BAB IV

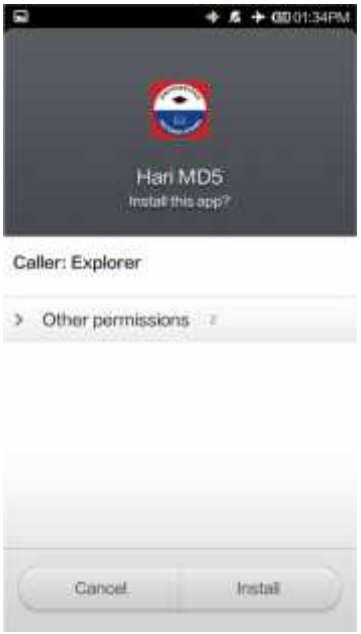
HASIL DAN UJI COBA

IV.1. Jalannya Uji Coba

Aplikasi Android yang telah berhasil dibuat akan memiliki ekstensi *.apk (* merupakan nama file aplikasi). Aplikasi yang penulis rancang agar dapat digunakan di perangkat Android pengguna, haruslah diinstalasi terlebih dahulu. Kita bisa mengirimkan file tersebut dari lingkungan pengembangan Android di komputer ke perangkat Android melalui media *bluetooth* atau media kabel USB. Setelah terkirim, buka aplikasi *File Browser* yang ada pada perangkat Android untuk mencari keberadaan lokasi file instalasi. Aplikasi yang penulis rancang bernama “HariMD5” dan berekstensi .apk.

Gambar IV.1. Lokasi aplikasi yang akan diinstalasi

Jika lokasi beserta file aplikasi tersebut ditemukan, klik pada aplikasi tersebut untuk memulai penginstalasian. Umumnya perangkat Android akan mencoba memverifikasi tindakan pengguna dengan menampilkan *form permission*. Klik pada tombol *Install* untuk menyetujui penginstalan aplikasi.



Gambar IV.2. Form *permission* saat penginstalan aplikasi

Selanjutnya penginstalan akan berlangsung beberapa saat, dan jika sudah penginstalan aplikasi sudah selesai, akan muncul notifikasi bahwa instalasi telah berhasil (*installation complete*). Klik pada tombol Launch untuk dapat segera menggunakan aplikasi *Checksum Hash* MD5.

Gambar IV.3. Penginstalan aplikasi berhasil

Sebelum memulai pemasangan aplikasi yang dirancang ke perangkat Android, terlebih dahulu pengguna harus mengubah pengaturan *security* dari perangkat Android dalam hal pengizinan pemasangan aplikasi dari sumber luar (*unknown sources*) agar aplikasi tersebut bisa di pasang dan berjalan dengan baik di perangkat Android tersebut. Perlu diketahui bahwa ada kemungkinan letak pengaturan ini berbeda-beda di beberapa perangkat Android, tergantung dari UI (*user interface*) bawaan pabrikan perangkat Android masing-masing. Dalam pengujian ini, penulis menggunakan perangkat Android bertipe Xiaomi Redmi 1S yang menggunakan UI MIUI rancangan pabrikan Xiaomi.

Pertama, buka menu *Settings*. Di bagian *Personal*, klik pada bagian *Security*.

Gambar IV.4. Pengaturan *Security* pada bagian *Personal*

Setelah tampilan pengaturan *Security* muncul, kita perlu mencari pengaturan bernama *Unknown Sources*. Pengaturan ini berada di bagian *Device Administrator* pada perangkat Android penulis. Bisa terlihat bahwa *toggle* dari pengaturan ini masih berada dalam keadaan *off*. Ini artinya saat ini perangkat Android tersebut menolak pemasangan aplikasi dari sumber luar (*unknown sources*) untuk mencegah terinstalasinya aplikasi Android yang tidak dikenal.

Gambar IV.5. Pengaturan *Unknown sources* pada bagian *Device Administrators*

Adakalanya saat kita mencoba untuk melakukan instalasi aplikasi Android, perangkat Android tersebut memberikan peringatan dan menolak untuk menginstal aplikasi. Hal ini dikarenakan ada sebuah fitur keamanan di dalam perangkat Android bernama "*Unknown sources*" yang belum aktif. Fitur ini akan memperingatkan pengguna agar berhati-hati atas kerusakan yang mungkin saja bisa diakibatkan dari penginstalasian aplikasi Android yang tak dikenal. Untuk keperluan uji coba, kita perlu menyalakan fitur *Unknown sources* agar aplikasi Android yang kita rancang bisa di pasang dan berjalan dengan baik di perangkat Android tersebut.

Klik pada bagian *Unknown sources* untuk mengubah *toggle* fitur ini dari posisi *off* menjadi posisi *on*. Pada umumnya saat kita berusaha merubah pengaturan ini, sebuah *pop-up* akan muncul, yang berisikan peringatan untuk berhati-hati atas kerusakan yang mungkin saja bisa diakibatkan dari penginstalasian aplikasi Android yang tak dikenal. Klik tombol *OK* untuk menyetujui peringatan ini.

Gambar IV.6. Peringatan saat mengubah pengaturan *Unknown sources*

Setelah berhasil, maka pengaturan *Unknown sources* sudah aktif dan saat ini aplikasi Android yang kita rancang bisa di pasang dan berjalan dengan baik di perangkat Android tersebut.

Gambar IV.7. Pengaturan *Unknown sources* sudah aktif

IV.2. Tampilan Layar

Pada saat pertama kali dijalankan, akan muncul form utama. Pada form ini terdapat tiga tombol yang akan membuka aktivitas form lain berdasarkan fungsinya masing-masing. Gambar IV.5 berikut adalah form utama pada aplikasi:

Gambar IV.8. *Form* Utama Aplikasi *Checksum Hash* MD5

Apabila tombol “Bangkitkan *Hash* MD5 Dari Teks” di klik, maka akan muncul *form* “*Checksum Hash* MD5 Dari Teks”, yang berfungsi untuk melakukan pembangkitan *checksum hash* MD5 dari teks yang di-*input* oleh pengguna. Gambar IV.6 berikut adalah *form Checksum Hash* MD5 Dari Teks:

Gambar IV.9. Form Bangkitkan Hash MD5 Dari Teks

Apabila tombol “Bangkitkan *Hash* MD5 Dari *File*” di klik, maka akan muncul *form* “*Checksum Hash* MD5 Dari *File*”, yang berfungsi untuk melakukan pembangkitan *checksum hash* MD5 dari *file* yang berada di dalam perangkat Android tersebut. Gambar IV.7 berikut adalah *form Checksum Hash* MD5 Dari File:

Gambar IV.10. Form Pembangkitan Hash MD5 Dari File

Apabila tombol "Bandingkan *Hash* MD5" di klik, maka akan muncul *form* "Pembandingan *Hash* MD5", yang berfungsi untuk melakukan pembandingan antara *hash* MD5 yang otentik/asli dengan hash MD5 dari teks/*file* yang pengguna Android tersebut terima/miliki di dalam perangkat Android-nya. Gambar IV.8 berikut adalah *form Checksum Hash* MD5 Dari File:

Gambar IV.11. Form Pembandingan *Hash* MD5

IV.3. Hasil Uji Coba

Setelah berhasil terinstalasi, Penulis melakukan uji coba aplikasi dengan beberapa kondisi. Yang pertama, penulis akan membangkitkan *hash* MD5 dari sebuah teks. Dalam kasus ini, teks yang di-*input* merupakan sebuah alamat *email*. Penulis akan membangkitkan hash MD5 dari *input* tersebut untuk mendapatkan *string hash* MD5 otentik dari input tersebut. Klik pada *textbox*, kemudian *input* alamat *email* yang dimaksud. Kemudian kita bisa langsung menekan tombol "Bangkitkan *Hash* MD5" untuk mendapatkan *hash* MD5 dari *input* tersebut. Nantinya *string hash* MD5 yang berhasil dibangkitkan akan tampil di *textview* yang berada di bagian bawah tampilan aplikasi.

Gambar IV.12. Pembangkitan *hash* MD5 dari teks

Selanjutnya untuk dapat menyalin *hash* MD5 yang berhasil dibangkitkan untuk nantinya dapat dibandingkan di form "Bandingkan *Hash* MD5", kita dapat

menekan tombol “Salin *Hash* ke *Clipboard*”. Nantinya sebuah pop-up bertuliskan “*hash* tersalin” akan muncul. Ini menandakan hash MD5 tersebut berhasil disalin ke *clipboard* dan siap ditempelkan nantinya ke bagian *form* “Bandingkan *Hash* MD5”. Kesimpulan dari kondisi uji coba ini, seberapapun panjangnya teks yang pengguna *input* ke dalam *textbox*, hasil *digest*/pembangkitan hash MD5 akan tetap sepanjang 32 karakter sesuai fungsi MD5 itu sendiri, bahkan *empty string* juga memiliki nilai *hash* MD5.

Gambar IV.13. Hash berhasil tersalin ke *Clipboard*

Adapun analisa hasil pada pembangkitan hash MD5 berbasis teks, dapat dilihat pada tabel IV.1 dibawah ini:

Tabel IV.1. Analisa Hasil Pada *Hash* MD5 dari Teks

ANALISA HASIL PADA HASH MD5 DARI TEKS		
Teks	**Status**	**Hash MD5**
asri.biantoro@gmail.com	Teks asli	5083c8a02a69aa2a96ee14fba66a4c3d
asri.biantora@gmail.com	Teks yang diterima user	a562db3c722e41dbc1613f8de37f7ac3
Hasil:		Hash tidak sama

Dapat terlihat dari tabel pengujian di atas, secara kasat mata tidak ada perbedaan yang mencolok antara teks yang asli dengan teks yang diterima oleh user. Perbedaan terjadi pada substitusi huruf “o” dengan huruf “a”. Perubahan satu huruf ini memicu perubahan *hash* MD5, yang dapat dipastikan teks telah termodifikasi saat sudah berada di tangan penerimanya.

Kondisi pengujian kedua yaitu membangkitkan hash MD5 dari berkas file yang diterima/dimiliki oleh pengguna Android di dalam perangkatnya. Sebagai contoh, penulis menyiapkan dua buah *file notepad*, yang kedua file tersebut sama-sama bernama Credential.txt (berekstensi .txt). Yang membedakan keduanya hanya isi teks rahasia yang ada di dalamnya. Benarkah dengan nama file yang sama, hash MD5 yang dibangkitkan akan menghasilkan string hash MD5 yang sama pula? Gambar IV.14 berikut adalah isi dari Credential.txt, yang merupakan file otentik/asli milik pengguna:

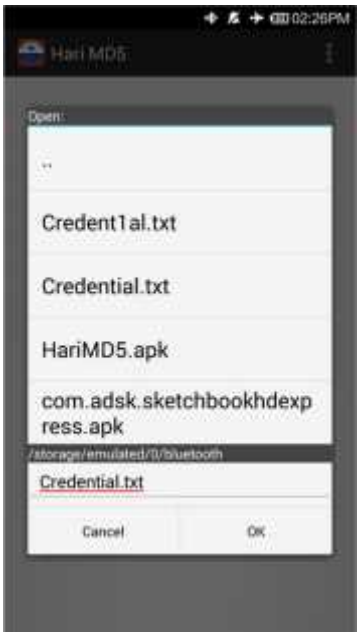
Gambar IV.14. File *notepad* Credential.txt yang otentik/asli

Sementara itu, gambar IV.15 berikut adalah isi dari Credential.txt, yang merupakan file yang telah termodifikasi isinya:

Gambar IV.15. File *notepad* Credential.txt yang termodifikasi

Dapat terlihat jelas bahwa pada kolom *password*, data *password* pengguna telah berubah sedikit saja, sebanyak satu karakter. Sekarang kita akan uji kedua file tersebut di dalam aplikasi Android penulis. Buka *form* "Bangkitkan *Hash* MD5 dari *File*." Klik tombol "Buka *File*", maka sebuah dialog Buka *File* akan terbuka agar pengguna dapat mencari file yang dimaksud melalui pilihan direktori yang muncul. File Credential.txt milik penulis berada di direktori

“/storage/emulated/0/bluetooth”. Harap diingat bahwa ini merupakan file otentik/asli milik pengguna. Klik pada file yang dimaksud, kemudian pilih OK.



Gambar IV.16. Dialog Buka *File*

Pada textview “File Anda” akan muncul keterangan *filepath*/direktori dari file yang pengguna pilih. Kemudian kita bisa langsung menekan tombol “Bangkitkan *Hash* MD5” untuk mendapatkan *hash* MD5 dari *file* tersebut. Nantinya *string hash* MD5 yang berhasil dibangkitkan akan tampil di *textview* yang berada di bagian bawah tampilan aplikasi. *String hash* MD5 yang berhasil dibangkitkan memiliki nilai “71c0c0cd24d0d966a80fa42e92002bf2” (tanpa tanda petik). Selanjutnya untuk dapat menyalin *hash* MD5 yang berhasil dibangkitkan untuk nantinya dapat dibandingkan di form “Bandingkan *Hash* MD5”, kita dapat menekan tombol “Salin *Hash* ke *Clipboard*”.

Gambar IV.17. Pembangkitan *hash* MD5 dari *file* (otentik)

Kemudian kita juga akan mencoba membangkitkan hash MD5 dari file Credential.txt yang sudah termodifikasi. Buka file tersebut, kemudian bangkitkan *hash* MD5-nya.

Gambar IV.18. Pembangkitan *hash* MD5 dari *file* (termodifikasi)

String hash MD5 yang berhasil dibangkitkan memiliki nilai "a7bd6d84212b16ac33e8ada74abf5f8e". Adapun analisa hasil pada pembangkitan hash MD5 berbasis *file* berekstensi .txt, dapat dilihat pada tabel IV.2 dibawah ini:

Tabel IV.2. Analisa Hasil Pada *Hash* MD5 dari *File* Berekstensi .txt

ANALISA HASIL PADA HASH MD5 DARI FILE .TXT		
File	**Status**	**Hash MD5**
Credential.txt	File asli	71c0c0cd24d0d966a80fa42e92002bf2
Credential.txt	File yang diterima user	a7bd6d84212b16ac33e8ada74abf5f8e
Hasil:		Hash tidak sama

Kondisi uji coba ini menunjukkan bahwa nama file yang sama sekalipun tidak mempengaruhi string hash MD5 yang akan dibangkitkan, tapi sebaliknya, isi dari file milik penggunalah yang sangat mempengaruhi perubahan signifikan dari string hash MD5.

Bagaimana jika *user* mengunduh file berekstensi .exe dari situs FileHippo.com, yang sudah tersedia informasi hash MD5 dari setiap file yang mereka sediakan di server mereka? Penulis kali ini mencoba mengunduh sebuah file aplikasi bernama CCleaner, versi 5.10. Menurut situs FileHippo.com, hash valid dari file tersebut adalah "bd4122d5b2830c8db3992cb9d2920f0e" (tanpa tanda petik). Setelah penulis berhasil mengunduh aplikasi tersebut melalui perangkat Android, penulis melakukan uji coba pembandingan hash.

Tabel IV.3. Analisa Hasil Pada *Hash* MD5 dari *File* Berekstensi .exe

ANALISA HASIL PADA HASH MD5 DARI FILE .EXE		
File	**Status**	**Hash MD5**
ccsetup510.exe	File asli	bd4122d5b2830c8db3992cb9d2920f0e
Ccsetup510.exe	File yang diterima user	bd4122d5b2830c8db3992cb9d2920f0e
Hasil:		Hash sama

Dari hasil pengujian tersebut, terlihat bahwa *hash* sama. Ini menandakan selama proses transfer/pengunduhan, tidak terjadi modifikasi saat sampai ke tangan penerima. Hal ini juga menandakan informasi berupa hash MD5 valid yang diberikan oleh situs FileHippo dapat dipercaya dan dapat digunakan sebagai bahan acuan dalam pembandingan *hash* nantinya.

Pengujian dari file berekstensi .exe berlanjut dengan cara penulis mencoba untuk mengunduh file CCleaner versi yang sama dari situs torrent The Pirate Bay. Apakah hasilnya akan sama? Adapun analisa hasil pada pembangkitan hash MD5 berbasis *file* berekstensi .exe dari situs yang berbeda, dapat dilihat pada tabel IV.4 dibawah ini:

Tabel IV.4. Analisa Hasil Pada *Hash* MD5 dari *File* Berekstensi .exe

ANALISA HASIL PADA HASH MD5 DARI FILE .EXE		
File	**Status**	**Hash MD5**
ccsetup510.exe	File asli	bd4122d5b2830c8db3992cb9d2920f0e
ccsetup510.exe	File yang diunduh dari torrent	f119524883af4bac56581ed77ceef828
Hasil:		Hash tidak sama

Aplikasi yang diunduh serta versi yang digunakan sudah sama, tetapi kenapa hasil *hash string*-nya berbeda? Setelah mencoba membaca keterangan informasi dari file CCleaner yang ada di situs The Pirate Bay, ternyata di dalam aplikasi berekstensi .exe tersebut sudah tersusup *crack* agar semua fitur berbayar di aplikasi tersebut bisa digunakan secara bebas. Di dalam dunia keamanan data, *crack* adalah sebuah program yang membebaskan pengguna menggunakan sebuah aplikasi secara ilegal, atau dengan kata lain bajakan. Biasanya, tidak sedikit kasus di mana sebuah aplikasi yang telah di-*crack* tersusupi *trojan horse* di dalamnya, *malware* yang sangat berbahaya yang dapat mencuri identitas dan informasi di dalam perangkat Android/komputer tanpa merusak sistem perangkat tersebut.

Pada pengujian file berekstensi .rar, kita akan melihat apakah file terkompresi seperti .rar yang ter-*password* memiliki hash yang sama dengan .rar yang tidak ter-*password* lagi, walaupun isinya terlihat sama?

Tabel IV.5. Analisa Hasil Pada *Hash* MD5 dari *File* Berekstensi .rar

ANALISA HASIL PADA HASH MD5 DARI FILE .RAR		
File	**Keterangan**	
kirim.rar **(file asli terpassword)**	Isi file	- ccleanersetup510.exe - Credential.txt
	Password	12345
	Hash MD5	b44b5be60af7233d27d71b7d9932650c
kirim.rar **(file diterima user)**	Isi file	- ccleanersetup510.exe - Credential.txt
	Password	(kosong)

	Hash MD5	097ef250177eeccdacf5fd4601f52e75
Hasil:		Hash tidak sama

Dari hasil pengujian ini, terlihat bahwa file .rar yang dilindungi *password* memiliki *hash string* MD5 yang berbeda dengan *hash string* MD5 dari file .rar yang *password*-nya telah di-*bypass* atau sudah dihilangkan. Walaupun isinya sama, namun kecurigaan penerima mengenai keamanan data yang diterima akan muncul ketika si pengirim mengatakan bahwa file .rar yang mereka kirim ternyata di beri *password*, sementara file yang diterima oleh si penerima file yang berhak tidak terlindungi oleh *password* lagi.

Karena mendapatkan hasil *hash* yang tidak sama, penerima file mencoba mendapatkan kembali file yang dikirim oleh si pengirim. Hasil pengujiannya terdapat pada tabel IV.6 berikut.

Tabel IV.6. Analisa Hasil Pada *Hash* MD5 dari *File* Berekstensi .rar

ANALISA HASIL PADA HASH MD5 DARI FILE .RAR		
File	**Keterangan**	
kirim.rar **(file asli terpassword)**	Isi file	- ccleanersetup510.exe - Credential.txt
	Password	12345
	Hash MD5	b44b5be60af7233d27d71b7d9932650c
kirim.rar **(file diterima user)**	Isi file	- ccleanersetup510.exe - Credential.txt
	Password	12345

	Hash MD5	b44b5be60af7233d27d71b7d9932650c
Hasil:		Hash sama

Dengan metode MD5 ini, pengguna semakin dipermudah untuk mendeteksi dan mengindentifikasi perubahan sekecil apapun pada sebuah file. Selain dari pada file *text*, pengguna juga bisa memanfaatkan aplikasi ini untuk mengidentifikasi beberapa macam jenis file, seperti file kompresi seperti *.rar dan *.zip. Selama pengguna mendapatkan/memiliki hash MD5 dari file yang otentik, pengguna Android dapat membandingkannya melalui form “Bandingkan Hash MD5” yang berikutnya akan dibahas.

Kondisi pengujian ketiga adalah membandingkan antara hash MD5 yang otentik/asli dengan hash MD5 dari file/teks yang diterima pengguna. Penulis kali ini memanfaatkan kembali hash MD5 dari file Credential.txt yang diuji coba sebelumnya. Buka form “Bandingkan Hash MD5”, kemudian pertama-tama tempelkan hash MD5 dari file Credential.txt yang otentik.

Gambar IV.19. *Input hash* MD5 otentik

Selanjutnya tempelkan hash MD5 dari file Credential.txt yang dimiliki pengguna namun sudah termodifikasi.

Gambar IV.20. *Input hash* MD5 dari data yang termodifikasi

Untuk mulai membandingkan, klik tombol “Bandingkan Hash MD5”. Jika hash MD5 dari keduanya berbeda, akan muncul pemberitahuan berupa “Hash asli dengan hash milik user tidak sama. Data Anda dicurigai telah termodifikasi dari data aslinya.” Terlihat jelas dari sini, jika hash dari keduanya dinyatakan berbeda, maka sudah pasti isi file-nya sudah termodifikasi saat sampai di tangan penerima data. Dalam dunia kriptografi, “termodifikasi” di sini bisa jadi terjadi penyisipan, penghapusan, dan pensubstitusi data lain ke dalam data yang sebenarnya, sebelum diterima oleh penerima yang berhak.

Pengirim tentu sebelumnya harus menyiapkan hash MD5 dari data yang asli agar bisa dibandingkan oleh si penerima data. Jika pengguna menerima file tersebut namun setelah diverifikasi hash ternyata berbeda, pengguna bisa mencoba menghubungi kembali sang pemilik asli file, atau mentransfer/mengunduh kembali file untuk verifikasi ulang.

Gambar IV.21. Pemberitahuan hash tidak sama/tidak valid

Sementara itu, jika hash keduanya ternyata sama, maka akan muncul pemberitahuan berupa “Hash asli dengan hash milik user sama. Integritas file Anda terjamin keotentikannya.”

Gambar IV.22. Pemberitahuan hash sama/valid

IV.4. Kelebihan dan Kekurangan

Didalam pembuatan skripsi ini penulis menyadari adanya kelebihan dan kekurangan dikarenakan keterbatasan kemampuan penulis. Berikut adalah kelebihan dan kekurangan dari Perancangan Aplikasi Checksum Hash Berbasis Android.

IV.4.1. Kelebihan

Adapun kelebihan perancangan aplikasi di dalam skripsi ini antara lain:

1. Mobilisasi yang tinggi mengakibatkan meningkatnya penggunaan perangkat Android di berbagai tempat, sehingga segala aktivitas transfer data/berkas dilakukan langsung melalui perangkat para pengguna Android. Aplikasi ini sangat berguna bagi pengguna sebagai pihak “pengirim” file, terutama bagi pengguna yang sangat menjaga integritas data yang hendak dikirimnya dan sampai dalam keadaan “utuh”. Utuh di sini dalam artian file yang dikirimnya tidak mengalami proses pengubahan apapun (modifikasi, injeksi virus, dll) sebelum sampai ke tangan “penerima” file yang berhak.
2. Dengan adanya aplikasi ini, pengguna dapat membangkitkan *hash string* MD5 dari sebuah *file* yang telah diunduh, sehingga selanjutnya dapat dicocokkan dengan keterangan *hash string valid*/otentik yang sudah disediakan oleh pengirim/situs penyedia *file* (jika data tersebut di dapat dari Internet).
3. Aplikasi ini sangat ringan, baik dalam hal ukuran file aplikasi itu sendiri maupun prosesnya yang tidak membebani resources pada perangkat Android.

IV.4.2. Kekurangan

Adapun kekurangan perancangan aplikasi di dalam skripsi ini antara lain:

1. Tergantung dari spesifikasi perangkat Android yang digunakan, proses pembangkitan hash MD5 pada file berukuran besar (>250MB) cukup berlangsung lama.

2. Tidak semua karakter khusus dan simbol bisa di-input ke dalam *textbox input* teks di dalam form “Bangkitkan Hash MD5 dari Teks” (seperti karakter *backspace)* karena keterbatasan karakter yang tersedia dari *keyboard* bawaan perangkat Android.
3. Untuk file aplikasi berekstensi .exe, tidak semua *provider*/penyedia file menyertakan informasi hash MD5 dari file ada di server mereka, sehingga menyulitkan penerima file untuk melakukan *checksum*.
4. Tampilan aplikasi yang sangat sederhana, dimaksudkan agar para pengguna dapat menggunakan aplikasi ini tanpa kebingungan dan kendala yang berarti.