Penyisipan Identitas Pemilik Aplikasi Ke Dalam Executable File Pada Aplikasi STEAM

Benedikus Holyson Tjuatja – 13510101
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13510101@std.stei.itb.ac.id

***Abstract*—Pada makalah ini akan dibahas bagaimana salah satu cara untuk mencegah penyebaran aplikasi secara illegal dengan cara membajakan aplikasi yang didistribusikan oleh aplikasi STEAM. Cara yang diusulkan untuk mencapai tujuan ini adalah dengan cara menyimpan identitas pengguna pada aplikasi STEAM ke dalam *file executable* aplikasi yang menjadi target penyebaran. Identitas yang disisipkan ini nantinya akan digunakan sebagai bukti kepemilikan aplikasi. Pada pembuatan makalah ini juga dibuat sebuah program sederhana dengan menggunakan bahasa JAVA.**

***Kata Kunci: steganografi, executable, STEAM*.**

I. PENDAHULUAN

Sejak munculnya jaringan internet, perkembangan teknologi dan dunia digital meningkat dengan sangat cepat. Banyak hal yang dulunya sangat terbatas kini dapat menyebar secara cepat dan luas. Perpindahan data dan informasi kini tidak mengenal batas tempat dan waktu. Semua hal tersebut dapat diakses dengan menggunakan jaringan internet. Dengan semakin cepatnya pertukaran data dan informasi yang berada pada jaringan internet, hal-hal yang akhirnya bergerak dibidang digital semakin bertambah banyak. Penambahan ini dapat berupa penambahan layanan jasa ataupun barang dengan menggunakan jaringan digital sebagai katalisnya maupun pemberian layanan yang hanya berada pada dunia digital.

Perkembangan dunia digital ini juga membuat proses transaksi jual beli melalui via digital menjadi sangat digemari oleh masyarakat. Toko-toko online yang menjual barang-barang digital maupun nyata juga mulai banyak bermunculan. Dengan menggunakan suatu akun virtual sebagai identitas pembeli dan kartu kredit sebagai mata uang pembayaran secara digital, transaksi jual beli memberikan kemudahan bagi para pembeli dan keuntungan bagi para penjual. Tak heran jika kini orang mulai berlomba-lomba membuat suatu aplikasi yang dapat menjadi distributor atau perantara yang mempertemukan calon penjual dan calon pembeli.

Salah satu aplikasi yang memberikan layanan sebagai distributor secara digital tersebut adalah STEAM. Aplikasi yang dibuat dan dikembangkan oleh Valve Corporation ini selain digunakan sebagai distributor aplikasi digital juga memberikan layanan pengaturan hak kepemilikan benda digital. Aplikasi digital yang didistribusikan oleh aplikasi STEAM ini adalah aplikasi perangkat lunak secara umum, dan sebagian besar dari perangkat lunak tersebut merupakan aplikasi permainan komputer.

Sebagai distributor dan pengelolah kepemilikan benda digital, STEAM menggunakan sistem akun virtual sebagai penanda kepemilikan seseorang terhadap suatu aplikasi perangkat lunak. Sistem ini menghubungkan akun virtual suatu pengguna terhadap aplikasi-aplikasi perangkat lunak yang dimiliki oleh pengguna tersebut dan kemudian menyimpan data tersebut ke dalam server. Ketika akun pengguna dinyatakan memiliki suatu aplikasi perangkat lunak, maka pengguna tersebut dapat melakukan instalasi perangkat lunak tersebut ke dalam komputer miliknya.

Meskipun demikian, aplikasi yang didistribusikan dengan menggunakan STEAM ini tidak sepenuhnya aman terhadap penyebaran luasan secara illegal. Pihak yang tidak bertanggung jawab dapat mengubah suatu data dari perangkat lunak ini kemudian menyebar luaskannya melalui jaringan internet secara ilegal. Salah satu cara pembajakan yang dilakukan untuk menyebar luaskan aplikasi tersebut adalah dengan merubah suatu data pada aplikasi sehingga sistem hubungan kepemilikan yang terdapat pada server STEAM tidak lagi digunakan.

Pada makalah ini akan dibahas bagaimana cara mengatasi masalah ini dengan menggunakan metode steganografi. Metode steganografi yang digunakan disini adalah dengan menyisipkan kode identitas ataupun akun seseorang kedalam file executable aplikasi yang dimilikinya. Dengan cara ini diharapkan penyebaran suatu aplikasi secara illegal dapat berkurang .

II. STUDI LITERATUR

Steganografi atau seni penyembunyian pesan ini telah berkembang sejak lama dan pertama kali tercatat pada tahun 1499. Pada steganografi modern, metode ini

biasanya digunakan untuk menyembunyikan suatu data atau pesan digital kedalam data digital lainnya seperti filde dokumen, gambar, audio, video, program, dan lainnya. Penyembunyian pesan ini sendiri biasanya digunakan untuk membubuhkan tanda tangan digital kedalam suatu data sebagai tanda kepemilikan.

Pada steganografi terdapat beberapa hal yang menjadi dasar dalam menyisipkan pesan. Yang pertama adalah data yang ingin disimpan, pada makalah ini data yang disimpan berupa identitas pemilik. Kedua adalah kunci steganografi yang akan digunakan sebagai petunjuk untuk mengambil dan menyimpan data, pada makalah ini kunci berupa pola penyimpanan algoritma dan identitas pengguna. Ketiga adalah file target sebagai tempat penyimpanan pesan atau tempat melakukan steganografi, implementasi kali memanfaatkan file executable sebagai tempat penyimpanan.

Executable file (.exe) merupakan suatu ekstansi umum yang digunakan sebagai penanda utma dalam melakukan eksekusi suatu program. File executable terdiri dari kumpulan suatu binary yang mnyusun perintah-perintah untuk menjalankan suatu program. Selain hal tersebut data binary ini sendiri mengandung informasi lain seperti nilai jenis aplikasi, grafik bitmap dan ikon yang digunakan sebagai untuk membentuk tampilan antarmuka kepada pengguna.

III. RANCANGAN

Ide utama dalam pada makalah ini adalah bagaimana cara untuk memanfaatkan file executable yang merupakan data utama dalam melakukan suatu program sebagai salah cara untuk mencegah terjadinya penyebaran aplikasi secara illegal. Seperti yang telah dibahas sebelumnya, file executable ini merupakan bagian yang penting dalam menjalankan suatu program aplikasi dan perubahan nilai binary pada file ini dapat menyebabkan aplikasi tersebut tidak dapat berjalan. Meskipun demikian pada file executable ini masih terdapat bagian-bagian yang dapat digunakan sebagai tempat untuk melakukan penyisipan suatu data. Data yang akan dimasukan ini nantinya akan menjadi suatu penanda apakah suatu pengguna dapat menjalan aplikasi tersebut.

Terdapat beberapa cara yang akan dilakukan dalam melakukan penyisipan ini. Cara yang pertama adalah dengan menambahkan nilai identitas dari pengguna pada bagian akhir dari file executable program dalam bentuk binary. Hal ini dilakukan dengan asumsi file executable tidak akan menjadi rusak setelah selesai dienkrispsi sehingga file executable dapat berjalan dengan baik.

Cara yang kedua adalah dengan melakukan penyisipan data identitas dari pengguna kedalam bagian-bagian kosong yang terdapat pada file executable ini. Beberapa literature mengatakan bahwa pada file executable sendiri terdapat bagian-bagian yang kosong sehingga dapat disisipkan dengan suatu data. Penyisipan ini tidak akan mengganggu kinerja dari file executable karena daerah penyisipan yang digunakan adalah daerah yang kosong.

Cara yang terakhir adalah dengan mengubah beberapa nilai binary yang tedapat pada file executable ini dengan nilai binary dari identitas pengguna. Karena cara ini merubah nilai binary pada suatu file executable, maka penyisipan ini dapat dilakukan pada awal, akhir, tengah file ataupun secara random dengan menyebarkan nilai binary dari pengguna yang sudah dibagi-bagi.

Untuk memperkuat data enkripsi, sebelum disisipkan kedalam file executable data binary indentitas pengguna dapat dienkripsi terlebih dahulu. Cara enkripsi yang digunakan dapat bermacam-macam seperti vigenere, CBC, stream chipper, dll. Namun karena makalah ini lebih difokuskan kepada penyisipan file, sehingga hal ini tidak diimplementasikan.

Pada penyisipan ini juga dibutuhkan penentuan penanda akhir dari suatu file (*End of File*). *End of File* ini dapat ditemukan dengan cara menelusuri nilai binary pada file tersebut. Biasanya nilai *End of File* dari suatu data adalah nilai binary dari karakter "\0" atau nilai binary dari karakter "-1". Pada beberapa bahasa, nilai dari *End of File* ini juga direpresentasikan sebagai karakter NUL.

Pengujian pada makalah ini dilakukan dengan membuat suatu program sederhana yang dapat diibaratkan sebagai pengganti aplikasi STEAM yang menerapkan sistem steganografi. Program ini dibuat dengan menggunakan kakas Netbeans 7.3 yang menggunakan bahasa pemrograman JAVA. Pada program ini teknik untuk mengolah nilai binary dari file executable akan menggunakan FileInputStream sebagai pembacaan file dan FileOutputStream sebagai penulisan file.

IV. IMPLEMENTASI

Seperti yang telah dibahas pada tahap rancangan, implementasi dari makalah ini akan menggunakan bahasa pemrograman Java dengan melakukan tiga buah teknik penyisipan pesan. Proses pengolahan ini juga terdiri dari dua tahap, yiatu tahap enkripsi file aplikasi dan tahap dekripsi file aplikasi. Karena program ini merupakan suatu contoh apliasi STEAM sederhana, maka nilai keberhasilan dari proses dekripsi juga dilihat dari dapat dijalankannya file executable tersebut dengan menggunakan program ini.

A. Menambahkan

Penambahan identitas pemilik ini dilakukan pada bagian akhir dari suatu file atau setelah pembacaan nilai *End of File*. Karena penyisipan dilakukan pada akhir file executable, maka cara ini merupakan cara yang paling sederhana dibandingkan dengan dua cara lainnya. Proses enkripsi pada cara ini, nilai binary dari file executable yang akan dienkripsi pertama kali akan disalin ulang kedalam file executable yang baru. Penyalinan file ini akan terus dilakukan hingga menemukan nilai *End of File*.

```
// Copy File asli
tempRead = file.read();
while(tempRead != -1){
        outFile.write(tempRead);
        tempRead = file.read();
}
outFile.write(tempRead);
```

Pada file executable yang baru kemudian akan ditambahkan nilai binary dari identitas pengguna. Karena pemasukan dilakukan secara binary makan identitas dari pengguna ini akan dimasukan secara per karakter dengan nilai binary dari karakter tersebut diambli berdasarkan nilai ASCII. Setelah semua karakter data berhasil dimasukan maka pada bagian akhir dari program akan ditambahkan lagi karakter *End of File* yang palsu.

```
// Tambahkan string id di ujung
for (int i = 0; i < user.length(); i++){
    outFile.write(user.charAt(i));
}
outFile.write(-1);
```

Dengan cara ini file executable yang dihasilkan akan mengandung nilai dari identitas dari pengguna pada bagian akhir setelah nilai *End of File* yang asli.

Proses dekripsi dari cara ini juga dapat dikatakan sederhana. File executable pertama kali akan dibaca hingga menemukan nilai dari *End of File* yang asli

.

```
// Cara 1
// Baca hingga ketemu EoF
tempRead = file.read();
while(tempRead != -1){
        tempRead = file.read();
}
```

Kemudian untuk selanjutnya akan dilakukan pengecekan apakah nilai binary yang terdapat pada file executable tersebut akan sama dengan nilai binary dari identitas pengguna.

```
// Bandingkan apakah ID user sama
boolean tempbol = true;
tempRead = file.read();
int tempi = 0;
while(tempRead != -1 && tempbol){
    if (tempRead != user.charAt(tempi)){
        tempbol = false;
    }
    tempi++;
    tempRead = file.read();
}
file.close();
```

Setelah dilakukan pengecekan makan akan lihat hasil pengecekan tersebut, jika ternyata identitas pengguna sesuai dengan data identitas yang terdapat pada file executable tersebut maka program tersebut akan dijalankan.

```
// Jika ID user sama, jalankan prosesnya
if (tempbol){
    Process p = Runtime.getRuntime().exec
        ("cmd /c start " +
         newf.getAbsolutePath());
}
```

B. Menyisipkan

Penyisipan identitas ini dilakukan dengan mencari bagian-bagian binary dari file executable ini yang kosong kemudian merubah nilai binary tersebut menjadi nilai binary identitas pengguna. Berbeda dengan cara yang sebelumnya, file executable yang baru tidak menyalin nilai binary dari file executable secara keseluruhan. Program akan membaca nilai dari binary file executable awal secara bertahap, jika nilai terbaca ternyata kosong maka file executable yang baru akan diisi dengan nilai binary dari identitas pengguna, jika tidak maka file executable yang baru akan menyalin nilai binary dari file executable yang lama. Selain pengubahan nilai binary kosong menjadi nilai binary identitas pengguna, juga akan dicatat posisi-posisi dari binary yang disisipkan. Asumsi bahwa jumlah binary identitas pengguna yang akan disisipkan akan lebih sedikit dibandingkan dengan jumlah nilai binary kosong yang terdapat pada file executable.

```
// Cara 2 - Penyelipan di tempat kosong tengah
// Copy File asli
tempRead = file.read();
int tempint = 0;
int tempi = 0;
while(tempRead != -1){
    if (tempRead == 0
            && tempint < user.length()){
        filePosition.add(tempi);
        outFile.write(user.charAt(tempint));
        tempint++;
    }else{
        outFile.write(tempRead);
    }
    tempRead = file.read();
    tempi++;
}
outFile.write(tempRead);
// Tutup filenya
file.close();
```

Proses dekripsi cara ini sedikit lebih rumit dibandingkan proses enkripsinya. Penyisipan yang dilakukan pada bagian tengah file executable diasumsikan dapat menimbulkan perubahan data sehingga file executable yang dienkripsi tidak dapat berjalan dengan semestinya. Hal ini menyebabkan diperlukan pembuatan

file menjadi file executable yang sebelum dienkripsi dan kemudian akan menggantikan file executable yang telah dienkripsi. Seperti pada proses enkripsi, pada proses dekripsi nilai binary dari file executable akan dibaca satu persatu, ketika posisi binary binary yang dibaca telah sesuai dengan posisi binary yang telah dicatat sebelumnya, maka nilai dari binary tersebut akan diganti dengan nilai binary kosong dan dibandingkan dengan nilai binary identitas pemilik. Hal ini dilakukan terus menerus hingga semua posisi binary yang tercatat telah dibandingkan semuanya.

```
tempRead = file.read();
int tempPos = 0;
while(tempRead != -1){
    if (filePosition.size() > 0
            && tempPos == filePosition.get(0)){
        outFile.write(0);
        filePosition.remove(0);
    }else{
        outFile.write(tempRead);
    }
    tempPos++;
    tempRead = file.read();
}
file.close();
outFile.close();
```

Jika ternyata sesuai maka file executable tersebut akan ditukar dengan file executable yang baru kemudian program tersebut akan dijalankan.

```
// Timpa .exe chipper dengan .exe dechipper
outFile = new FileOutputStream(hasilUrl);
file = new FileInputStream(f2);
tempRead = file.read();
while(tempRead != -1){
    outFile.write(tempRead);
    tempRead = file.read();
}
outFile.write(tempRead);
outFile.close();
file.close();
f2.delete();
// Run
Process p = Runtime.getRuntime().exec
        ("cmd /c start "+
        newf.getAbsolutePath());
```

C. Memanipulasi

Cara yang terkakhir ini memiliki tingkat kerumitan yang sama dengan cara kedua. Cara yang ketiga ini dilakukan dengan cara merubah binary file executable menjadi nilai binary dari identitas pengguna, Posisi binary yang akan dirubah akan dilakukan berurutan pada bagian awal, tengah, akhir file ataupun dengan menyebarkan nilai binary identitas pengguna secara random. Jika berurutan maka hilai yang perlu disimpan hanya posisi awal dari binary yang dirubah, namun jika secara random akan dicatat semua posisi binary yang telah disebar. Nilai binary identitas pengguna ini tidak akan digunakan pada proses dekripsi, karena cukup menggunakan posisi dan nilai binary sebelumnya, namun nilai binary ini dapat digunakan sebagai salah satu alternatif untuk melakukan perandoman posisi binary yang akan dirubah. Pada pembuatan kali ini, nilai binary yang diganti hanya nilai awal binary pada file executable menjadi nilai binary identitas pengguna secara berurutan. Pada proses enkripsi akan dilakukan pengubahan nilai binary executable menjadi nilai binary identitas pengguna, kemudian dilakukan pencatatan nilai binary yang dirubah agar nantinya dapat di dekripsi kembali. Proses ini dilakukan terus hingga semua nilai binary identitas pengguna berhasil disispkan kedalam file executable.

```
// Cara 3 - Mengubah menjadi data user
// Copy File asli
tempRead = file.read();
int tempint = 0;
while(tempRead != -1){
    if (tempint < user.length()){
        filePosition.add(tempRead);
        outFile.write(user.charAt(tempint));
        tempint++;
    }else{
        outFile.write(tempRead);
    }
    tempRead = file.read();
}
outFile.write(tempRead);
// Tutup filenya
file.close();
```

Pada proses dekripsi juga hampir menyerupai cara kedua, dilakukan penelusuran binary executable hingga sesuai dengan posisi yang dicatat. Nilai tersebut kemudian akan ditukar dengan nilai binary executable awal yang telah dicatat. Untuk berikutnya akan dilakukan penggantian nilai executable yang lama dengan file executable yang baru.

```
tempRead = file.read();
while(tempRead != -1){
    if (filePosition.size() > 0){
        outFile.write(filePosition.get(0));
        filePosition.remove(0);
    }else{
        outFile.write(tempRead);
    }
    tempRead = file.read();
}
file.close();
outFile.close();
```

```
// Timpa .exe chipper dengan .exe dechipper
outFile = new FileOutputStream
        ("D:\\Prototype\\prototypef.exe");
file = new FileInputStream(f2);
tempRead = file.read();
while(tempRead != -1){
    outFile.write(tempRead);
    tempRead = file.read();
}
outFile.write(tempRead);
outFile.close();
file.close();
f2.delete();
// Run
Process p = Runtime.getRuntime().exec
        ("cmd /c start "+
        "D:\\Prototype\\prototypef.exe");
```

V. Analisis Hasil

Dari percobaan yang dilakukan didapatkan hasil yang berbeda-beda. Ketika menggunakan teknik yang pertama yaitu penyisipan pada bagian akhir file, file tersebut masih dapat dijalankan dengan baik. Hal ini dapat dikatakan bahwa penambahan binary pada akhir file tidak akan merusak nilai dari file tersebut. Penyisipan data identitas pengguna dengan menggunakan cara ini hanya dapat digunakan sebagai tanda tangan virtual. Penyisipan tersebut tidak akan memberikan pengaruh apapun, dan akan memberikan maanfaat ketika data yang disisipan tersebut dicari.

Hasil percobaan dengan cara yang kedua memberikan hasil yang sama dengan cara yang pertama. File yang telah terenkripsi masih dapat berjalan dengan baik. Cara ini juga hanya dapat digunakan sebagai tanda tangan virtual saja. Perbedaan dengan cara yang pertama adalah pada besar ukuran file yang dihasilkan dan jumlah data yang dapat disimpan. Hasil ukuran file enkripsi dengan menggunakan cara yang kedua tidak akan berbeda jauh dengan ukuran file sebelum dienkripsi, karena menggunakan jumlah binary yang sama dan hanya mengubah nilai binarynya saja. Namun cara ini memiliki batasan pada jumlah data yang dapat disisipkan. Jika pada cara pertama jumlah data dapat dimasukkan secara bebas dikarenakan menggunakan proses konkatenasi, jumlah data yang dapat disisipkan dengan cara kedua hanya berdasarkan jumlah binary kosong yang terdapat pada file executable tersebut.

Hasil percobaan dengan cara yang ketiga memberikan hasil yang berbeda dengan cara yang pertama dan kedua. File executable dari cara ini tidak dapat digunakan dan seolah-oleh seperti file yang sudah rusak. Ketika dilakukan dekripsi terhadap file tersebut, file executable tersebut kembali berkerja dengan sempurna. Dapat dikatakan cara ketiga ini akan memekasa pengguna untuk menjalankan program tersebut dengan menggunakan program dekripsinya, dalam hal ini STEAM. Penggunaan identitas pengguna dengan cara ini sedikit berbeda dengan cara yang pertama dan kedua dimana identitas user tersebut dibandingkan secara langsung dengan file binarynya. Identitas user pada cara ketiga ini digunakan sebagai penanda posisi data binary yang akan digantikan. Penyebaran aplikasi secara legalpun tidak dapat menggunakan cara ini, meskipun data tersebut diberikan kepada user yang juga memilki program yang sama namun posisi binary yang akan ditukar berbeda sehingga aplikasi tersebut tidak dapat dijalankan.

Untuk menangani kasus penyebaran data secara illegal maka dapat dikatakan penyisipan dengan cara ketiga memberikan hasil yang efektif.

VI. Kesimpulan

Penyisipan identitas pengguna kedalam file executable aplikasi dapat digunakan sebagai “password” atau tanda tangan digital. Sebagai tanda tangan digital, penyisipan ini tidak dapat mencegah pembajakan aplikasi namun dapat diketahui pemiliki dari aplikasi tersebut. Sedangkan sebagai “password”, hak akses aplikasi tersebut akan dipersempit karena hanya bisa dijalankan melalui aplikasi STEAM, namun hal ini dapa membuat proses pembajakan aplikasi menjadi lebih sulit.

References

[1] Wayner, Peter. Disappearing cryptography: information hiding: steganography & watermarking. Armsterdam: MK/Morgan Kaufman Publisher. 2009.

[2] http://brasil.cel.agh.edu.pl/~11ugbogusz/index.php?option=com_content&view=article&id=12&Itemid=17&lang=en. diakses pada tanggal 18 Maret 2014.

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 17 Maret 2014

ttd

Benedikus Holyson Tjuatja
13510101