan menyadi inode dengan fungsi inode_operation.lookup dan setelah menemukan inode tersebut kernel akan memanggil fungsi super_operation.write_inode nan menuliskan page pada inode tersebut dengan memanfaatkan fungsi *address_operations.writepage*

apabila pengguna ingin menghapus file pesan rahasia tersebut, maka ia akan memaskan perintah

rm /mnt/myp3fs/sembunyi.txt

perintah tersebut akan mencari inode untuk file tersebut dan menghapus inode tersebut dari struktur filesystem. Dengan menghapus inode pada filesystem ini. Pada saat file dihapus. Filesystem akan mengkonstruksikan blok inode yang disimpan pada file mp3 tersebut.

Untuk melepaskan filesystem dari root filesystem maka pengguna cukup memasukan perintah

umount [mount point]

contoh

umount /mnt/myp3fs

modul akan memanggil fungsi myp3fs.kill_sb dan menuliskan perubahan yang terjadi pada filesystem dengan cara menyembunyikan data-data tersebut pada file MP3.

4. KESIMPULAN

Filesystem ini berfungsi untuk melakukan penyembunyian data pada suatu file MP3 dengan mudah. Dengan filesystem ini user tidak akan mengalami kesulitan utuk melakukan organisasi file. User memandang filesystem ini semagaimana filesystem lainya. Program lain juga dengan mudah menuliskan, mengakses atau menghapus suatu file. Filesystem ini akan menyembunyikan kerumintan proses penyembunyia data.

MYP3FS filesystem ini memiliki beberapa kelemahan. Diantaranya adalah hanya mendukung suatu file dan mengasumsikan bahwa file yang MP3 sebagai *cover image* sangat besar sehingga cukup untuk menampung data. Pada kenyataanya file mp3 rata – rata berukuran 3-5 MB. Jika metode yang digunakan untuk menyembunyikan data adalah dengan menggunakan 1 LSB. Maka data yang disimpan adalah sebesar 1/8 dari 5 MB atau sebesar 625 Kb. Kapasitas tersebut digunakan untuk menyimpan struktur hierarki dan struktur data juga untuk data. Maka dalam kondisi nyata filesystem ini kurang bermanfaat.

Apabila filesystem ini telah bisa berdiri diatas beberapa file MP3 maka kapasistanya tidak lagi bergantung pada satu buah file saja, melainkan 1/8 dari jumlah file mp3 yang ada. Untuk memenuhi hal tersebut, perlu dipikirkan bagaimana blok data pada filesystem disimpan secara kontinu diatas beberapa file mp3.

DAFTAR REFERENSI

1[1] Simon Batara, “Studi Steganografi dalam File MP3”, Program Studi Teknik Informatika, Institut Teknologi Bandung, 2006.

2[2] Kiran, Ravi UVS “Writing a Simple File System”, http://www.geocities.com/ravikiran_uvs/articles/rkfs.html

3[3] Jonathan Corbet, Alessandro Rubini, and Greg Kroah-Hartman , “Linux Device Drivers”, O'Reilly. 2007

yang digunakan adalah Times New Roman.