UNIVERSITAS INDONESIA

PENGEMBANGAN SISTEM KEAMANAN SIMPLE-O

SKRIPSI

GREGORIUS HANDOYO
06 06 07 8342

FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK ELEKTRO
DEPOK
JUNI 2010

UNIVERSITAS INDONESIA

PENGEMBANGAN SISTEM KEAMANAN SIMPLE-O

SKRIPSI

Diajukan sebagai salah satu syarat untuk memperoleh gelar Sarjana Teknik

GREGORIUS HANDOYO
06 06 07 8342

FAKULTAS TEKNIK
PROGRAM STUDI TEKNIK ELEKTRO
DEPOK
JUNI 2010

HALAMAN PERNYATAAN ORISINALITAS

Skripsi ini adalah hasil karya saya sendiri,

dan semua sumber baik yang dikutip maupun dirujuk

telah saya nyatakan dengan benar.

Nama	**: Gregorius Handoyo**
NPM	**: 0606078342**
Tanda Tangan	**:** 
Tanggal	**: 9 Juli 2010**

HALAMAN PENGESAHAN

Skripsi ini diajukan oleh :
Nama : Gregorius Handoyo
NPM : 0606078342
Program Studi : Teknik Komputer
Judul Skripsi : Pengembangan Sistem Keamanan Simple-O

Telah berhasil dipertahankan di hadapan Dewan Penguji dan dterima sebagai bagian persyaratan yang diperlukan untuk memperoleh gelar Sarjana Teknik pada Program Studi Teknik Komputer, Fakultas Teknik, Universitas Indonesia.

DEWAN PENGUJI

Pembimbing : Dr. Ir. Anak Agung Putri Ratna M.Eng ( )

Penguji : Muhammad Salman ST., MIT

Penguji : Ir. Endang Sriningsih MT, Si ( )

Ditetapkan di : Depok
Tanggal : 9 Juli 2010

KATA PENGANTAR

Puji dan syukur penulis panjatkan kehadirat Tuhan Yang Maha Esa karena dengan rahmat dan karunia-Nya lah penulis dapat menyelesaikan buku skripsi ini. Mulai dari proses pembelajaran dan analisis yang telah dijalani dan proses penyusunan dari buku skripsi ini, penulis ingin mengucapkan terima kasih kepada:

1. Dr. Ir. Anak Agung Putri Ratna M.Eng, selaku pembimbing telah menyediakan waktu, tenaga, dan pikiran untuk mengarahkan saya dalam penyusunan skripsi ini;
2. Muhammad Salman ST., MIT, yang telah membantu dalam memberikan ide untuk pengembangan keamanan aplikasi Simple-O.
3. Orang tua dan keluarga yang telah memberi dukungan material dan moral;
4. Dedi (Elektro 2004) yang telah mengajarkan tentang keamanan *web*;
5. Ari Nugraheni yang telah membantu dan memberi dukungan dalam penyusunan skripsi ini;
6. Teman-teman satu bimbingan dan satu angkatan saya: Boma, Bram, Mei, dan Vivi; dan
7. Seluruh keluarga besar Sivitas Akademika Departemen Teknik Elektro Fakultas Teknik Universitas Indonesia yang tidak dapat saya sebutkan satu persatu.

Akhir kata, saya berharap Tuhan Yang Maha Esa berkenan membalas segala kebaikan semua pihak yang telah membantu. Semoga skripsi ini membawa manfaat bagi pengembangan ilmu.

Depok, 9 Juli 2010

Penulis

HALAMAN PERNYATAAN PERSETUJUAN PUBLIKASI TUGAS AKHIR UNTUK KEPENTINGAN AKADEMIS

Sebagai sivitas akademika Universitas Indonesia, saya yang bertanda tangan di bawah ini:

Nama	: Gregorius Handoyo
NPM	: 0606078342
Program Studi	: Teknik Komputer
Departemen	: Teknik Elektro
Fakultas	: Teknik
Jenis Karya	: Skripsi

demi pengembangan ilmu pengetahuan, menyetujui untuk memberikan kepada **Universitas Indonesia Hak Bebas Royalti Noneksklusif (*Non-exclusive Royalty Free Right*)** atas karya ilmiah saya yang berjudul :

PENGEMBANGAN SISTEM KEAMANAN SIMPLE-O

beserta perangkat yang ada (jika diperlukan). Dengan Hak Bebas Royalti Noneksklusif ini Universitas Indonesia berhak menyimpan, mengalihmedia/formatkan, mengelola dalam bentuk pangkalan data (database), merawat, dan memublikasikan tugas akhir saya selama tetap mencantumkan nama saya sebagai penulis/pencipta dan sebagai pemilik Hak Cipta.

Demikian pernyataan ini saya buat dengan sebenarnya.

Dibuat di : Depok

Pada Tanggal : 9 Juli 2010

Yang Menyatakan

(Gregorius Handoyo)

ABSTRAK

Nama : Gregorius Handoyo
Program Studi : Teknik Komputer
Judul : PENGEMBANGAN SISTEM KEAMANAN SIMPLE-O

Skripsi ini membahas mengenai pengamanan pada Simple-O, sistem penilaian ujian esai secara *on-line* yang telah dikembangkan pada Departemen Elektro Univeritas Indonesia. Dengan meninjau banyaknya program yang dapat menembus keamanan suatu *website*, maka diperlukan pengamanan yang lebih handal pada aplikasi Simple-O tanpa menimbulkan kesulitan bagi pengguna. Permasalahan keamanan yang perlu diperhatikan dari suatu *website* meliputi keamanan *server* dan keamanan aplikasi. Pada skripsi ini akan diterapkan beberapa keamanan dari segi aplikasi. Beberapa penerapan sistem keamanan yang berhasil dilakukan, menyebabkan sistem bekerja lebih lama terutama dalam proses autentifikasi. Rata-rata peningkatan terjadi berkisar antara 28% - 36% atau sekitar 0,0087 – 0,0272 detik. Diharapkan dengan melakukan pengamanan yang lebih handal tersebut dapat mencegah terjadinya serangan dari luar.

Kata Kunci:
Penilaian ujian esei, simple-o, keamanan, *website*.

ABSTRACT

Name : Gregorius Handoyo
Study Program : Computer Engineering
Title : SECURITY SYSTEM DEVELOPMENT OF SIMPLE-O

This undergraduate thesis discusses about the security in Simple-o, an online essay grading system that has been developed at the Department of Electrical Engineering University of Indonesia. By reviewing the many programs that can hack the security of a website, the more reliable security needed at Simple-O applications without causing difficulties for users. Security issues that need to be considered from a website includes the server security and application security. This undergraduate thesis will discuss security issues in terms of application. Some implementation of security systems that successfully carried out, causing the system to work longer, especially in the authentication process. The average increase is occurred in the range between 28% - 36%, or about 0,0087 to 0,0272 seconds. It is expected by performing more reliable security can prevent attacks from outside.

Keywords:
Essay grading, simple-o, security, website.

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB 1
PENDAHULUAN

1.1 Latar Belakang

Perkembangan teknologi yang begitu cepat, sangat membantu kegiatan manusia setiap harinya. Salah satu diantaranya adalah teknologi informasi. Dengan teknologi informasi saat ini, dapat diperoleh segala bentuk informasi, seperti video, gambar, suara dengan cepat dan mudah. Teknologi yang paling banyak berpengaruh di masyarakat dunia saat ini adalah *internet*. Dengan kemudahan dan kecepatan akses dari teknologi ini untuk mendapatkan informasi yang baru, tidak dapat dipungkiri bahwa teknologi ini menjadi tulang punggung dalam penyebaran maupun untuk mendapatkan informasi. Salah satu diantaranya adalah *website.*

Website atau situs dapat diartikan sebagai kumpulan halaman yang menampilkan informasi berupa data teks, data gambar diam atau bergerak, data animasi, suara, video dan atau gabungan dari semuanya, baik yang bersifat statis maupun dinamis yang membentuk satu rangkaian bangunan yang saling terkait dimana masing-masing dihubungkan dengan jaringan-jaringan halaman. *Website* bersifat statis apabila isi informasi *website* tetap, jarang berubah, dan isi informasinya searah hanya dari pemilik *website*. Sedangkan *website* bersifat dinamis apabila isi informasi *website* selalu berubah-ubah, dan isi informasinya interaktif dua arah berasal dari pemilik serta *user website*. Contoh *website* statis adalah berisi profil perusahaan, sedangkan *website* dinamis adalah seperti *Facebook, Friendster*, dan sejenisnya.

Banyak sekali *website* yang dapat dilihat di *internet*. Tujuan dari seseorang yang menggunakan sebuah *website* tersebut bermacam-macam, mulai dari melakukan penjualan, memperkenalkan perusahaan baru, membagi informasi serta ilmu pengetahuan, hingga transaksi *on-line*. Penggunaan internet tersebut dapat mempersingkat alokasi waktu yang kita butuhkan untuk melakukan suatu pekerjaan, di samping itu juga dapat menghemat biaya.

Seiring dengan perkembangan fungsi dari *website* bagi kehidupan manusia, maka berkembang pula tingkat kejahatan pada dunia *internet*. Kejahatan

yang dilakukan tersebut antara lain pencurian *account* dengan tujuan tertentu seperti misalnya untuk penipuan, manipulasi data, sampai menghancurkan suatu sistem. Dari permasalahan-permasalahan tersebut, maka seseorang yang akan membuat suatu *website* perlu melindungi *website* dengan cara menambahkan sistem keamanan di dalamnya.

Pada skripsi ini akan mencoba diterapkan sistem keamanan suatu *web* penilaian ujian esei yang akan digunakan pada lembaga pendidikan secara *on-line*. Untuk meminimalisir terjadinya serangan dari luar terhadap sistem penilaian tersebut, maka akan dilakukan pengamanan yang baik pada *website* tersebut.

1.2 Tujuan Penulisan

Tujuan dari skripsi ini adalah untuk pengembangan sistem keamanan pada aplikasi Simple-O saat ini, guna memberikan batasan-batasan yang jelas terhadap pengguna serta mengurangi tindakan *hacking* terhadap aplikasi tersebut dan untuk menganalisa pengaruh performa pada aplikasi Simple-O setelah diimplementasikan sistem keamanan.

1.3 Pembatasan Masalah

Pada skripsi ini sistem keamanan aplikasi Simple-O yang dibahas dibatasi pada penerapan rancangan keamananan aplikasi Simple-O ditinjau dari *script programming*.

1.4 Metodologi Penulisan

Metode penulisan yang digunakan pada buku skripsi ini adalah:

a. Studi literatur, yaitu dengan mencari buku-buku yang digunakan untuk referensi.
b. Konsultasi ahli-ahli *web security*, serta beberapa forum diskusi.
c. Teknik observasi, yaitu dengan meninjau kembali secara cermat sistem keamanan yang telah diimplementasikan.

1.5 Sistematika Penulisan

Skripsi ini terdiri dari lima bab, dimana masing-masing bab akan menjelaskan sebagai berikut:

a. Bab 1: Pendahuluan

   Pada bab ini, akan dijelasan mengenai Latar Belakang, Tujuan, Pembatasan Masalah, Metodologi Penulisan, dan Sistematika penulisan.

b. Bab 2: Keamanan Web dan Simple-O

   Pada bab ini, akan dijelaskan mengenai Celah Keamanan *Web*, Sistem Ujian Esei, SIMPLE-O, *Essay Grading* dengan metode LSA, SVD, dan Program Pendukung dari *Essay Grading.*

c. Bab 3: Perancangan Keamanan Simple-O

   Pada bab ini, akan dijelaskan mengenai Analisa Keamanan Simple-O saat ini dan perancangan beberapa sistem keamanan yang akan diterapkan pada aplikasi Simple-O.

d. Bab 4: Penerapan Sistem Keamanan Pada SIMPLE-O

   Pada bab ini, akan dijelaskan mengenai penerapan dan analisa dari sistem keamanan yang telah diterapkan pada SIMPLE-O.

e. Bab 5: Kesimpulan

   Pada bab ini, akan dijelaskan mengenai kesimpulan yang dapat diambil dari pembahasan buku skripsi ini.

BAB 2
KEAMANAN WEB DAN SIMPLE-O

2.1 Celah Keamanan *Website*

Perkembangan peranan *internet* terhadap kehidupan sehari-hari yang semakin pesat memicu munculnya niat seseorang untuk memanfaatkan internet untuk kepentingan pribadi yang merugikan orang lain. Hal ini dapat dicegah dengan cara mengenali celah apa saja yang dimiliki oleh suatu *website* yang mungkin dapat dimanfaatkan bagi para *hacker*. Dengan mengetahui letak celah yang memungkinkan seorang *hacker* dapat mengambil alih kendali suatu *website*, maka pemilik *website* dapat membuat suatu sistem keamanan yang efektif serta efisien terhadap *website* yang dibuatnya.

Open Web Application Security Project (OWASP) merupakan proyek *open source* yang dibuat dengan tujuan dapat menemukan celah tidak amannya suatu *software* dalam hal ini adalah penerapannya dalam *website* dan cara menanganinya.[1] Berikut ini beberapa celah keamanan yang dimiliki suatu *website*:

a. *Unvalidated input* [1]

Operasi yang dilakukan oleh *website* baik yang dikategorikan statis maupun dinamis menggunakan data dari HTTP *request* yang dibuat oleh pengguna. HTTP *request* tersebut terdiri dari beberapa bagian seperti *query string*, *cookie information*, *header*. Bagian dari HTTP *request* tersebut dimanipulasi oleh *hacker* untuk memboikot *website* tersebut.

Dalam hal ini manipulasi bagian dari HTTP *request* dapat diklasifikasikan sebagai berikut :

- *Cross site scripting* (XSS)

  XSS merupakan celah keamanan yang dimiliki oleh *website* dinamis. Dalam hal ini, suatu *website* dikatakan memiliki celah XSS adalah ketika *server* tidak mampu memvalidasi input yang diberikan pengguna kepada sistem. Hal ini dapat menyebabkan tampilnya suatu data yang sebenarnya tidak diijinkan oleh sistem. Dengan bagitu, *hacker* dapat memasukkan data dan mengirimkan ke web

browser tanpa harus melewati suatu validasi dan encoding data. Selain itu dengan adanya celah XSS, *hacker* dapat menjalankan skrip miliknya di *browser* target dan mengambil *user session* milik target.

- *Buffer overflows*

  Merupakan celah keamanan *website* yang ditandai dengan variabel yang tersedia pada aplikasi tidak dapat menampung masukkan yang sengaja dibuat berlebihan. Akibat dari kelebihan data tersebut akan mengisi alamat memori lain yang bukan milik variabel tersebut atau dalam hal ini disebut dengan *overwrite.*

- *Injection flaws*

  Celah ini mampu membuat *hacker* untuk melakukan injeksi terhadap basis data dari suatu *website*. Hal ini mungkin terjadi apabila pengguna memasukkan data sebagai bagian dari perintah (*query*) yang menipu interpreter untuk menjalankan perintah tersebut atau mengubah suatu data.

Hal-hal yang harus dihindari berkaitan dengan penanganan validasi *website* antara lain :

- Penggunaan batas-batas yang jelas pada sistem validasi

  Membatasi suatu format atau pola untuk nilai yang diijinkan dan memastikan input tersebut sesuai dengan format yang telah ditentukan sebelumnya.

b. Broken Access Control [1]

Beberapa *website* membuat beberapa *role* dan *level* yang berbeda untuk para pengguna yang memanfaatkan *website* tersebut. Dari *role* dan *level* yang berbeda tersebut informasi yang dapat diambil dari *website* pun berbeda tergantung pada kategori yang dimiliki pengguna. Salah satu contohnya, banyak aplikasi yang terdapat *user role* dan *admin role* dimana hanya *admin role* yang diijinkan untuk mengakses halaman khusus atau melakukan *action administration*.

Masalah yang berhubungan dengan *access control* adalah:

- *Insecure Ids*

  Pembuatan identitas maupun kata sandi hendaknya merupakan suatu kombinasi yang sulit untuk ditebak oleh *hacker*. Hal ini sedikit mencegah terjadinya pembobolan situs oleh *hacker*.

- *File* permissions

  Kebanyakan situs dan aplikasi *server* percaya kepada sebuah *file* yang menyimpan daftar dari pengguna yang terotorisasi dan *resources* mana saja yang dapat dan/atau tidak dapat diakses. Apabila *file* ini dapat dibaca dari luar, maka *hacker* dapat memodifikasi dengan mudah untuk menambahkan dirinya pada daftar pengguna yang diijinkan.

Beberapa hal yang bisa dilakukan untuk mengatasi masalah yang berhubungan dengan *access control* :

- *Insecure Ids.*

  Dengan mengembangkan penyaringan atau komponen yang dapat dijalankan pada *file* tertentu dapat menjamin hanya pengguna yang terotorisasi yang dapat mengakses.

- *File Permission*

  Untuk menghindari adanya celah pada *file* permission maka data-data tersebut harus berada pada lokasi yang tidak dapat diakses oleh *web browser* dan hanya role tertentu yang dapat mengakses.

c. Broken Authentication dan Session Management [1]

Autentifikasi dan manajemen sesi menunjuk kepada semua aspek dari pengaturan autentifikasi pengguna dan manajemen sesi yang aktif. Berikut ini beberapa hal yang perlu diperhatikan :

- *Password strength*

  Pada saat pembuatan identitas dan kata sandi hendaknya terdapat tingkat minimal dari kekuatan kata sandi pada aplikasi *website*. Dengan begitu pengguna dapat memperkirakan apakah kata sandi

yang telah dibuat telah memenuhi standar keamanan. Tingkat kekuatan kata sandi dapat dilihat dari banyaknya karakter yang digunakan dan kompleksitas dari kombinasi karakter yang dipakai.

- *Password use*

  Suatu *website* kadang memerlukan suatu pembatasan tenggang waktu tertentu kepada pengguna untuk melakukan akses. Hal ini dapat menghindarkan adanya serangan *brute force* yang merupakan sebuah teknik penyerangan terhadap suatu *website* yang menggunakan percobaan terhadap semua kata sandi yang mungkin digunakan.

- *Password storage*

  Penyimpanan kata sandi yang telah dibuat hendaknya diubah dalam bentuk yang telah terenkripsi kemudian disimpan di dalam suatu basis data dan tidak disimpan di dalam aplikasi.

- *Session ID Protection*

  Penggunaan *session id* cukup penting karena bertujuan untuk mengidentifikasi pengguna yang masuk ke dalam sesi, namun hal ini dapat menjadi boomerang bagi pemilik *website* apabila *session id* dapat dilihat oleh orang lain pada jaringan yang sama. Hal ini dapat membuat orang tersebut menjadi *client*. Pencegahannya dapat dilakukan dengan melakukan komunikasi antara *server* dan *client* pada sebuah *SSL-protected channel*.

- CAPTCHA

  CAPTCHA merupakan singkatan dari *Completely Automated Public Turing Test to Tell Computers and Humans Apart* yang dapat digunakan dalam meningkatkan keamanan suatu *website*. Pada tahun 2000 CAPTCHA diciptakan oleh Luis von Ahn, Manuel Blum, Nicholas Hopper dan John Langford dari *Carnegie Mellon University*.[2] Aplikasi CAPTCHA pada suatu website biasanya dimanfaatkan dalam *form login*, pengisian komentar atau buku tamu website tersebut, proses verifikasi dan sebagainya.

Gambar 2.1 Salah satu tampilan dari program CAPTCHA (phpcaptcha.org)

CAPTCHA digunakan sebagai program uji respon yang digunakan pada komputer untuk memastikan bahwa penggunanya adalah manusia dan bukanlah suatu komputer lain. Uji yang dilakukan mengharuskan pengguna mengetik ulang huruf atau angka yang terdistorsi yang dimuat dalam suatu gambar yang ditampilkan oleh program CAPTCHA pada layar komputer. Alasan pemilihan metode ini karena pada kenyataannya sulit bagi komputer untuk mengambil teks dari suatu gambar dan yang bisa melakukan hal tersebut hanya manusia. Bagi komputer, sebuah gambar hanyalah kumpulan kode-kode warna dari setiap pixelnya. Dibutuhkan proses yang cukup rumit untuk bisa mengenali objek pada gambar, apalagi untuk memahami arti sebuah gambar. Sedangkan bagi manusia hanya dalam hitungan seper sekian detik kode dalam gambar sudah terbaca.

Penggunaan CAPTCHA pada suatu *website* mempunyai fungsi antara lain :

- Menghindari *comment spam* [2]

  Sistem keamanan yang memanfaatkan CAPTCHA dibutuhkan pada saat sekarang karena semakin banyak program *comment spam* yang dapat menyerang *website*. Program tersebut dapat menampilkan komentar yang tidak selayaknya muncul pada sebuah *website*. *Spam* ini biasanya secara otomatis melakukan kirim komentar ke buku tamu atau halaman komentar *website*.

- Melindungi pendaftaran *website* [2]

  CAPTCHA juga dapat menghindarkan proses pendaftaran website yang dilakukan oleh *bot,* dimana *bot* ini dapat mendaftarkan *account* hingga ribuan kali setiap menit pada *website* yang menawarkan *account* gratis. Dengan menggunakan CAPTCHA maka pendaftaran *account* gratis ini hanya dapat dilakukan oleh manusia.

- Mencegah *dictionary attack* [2]

  CAPTCHA juga dapat digunakan untuk mencegah serangan kamus (*dictionary attack*) dalam sistem sandi. Hal yang dilakukan oleh CAPTCHA dalam menghindarkan serangan tersebut adalah dengan mencegah komputer untuk beriterasi melalui seluruh ruang sandi dengan syarat harus memecahkan kode CAPTCHA setelah sejumlah *login* gagal. Ini lebih baik dibandingkan dengan cara klasik yaitu mengunci *account* setelah sejumlah *login* gagal, karena melakukan hal itu memungkinkan penyerang dapat mengunci *account* pada kondisi berikutnya.

d. *Insecure storage* [1]

Proses enkripsi diperlukan untuk mengubah kata sandi menjadi suatu bentuk yang tidak dapat dibaca oleh manusia. Hal ini penting dilakukan untuk menutup celah keamanan suatu *website*. Namun ada beberapa hal yang perlu diperhatikan dalam proses enkripsi yang dapat membuat proses ini tidak memberikan manfaat sebagaimana mestinya, antara lain :

- Kesalahan untuk mengenkripsi data penting
- Kurang amannya lokasi penyimpanan data
- Kurangnya penghitungan dari randomisasi
- Kesalahan pemilihan algoritma
- Mencoba untuk menciptakan algoritma enkripsi yang baru

e. Denial of Service [1]

Denial of Service merupakan suatu bahaya keamanan yang ditimbulkan oleh jumlah permintaan yang dikirimkan oleh *hacker* ke sistem yang cukup besar dalam waktu bersamaan sehingga kerja *server* menjadi berat dan secara tidak langsung mencegah pengguna yang lain memperolah akses layanan dari sistem tersebut. Hal ini dapat dianalogikan bahwa sumber yang disediakan oleh sistem habis karena permintaan yang terlalu banyak sehingga sistem tidak dapat menjalankan fungsinya dengan baik. Serangan *DoS* mampu menghabiskan bandwidth yang ada pada *server*. Selain itu dapat juga menghabiskan *memory*, koneksi basis data, dan sumber yang lain.

Teknik yang digunakan dalam *denial of service* antara lain :

- *Traffic Flooding*

  Teknik ini mengijinkan *hacker* untuk membuat suatu kemacetan pada lalu lintas jaringan dengan banyak data dengan begitu pengguna yang terdaftar yang lain tidak dapat mengakses layanan dari sistem tersebut.

- *Request Flooding*

  Teknik ini dilakukan dengan cara membuat permintaan yang jumlahnya cukup banyak ke suatu layanan jaringan sehingga permintaan dari pengguna terdaftar yang lain tidak dapat dilayani oleh sistem.

- Mengubah informasi konfigurasi sistem atau perusakan fisik agar komunikasi antara *server* dan *client* terganggu.

Untuk mengatasi bahaya keamanan ini adalah sangat sulit, namun ada beberapa hal yang masih mungkin dilakukan yaitu :

- Membuat *load quota* yang membatasi *load* data yang dapat diakses pengguna sehingga jumlah sumber yang dapat diakses menjadi minimal.

- Mendesain suatu *website* yang mengijinkan penggunanya yang telah terotorisasi yang memiliki akses ke basis data

f. *Insecure Configuration Management* [1]

Celah keamanan selain dimiliki oleh aplikasi *web* juga dimiliki oleh *server*. Beberapa hal yang perlu diperhatikan dari *server* yang dapat menjadi celah bagi *hacker* untuk melakukan pembobolan antara lain :

- Administrator tidak melakukan *patch software* yang ada pada *server*.
- *Server* bisa menampilkan list dari direktori atau juga serangan berupa *directory traversal.*
- *File-file backup* atau contoh *file, file-file* script, *file* konfigurasi yang tertinggal / tidak perlu.
- Hak akses direktori atau *file* yang salah.
- Penggunaan default account dan default password.
- Fungsi administratif atau fungsi *debug* yang bisa diakses.
- Adanya pesan *error* yang informatif dari segi teknis.
- Kesalahan konfigurasi *SSL certificate* dan pengaturan enkripsi.
- Penggunaan *default certificate.*

2.2 Perkembangan Ujian Di Indonesia

Seiring dengan perkembangan kemajuan teknologi, banyaknya kegiatan manusia yang dibantu dengan memanfaatkan teknologi tersebut semakin meningkat. Pada dunia pendidikan, kecanggihan teknologi yang sudah ada sampai saat ini seperti *e-book*, *e-learning*, sistem penilaian *multiple choice* pada Ujian Nasional di Indonesia, dsb, yang dapat mempermudah manusia dalam melaksanakan kegiatannya sehari-hari.

Terdapat 2 jenis bentuk ujian, yaitu bentuk objektif dan bentuk subjektif. Contoh dari bentuk objektif adalah ujian *multiple choice* (pilihan ganda) dan jawaban singkat. Sedangkan, contoh dari bentuk subjektif adalah ujian esei. Ujian secara tertulis yang biasanya diadakan oleh lembaga pendidikan merupakan suatu sarana untuk melakukan pengambilan nilai aspek kognitif. Seorang pengajar

ketika akan membuat soal ujian tentu akan mempertimbangkan tujuan yang ingin dicapai dari pertanyaan tersebut serta bobot dari soal tersebut.

Terdapat kelebihan dari sistem pilihan ganda yaitu memudahkan para pembuat soal dan para penjawab untuk menjawab dari pertanyaan walaupun diketahui bahwa, terdapat tingkat dari soal pilihan ganda dari C1 – C6. C1 - C6 merupakan 6 cakupan penilaian secara teoritis (kognitif) dilihat dari tujuan pembuatan soal yang mendasari perkiraan bobot soal yang mengacu kepada teori taksonomi bloom. Nilai C1 adalah tingkat terendah dan C6 adalah tingkat tertinggi dalam penilaian.[3] Namun pada soal-soal pilihan ganda juga mendorong siswa untuk menebak jawaban tanpa berpikir terlebih dahulu serta lebih membuka peluang terjadinya ketidakjujuran.

Sejak tahun 1990-an hingga saat ini Indonesia terlibat dalam tes internasional yang diikuti siswa dari negara-negara maju dan negara berkembang, yakni *Programme for International Student Assesment* (PISA) di bidang membaca, Matematika, dan *Sains* untuk siswa SMP (Sekolah Menengah Pertama). Indonesia juga mengikuti *Progress in International Reading Literacy Study* (PIRLS) bidang membaca untuk siswa SD (Sekolah Dasar) serta *Trends in International Mathematics and Science Study* (TIMSS) bidang Matematika dan *Sains* untuk siswa SMP. Hasil tes menunjukkan, kemampuan siswa Indonesia di bawah standar internasional.[4] Kemampuan rata-rata siswa Indonesia dalam merespons item format uraian lebih rendah dibandingkan pilihan ganda. Kondisi itu secara umum menunjukkan siswa Indonesia lemah untuk melakukan analisis, prediksi, dan membuat kesimpulan. Hal ini dikemukakan sejumlah peneliti dari beberapa perguruan tinggi berdasarkan hasil-hasil tes internasional yang diikuti siswa Indonesia dalam seminar bertema mutu pendidikan dasar dan menengah. Penelitian dilakukan berkolaborasi dengan Badan Penelitian dan Pengembangan Depdiknas di Jakarta, Oktober 2009.[4]

Jenis ujian lainnya, yaitu bentuk esei. Ujian ini bertujuan agar dalam pengujiannya dapat melihat kemampuan yang sebenarnya dari pelajar. Melalui ujian esei pelajar tidak hanya berlatih daya pikir, namun dapat juga berlatih cara mengolah kata, mengembangkan kreativitas dalam menjawab soal, mengembangkan cara menganalisis, dan sebagainya.

Terdapat pula kekurangan dalam sistem ujian esei yang terletak pada penilaian ujian esei (*essay grading*) tersebut. Penilaian terhadap ujian esei dirasakan lebih banyak bersifat subjektivitas dari seorang guru/dosen. Untuk menanggulangi subjektifitas tersebut, maka perlu dibuat bobot nilai terlebih dahulu sebelum soal dijawab oleh pelajar. Dengan tinjauan data diatas, untuk meningkatkan kualitas dari pelajar Indonesia perlu dilakukan perubahan dalam cara menguji kualitas dari para pelajar.

2.3 SIMPLE-O

Simple-O merupakan sistem penilaian esei otomatis dengan menggunakan metode *Latent Semantic Analysis* yang dilengkapi dengan pembobotan kata dan persamaan kata untuk meningkatkan ketelitian penilaian esei. Pada Simple-O ini digunakan untuk menilai jawaban ujian esei dalam bahasa Indonesia. Ujian dilakukan secara *on-line* berbasiskan *web*. Simple-O terdiri dari 3 (tiga) modul, yaitu:

a. Modul *login*,
b. Modul dosen,
c. Modul Mahasiswa.

2.3.1 Modul *login* atau Menu Utama

Modul *login* atau menu utama digunakan untuk membedakan yang *login* mahasiswa, dosen atau admin. Bila *login* dengan *login* ID dosen mendapat fasilitas sebagai dosen, begitu juga bila *login* dengan *login* ID mahasiswa. Selain itu, terdapat juga ID *root* yang berfungsi sebagai super *user* dari sistem ini. *Root* memiliki kemampuan untuk mengasign pengguna baru sebagai dosen atau mahasiswa dan menentukan mata kuliahnya. Adapun algoritma tampilan untuk modul *login*/menu utama ditunjukkan pada Gambar 2.2.

```
prog();
    [Pengecekan idetifikasi pengguna]

    if id login = 1 then                    {login dosen}
    (
        [tampilan utama untuk dosen]
        if userName = root then
        (
                [tampilkan menu tambahan untuk root]
        }
        read (pil);

    [ke sub modul sesuai pilihan : listing nilai, tampilkan dan
  mengisi soal, registrasi atau logout]
     )

    if id login = 2 then                        {login mahasiswa}
    (
        [tampilan utama untuk mahasiswa]
        read (pil);
        [ke sub modul sesuai pilihan : listing nilai, tampilkan dan menjawab soal,
    registrasi atau logout]
    )
Eprog
```

Gambar 2.2 Algoritma Tampilan Utama dan Pilihan pada Menu Utama

(Sumber: Anak Agung Putri Ratna, 2007)

2.3.2 Modul Dosen

Modul untuk dosen terdiri dari 3 (tiga) modul :

a. Modul List Nilai, yang merupakan modul untuk melihat nilai dari mahasiswa yang mengambil ujian untuk matakuliah tersebut,
b. Modul Soal, yang merupakan modul untuk memasukkan soal yang baru, mengedit dan menghapus soal yang lama,
c. Modul Mata Kuliah, yang merupakan modul untuk melihat matakuliah yang ada pada sistem,

2.3.2.1 Modul List Nilai

Pada modul List Nilai ini dosen dapat melihat nilai dari mahasiswa yang mengambil ujian pada matakuliah yang dikelola oleh dosen yang bersangkutan. Adapun algoritma untuk modul list nilai terlihat pada Gambar 2.3.

```
Proc listnilai ()
        [load nilai_mhs untuk matkul dari basis data]
        i=0;
        while(score =! EOF)
                i++;
                write ("idmk");
                write ("nama_mhs");
                write ("score");
        ewhile
Eproc
```

Gambar 2.3. Algoritma melihat listing nilai

(Sumber: Anak Agung Putri Ratna, 2007)

2.3.2.2 Modul Soal

Pada modul soal dapat dilakukan beberapa hal seperti uraian di bawah ini.

a. Mengedit soal
b. Menghapus soal
c. Men-input soal
d. Memilih kata bobot

Pada modul soal dapat dilakukan mengedit, menghapus dan input soal oleh dosen yang besangkutan. Algoritma Global untuk modul soal dapat ditunjukkan pada Gambar 2.4.

```
Proc soal ()

        [load mata kuliah untuk user dari basis data, dapat di
        delete, edit dan input soal mata kuliah tersebut]
        read (pil);

        if pil = Delete then        {bagian untuk menghapus soal}
                [hapus record dari basis data]
        Else

        if pil = Edit then{bagian untuk mengedit soal}
                [mengedit soal]

        if pil = Input Soal then   {bagian untuk menambah soal}

                [mengisi soal]
Eproc
```

Gambar 2.4. Algoritma Global untuk modul soal

(Sumber: Anak Agung Putri Ratna, 2007)

Pada modul soal dapat dibuat beberapa fitur untuk meningkatkan kinerja dari metoda *Latent Semantic Analysis.* Fitur tersebut diuraikan di bawah ini.

a. Fitur penambahan bobot dari *keyword*.

Keyword akan dipilih 2 kali oleh dosen yang bersangkutan. Untuk ujicoba tahapan pertama adalah *keyword* yang biasa. Untuk satu jawaban, *keyword* biasa dapat terdiri dari 10 – 20 *keyword* untuk percobaan pertama. Untuk percobaan kedua, *keyword* biasa ditentukan oleh minimal 3 dosen yang kompeten. Yang kedua adalah *keyword* bobot, yang mencakup hal yang dianggap penting sekali. *Keyword* bobot terdiri dari 5 – 8 kata *keyword* perjawaban untuk percobaan pertama. Untuk percobaan kedua, *keyword* biasa juga ditentukan oleh minimal 3 dosen yang kompeten. Untuk *keyword* yang biasa pembobotan matriks adalah 1. Sedangkan untuk *keyword* bobot pembobotannya adalah dikalikan 2. Cuplikan algoritma untuk fitur penambahan bobot dari *keyword* ditunjukkan pada Gambar 2.5.

```
if pil = Input Soal then
                {bagian untuk menambah soal}
    [input soal]
    [input jawaban]
    [input kata kunci]
    [input kata kunci bobot]
    [simpan soal dan kata kunci ke basis data]
```

Gambar 2.5. Cuplikan Algoritma input soal

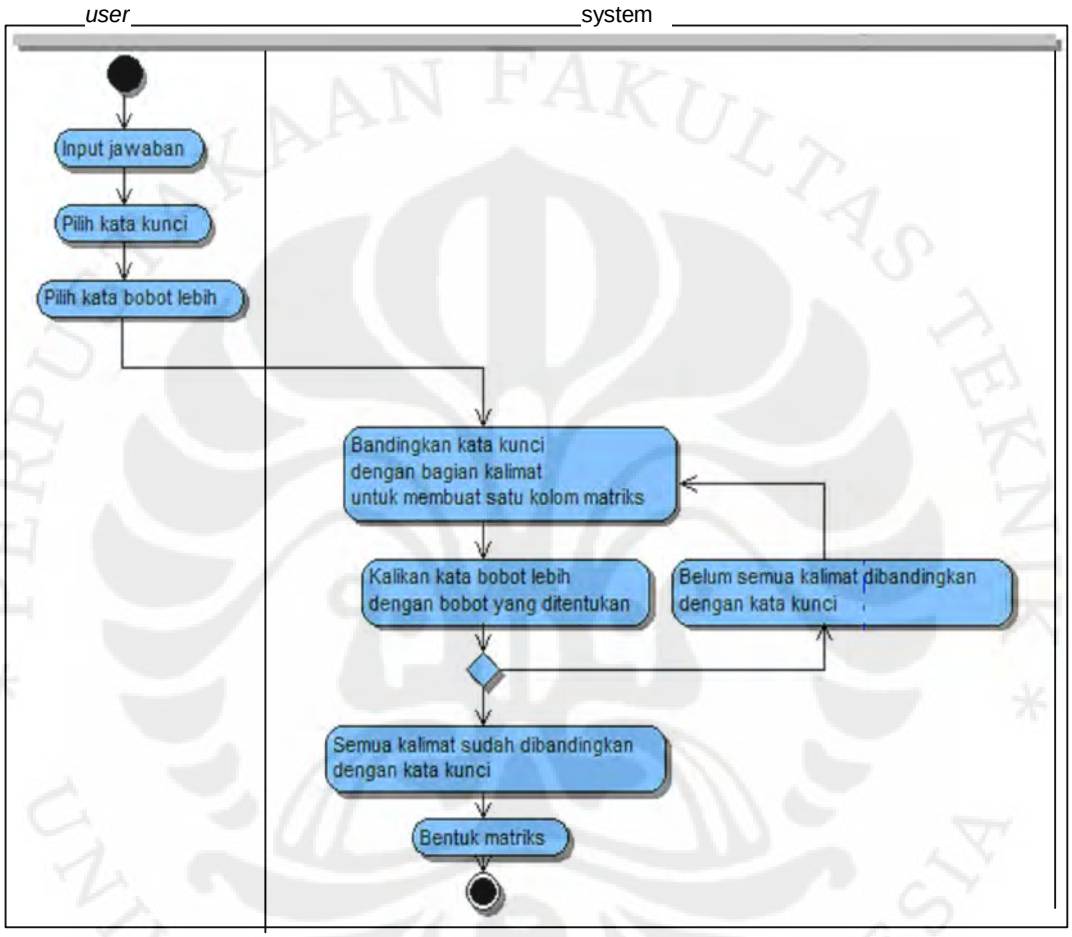
(Sumber: Anak Agung Putri Ratna, 2007)

```
if pil = Pilih Kata Bobot then
                {bagian untuk menambah soal}
    [pilih matkul]
    [pilih soal]
    [input kata kunci bobot]
    [bentuk matriks]
    [Proses SVD]
    [Simpan nilai frobenius yang sesuai]
```

Gambar 2.6. Cuplikan Algoritma Pilih Kata Bobot

(Sumber: Anak Agung Putri Ratna, 2007)

b. Pada fitur persamaan kata, kata yang sama atau yang memiliki arti yang sama akan dianggap sama. Persamaan ini berdasarkan tabel yang dibentuk seperti contoh pada Tabel 2.1. *Activity diagram* konversi matriks dan pembobotan ditunjukkan pada Gambar 2.7.



Gambar 2.7. *Activity diagram* konversi matriks dan pembobotan

(Sumber: Anak Agung Putri Ratna, 2007)

Tabel 2.1: Tabel Persamaan Kata

no	kata	kata dasar	kode kata	kode persamaan
1	memiliki	miliki	1	1
2	mempunyai	punya	2	1
3	berkesempatan	sempat	0	2

(Sumber: Anak Agung Putri Ratna, 2007)

Tabel 2.1: Tabel Persamaan Kata (Sambungan)

4	bisa	bisa	9	2
5	dapat	dapat	9	2
6	kemungkinan	mungkin	0	2
7	mampu	mampu	0	2
8	mungkin	mungkin	0	2
9	butuhkan	butuh	3	3
10	dibutuhkan	butuh	3	3
11	diperlukan	perlu	3	3
12	perlukan	perlu	3	3
13	butuh	butuh	2	4
14	membutuhkan	butuh	2	4
15	memerlukan	perlu	2	4
16	perlu	perlu	2	4
17	kebutuhan	butuh	1	5
18	keperluan	perlu	1	5
19	dimiliki	miliki	2	6
20	dipunyai	punya	3	6
21	menambah	tambah	2	7
22	menambahkan	tambah	2	7
23	menjumlah	jumlah	2	7
24	menjumlahkan	jumlah	2	7
25	modem	modem	0	8
26	connector	connect	0	9
27	konektor	konektor	0	9
28	penyambung	sambung	0	9
29	dijumlah	jumlah	3	10
30	dijumlahkan	jumlah	3	10
31	ditambah	tambah	3	10
32	ditambahkan	tambah	3	10
33	ditingkatkan	tingkat	3	10
34	penambahan	tambah	1	11
35	penjumlahan	jumlah	1	11
36	penambah	tambah	1	12
37	penjumlah	jumlah	0	12
38	standar	standar	1	13
39	standard	standard	0	13

(Sumber: Anak Agung Putri Ratna, 2007)

Tabel 2.1: Tabel Persamaan Kata (Sambungan)

40	standart	standar	1	13
41	kirim	kirim	0	14
42	penghantaran	hantar	0	14
43	pengirim	kirim	0	14
44	pengiriman	kirim	1	14
45	pengirimannya	kirim	0	14
46	pentransferan	transfer	1	14
47	sender	send	0	14
48	sending	send	0	14
49	transfer	transfer	0	14

(Sumber: Anak Agung Putri Ratna, 2007)

Kolom pertama pada Tabel 2.1, adalah kolom nomor yang menunjukkan urutan kata. Kolom kedua adalah kolom kata yang kemungkinan digunakan pada ujian oleh siswa. Kolom ini terdiri dari semua kata yang kemungkinan digunakan oleh siswa, baik dalam bentuk kata kerja transitif, kata kerja intransitif, benda, sifat dan lain sebagainya,

Kolom ketiga adalah kolom kata dasar. Semua kata yang muncul pada kolom kedua dikembalikan lagi dalam bentuk kata dasarnya. Kolom keempat adalah kolom kode kata. Pada kolom keempat ini semua kata dikategorikan pada daftar di bawah ini.

a. Kata benda : 1.
b. Kata kerja aktif : 2.
c. Kata kerja pasif : 3.
d. Kata sifat : 4.
e. Kata keterangan : 5.
f. Benda satuan : 6.
g. Kata majemuk : 7.
h. Ajektif (kata keterangan) : 8.
i. Adverb (kata sifat) : 9.
j. Kata sambung : 10.

Untuk sistem yang saat ini dikembangkan, kolom ke 4 tidak digunakan.

Kolom kelima adalah kolom untuk menyatakan kode persamaan kata. Proses yang dilakukan untuk persamaan kata adalah sebagai berikut. Pertama-tama kalimat yang dimasukkan oleh siswa sebagai jawaban diuraikan menjadi kata-kata. Kata-kata tersebut kemudian disesuaikan dengan kata kunci dari jawaban referensi dosen yang telah dimasukkan sebelumnya ke sistem. Kata-kata tersebut kemudian dicek lagi ke basis data persamaan kata. Bila ada kata yang kode persamaannya sama, maka kata tersebut diproses sama dengan kata kunci yang ada pada jawaban referensi dosen. Pembobotan diberikan apakah kata tersebut sesuai kata kunci atau kata bobot, Sebagai contoh terlihat pada Tabel 2.1, untuk no. 22 dan no. 23, yaitu ’menambahkan’ dan ’menjumlah’. Bila siswa menjawab dengan kata ’menambahkan’ dan jawaban referensi adalah ’menjumlah’ maka siswa tersebut mendapatkan nilai penuh untuk kata tersebut. Algoritma untuk modul pengecekan persamaan kata dapat ditunjukkan pada Gambar 2.8.

```
Proc ujian ()
        [load matkul untuk user dari basis data]
        while matkul =! EOF
                [tampilkan mata kuliah yang dipilih]
        ewhile
        read (pil);
        if pil = Back then
                goto Halaman_Muka;
        else

        if pil = Lihat Soal then
                write (“Soal”, soal[ ]);
                write (“Jawaban”);
                read jawab_mhs[ ];
                if submit = true then
                        [cek persamaan kata-kata jawaban mahasiswa]
                        [bentuk matriks dari jawaban mahasiswa];
                        [bandingkan matriks jawaban dengan
                referensi];
                        [kirim nilai ke tabel basis data];
                else
                ( );
        else
        ( );
Eproc
```

Gambar 2.8. Algoritma pengecekan persamaan kata

(Sumber: Anak Agung Putri Ratna, 2007)

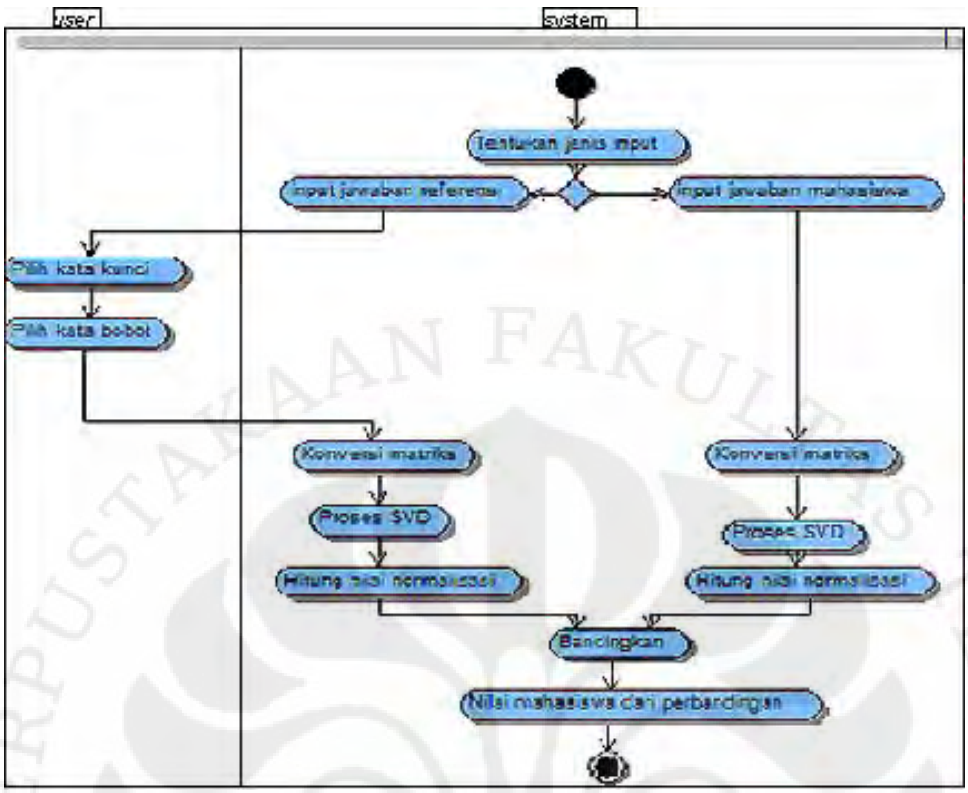
2.3.3 Modul mahasiswa

Modul mahasiswa terdiri dari 2 modul, yaitu modul lihat nilai dan modul ujian. Pada modul lihat nilai, mahasiswa dapat melihat nilai ujiannya. Mahasiswa hanya dapat mengikuti ujian hanya satu kali. Algoritma untuk mengikuti ujian dengan metoda LSA yang dikembangkan dapat ditunjukkan pada Gambar 2.9.

```
Proc ujian ()
        [load matkul untuk user dari basis data]
        i = 0;
        write (“Mata Kuliah”);
        while matkul =! EOF
                i++;
                write ["matkul"];
        write (“Lihat Soal”);
                write (“Back”);
        read (pil);
        if pil = Back then
                goto Halaman_Muka;
        else
        if pil = Lihat Soal then
                write (“Soal”, soal[ ]);
                write (“Jawaban”);
                read jawab_mhs[ ];
                if submit = true then
                        [cek persamaan kata-kata jawaban mahasiswa]
                        [bentuk matriks dari jawaban mahasiswa dengan
                pembobotan pada kata kunci bobot];
                        [bandingkan matriks jawaban dengan referensi,
        dengan membandingkan nilai normalisasi
        frobenius];
                        [kirim nilai ke tabel basis data];
                else
                ( );
        else
        ( );
Eproc
```

Gambar 2.9. Algoritma mengikuti ujian dengan metoda LSA yang dikembangkan

(Sumber: Anak Agung Putri Ratna, 2007)



Gambar 2.10. *Activity diagram* mengikuti ujian dengan metoda LSA yang dikembangkan (Sumber: Anak Agung Putri Ratna, 2007)

2.4 *Essay Grading* dengan metode LSA (*Latent Semantic Analysis*)

Dalam melakukan penilaian terhadap ujian esei diperlukan sebuah teknik dalam melakukan penilaian. *Essay grading* merupakan suatu teknik penilaian dengan tujuan membangun suatu sistem yang mampu memberikan penilaian terhadap jawaban esei secara otomatis secara komputerisasi. Terdapat beberapa metode dalam melakukan penilaian ujian esei, salah satu diantaranya adalah LSA (*Latent Semantic Analysis*). *Latent Semantic Analysis* (LSA) merupakan representasi terhadap jumlah dan kemungkinan kata untuk dibandingkan secara geometris (matrik). LSA merepresentasikan isi kata dalam matrik dua dimensi yang besar. Terdapat sebuah bagian yang penting dari pemrosesan dari LSA adalah komponen analisis bernama SVD (*Singular Value Decomposition*) yang mengkompresi informasi yang berkaitan dalam jumlah besar ke dalam ruang yang lebih kecil tetapi mewakili arti sebenarnya.

Berdasarkan data yang didapat dari Landauer, bahwa LSA telah diujikan dengan 5 skema penilaian, masing-masing dengan perlakuan berbeda dimana esei mahasiswa dibandingkan dengan esei referensi. Hal ini berkaitan dengan vektor yang harus dikomputasi. Dari hasil ujicoba yang telah dilakukan, pada kelas kecil diperoleh nilai kesesuaian dengan human raters berkisar 69.80 % – 94.64 %, sedangkan pada kelas menengah diperoleh nilai berkisar 77.18 % – 98.42 %. [5] Hasil-hasil ini setara dengan hasil metoda grading otomatis LSA terdahulu, yang melakukan penilaian terhadap jawaban ujian dalam bahasa Inggris.

2.5 Singular Value Decomposition (SVD)

Teknik SVD digunakan untuk melakukan perkiraan struktur penggunaan kata dalam dokumen-dokumen. SVD merupakan teknik untuk melakukan estimasi *rank* dari matriks. Jika diketahui matriks A dengan dimensi m x n, dimana nilai m > n dan rank (A) = r maka *singular value decomposition* dari A, dinotasikan sebagai SVD(A), didefinisikan melalui persamaaan: [6]

$$A = U \sum V^T \quad ...(2\text{-}1)$$

dimana,

$$U^T U = V^T V = I_n \quad ...(2\text{-}2)$$

dan memenuhi kondisi,

$$\sum = \text{diag}\,(\sigma_1, ..., \sigma_n) \quad ...(2\text{-}3)$$

dimana,

$\sigma_i > 0$ untuk $1 \leq i \leq r$

$\sigma_j = 0$ untuk $j \geq r + 1$

Kolom r pertama dari matriks U dan V mendefinisikan vektor eigen orthonormal yang bersesuaian dengan r nilai vektor eigen tidak-nol dari matriks AA^T dan A^TA berturut-turut. Kolom dari matriks U dan V berisi vektor, masing-masing-masing disebut vektor singular kiri dan kanan.

Nilai singular dari *A* merupakan elemen diagonal dari matriks $\sum$, dimana nilai singular didapat dari akar pangkat dua dari nilai absolut dari sejumlah *n* nilai eigen dari AA^T.

2.6 Program Pendukung

Dalam pembuatan program essay grading berbasis web ini, menggunakan sistem yang dibangun dengan bahasa *scripting* PHP yang menggandeng *basis data MySQL* dan *scripting HTML*. Pada proses perhitungan matriks di dalam program *essay grading* ini, menggunakan program JAMA (Java Matrix) yang di program ulang menjadi program PHP. *Web server* yang digunakan adalah Apache.

2.6.1 PHP (Hypertext Preprocessor)

PHP banyak digunakan oleh programmer dengan alasan memiliki kemiripan syntax dengan bahasa C/C++, Java, dan Perl. PHP merupakan bahasa *scripting open source* dengan menggunakan lisensi GPL (GNU Public Licence) yang dapat digunakan dengan gratis dan bebas. PHP biasa digunakan untuk membuat aplikasi *web* yang bersifat dinamis.

PHP menyerupai dengan HTML dimana kode-kode yang dibuat tidak perlu di-*compile* sebelum digunakan. Kode yang dibuat akan diproses saat diperlukan.

2.6.2 MySQL

MySQL merupakan *software* yang tergolong sebagai DBMS *(Data Base Management System)* yang bersifat *Open Source*. MySQL awalnya dibuat oleh perusahaan konsultan bernama TcX yang berlokasi di Swedia. Selanjutnya pengembangan MySQL berada dibawah naungan perusahaan MySQL AB. [7]

Pada tanggal 16 Januari 2008 Sun Microsystems, Inc mengumumkan aksi korporasi - akuisisi terhadap MySQL AB sehingga menjadikan Sun sebagai salah satu perusahaan dengan produk *platform open source* terbesar seperti Java, OpenSolaris dan akhirnya MySQL.

Berselang setahun kemudian, tepatnya pada tanggal 20 April 2009 giliran Oracle melakukan akusisi terhadap Sun Microsystems.

Sebagai *software* DBMS, MySQL memiliki beberapa fitur, diantaranya :

a. Multiplatform

   MySQL dapat digunakan dalam banyak *paltform*, seperti *Windows, Linux, Unix* dan *platform* lainnya.

b. *Multiuser*

   MySQL dapat digunakan oleh beberapa pengguna dalam waktu yang bersamaan tanpa mengalami masalah atau konflik.

c. Perintah dan Fungsi

   MySQL memiliki operator dan fungsi secara penuh yang mendukung semua perintah standar SQL

d. Antar Muka

   MySQL memiliki interface (antar muka) terhadap berbagai aplikasi dan bahasa pemrograman dengan menggunakan fungsi API (Application Programming Interface).

e. Andal, cepat dan mudah digunakan

   MySQL tergolong sebagai *database server* yang andal, dapat menangani basis data yang besar dengan kecepatan tinggi, mendukung banyak fungsi untuk mengakses basis data, dan sekaligus mudah untuk digunakan.

f. Jaminan keamaanan akses

   MySQL mendukung pengamanan basis data dengan berbagai kriteria pengaksesan. Contohnya, pada MySQL dapat dimungkinkan untuk mengatur pengguna tertentu agar bisa mengakses data yang bersifat pribadi dimana *user* lainnya tidak dapat mengakses data tersebut.

g. Dukungan SQL

   MySQL mendukung perintah SQL *(Structured Query Language)* yang merupakan bahasa yang yang digunakan untuk mengakses basis data relasional.

2.6.3 Apache

Server HTTP Apache atau *Server Web*/WWW Apache adalah *web server* yang dapat dijalankan di banyak sistem operasi seperti *Unix*, *BSD*, *Linux*, *Microsoft Windows* dan *Novell Netware* serta *platform* lainnya, yang berguna untuk melayani dan memfungsikan situs web. Protokol yang digunakan untuk melayani fasilitas *web* ini menggunakan HTTP. *Web Server* Apache berbasiskan *open source* dan mulai populer di *internet* semenjak tahun 1996.

Supaya dokumen-dokumen *web* berupa HTML ataupun PHP bisa diakses oleh *browser* maka dokumen-dokumen tersebut perlu diletakkan dalam direktori khusus yang diatur oleh Apache.

2.6.4 JAMA (Java Matrix)

Pada awalnya dalam melakukan perhitungan SVD menggunakan program matlab, sehingga hasil perhitungan dapat dihitung dalam program matlab tersebut. Namun untuk mendapatkan hasil dari perhitungan tersebut membutuhkan waktu yang cukup lama untuk satu orang pengguna saja, sehingga dibutuhkan suatu program yang memungkinkan mempercepat proses perhitungan dengan meninjau banyaknya *pengguna* saat melakukan ujian bersama-sama.

Dengan alasan tersebut, maka dipilih Java Matrix yang dapat mengerjakan proses perhitungan SVD lebih cepat dibandingkan dengan Matlab. JAMA yang digunakan merupakan skrip PHP untuk perhitungan matriks kompleks. Class dari package JAMA akan sering digunakan dalam operasi matriks seperti perkalian matriks, transpose, dan inverse.

BAB 3
PERANCANGAN KEAMANAN SIMPLE-O

3.1 Analisis Keamanan Simple-O

Keamanan yang ada saat ini pada aplikasi Simple-O, hanya terdapat *cookie* yang berisi identitas pengguna dengan waktu 1 jam. Dapat diartikan bahwa jika pengguna lupa keluar dari program maka data *cookie* yang ada di komputer *client* akan terhapus dalam waktu 1 jam. Namun setelah 1 jam tersebut, aplikasi Simple-O tidak keluar secara otomatis dari program. Pada bagian basis data, sudah diamankan dengan menggunakan kata sandi untuk membuka basis data dari aplikasi Simple-O. Hal ini mengakibatkan tidak mudahnya untuk membuka basis data *server* karena harus melewati proses autentifikasi terlebih dahulu. Disamping itu, pengamanan basis data juga dilakukan pada pengaturan akses, dimana untuk membuka basis data *server* tidak bisa dilakukan dari luar jaringan. Hal ini dilakukan untuk mengantisipasi penyerangan yang dilakukan dari luar jaringan terhadap basis data *server*.

Pada aplikasi Simple-O diperlukan beberapa keamanan untuk menjaga program dari serangan orang-orang luar yang tidak bertanggung jawab. Jenis serangan dapat bermacam-macam, mulai dari ingin mengetahui data-data di dalam program, ingin mengambil data-data di dalam program, sampai pada ingin merusak sebuah program. Untuk menghindari hal tersebut, maka dilakukan pendekatan keamanan pada aplikasi Simple-O yang cukup aman, namun tidak menurunkan performa dari aplikasi Simple-O. Dengan membatasi pengamanan yang dilakukan dari segi skrip PHP aplikasi Simple-O, maka dapat diterapkan beberapa sistem keamanan diantaranya, adalah:

a. *Session* dengan Pewaktuan;
b. *Password strength* meter;
c. *Hash* kata sandi dengan MD5;
d. Lupa kata sandi dan ubah kata sandi;
e. Proses *login* dengan sistem terkunci;
f. CAPTCHA pada proses *login*.

3.2 *Session* dengan Pewaktuan

Session/sesi merupakan suatu mekanisme untuk menyimpan suatu data tertentu seperti nama pengguna atau kata sandi pada saat mengakses suatu *website*. Data *session* tersimpan pada *web server*, yang berfungsi untuk membentuk interaksi antara *client* dan *web server*. Pada prinsipnya *session* dan *cookie* mekanisme kerjanya hampir sama, tetapi berbeda tempat penyimpanan, jika *session* menyimpan data pada *server* sedangkan *cookie* menyimpan data pada *client*.[8]

Autentifikasi sesi pengguna dapat dibuat untuk menahan *state information* untuk pengguna di antara perintah-perintah, ketika ada serangkain perintah yang datang dari pengguna yang sama. Menangani *state* di sebuah aplikasi dapat menjadi sebuah tantangan, terutama disebabkan oleh jumlah data yang mungkin terakumulasi. PHP tidak menawarkan ketahanan data selama suatu perintah dijalankan sehingga data tersebut harus disimpan di suatu tempat dimana data ini bisa diakses setelah semua perintah terpenuhi. Beberapa teknik manajemen sesi antara lain dibagi menjadi 2 hal yaitu:

a. *Client side session*

   Sesi sisi klien menggunakan *cookie* dan teknik kriptografi untuk mempertahankan id sebagai *state* tanpa menyimpan banyak data di *server*. Ketika menyajikan halaman *web* yang dinamis, *server* akan mengirimkan data kondisi terkini kepada *client* (*web browser*) dalam bentuk *cookie*. *Client* akan menyimpan *cookie* di memori atau pada *disk*. Berturut-turut dengan masing-masing permintaan, *client* mengirimkan kembali *cookie* ke *server*, dan *server* menggunakan data untuk "mengingat" keadaan aplikasi untuk *client* tertentu dan menghasilkan tanggapan yang tepat. Mekanisme ini dapat bekerja dengan baik dalam beberapa konteks, namun data yang tersimpan pada *client* adalah rentan terhadap gangguan oleh pengguna atau oleh perangkat lunak yang memiliki akses ke komputer *client*. Hal-hal yang perlu diperhatikan dari *client side session* antara lain :

   - Ada potensi untuk adanya gangguan
   - *Client side session* sulit untuk transportasi
   - Mempunyai potensi kehilangan

b. Server side session

Teknik ini dibangun di atas sisi *server* perangkat penyimpanan, yang akan menyimpan data untuk sesi tertentu. Setiap sesi memiliki sebuah identitas unik yang berhubungan dengan data di *server*. Identitas ini juga berhubungan dengan agen pengguna baik dalam bentuk *cookie*, sebuah parameter *query_string*, *field* yang tersembunyi atau semua pada waktu yang sama. Adapun keuntungan dari *server side session* antara lain adalah:

- Data sensitif seperti *username* pengguna, alamat *email*, preferensi dan seperti itu tidak perlu lagi melakukan perjalanan melalui jaringan pada setiap permintaan (yang merupakan kasus dengan *string query*, *cookies* dan *hidden_fields*). Satu-satunya yang bergerak di seluruh jaringan adalah identitas unik yang dihasilkan untuk sesi ("ID-1234", misalnya), yang seharusnya tidak disadari oleh pihak lain.
- Pengguna tidak akan memiliki data sensitif yang disimpan dalam komputer dalam format teks biasa tanpa jaminan keamanan(yang dimaksud adalah *file cookie*).
- Memungkinkan untuk menangani struktur data yang sangat besar dan kompleks (di memori) secara transparan.

Kelemahan pada sesi sisi *server* seperti:

- Jika terjadi banyak permintaan dari pengguna, maka kinerja dari *server* dalam memenuhi suatu permintaan menjadi tidak optimal. Hal ini dapat mengakibatkan tidak seimbangnya pemenuhan jumlah permintaan.
- Dengan menggunakan satu *server* pada suatu sistem akan dapat mengakibatkan rentannya kehilangan data dalam sistem tersebut, sehingga diperlukan backup data dari *server* tersebut (*server* cadangan). Hal ini dilakukan guna menghindari dampak terburuk terjadinya kerusakan pada *server* utama.

Pada aplikasi Simple-O saat ini, masih belum terdapat *session* didalamnya namun sudah terdapat *cookie* di dalamnya, sehingga diperlukan penggunaan *session* yang tepat agar tidak terjadi celah keamanan dalam aplikasi Simple-O ini.

Dengan tidak adanya *session* yang mengatur pada program ini, seseorang dapat masuk ke bagian penilaian, dosen, ataupun lainnya tanpa harus melakukan proses *login*. Pada aplikasi Simple-O ini akan ditambahkan penggunaan *server side session* dengan pertimbangan bahwa pemakaian komputer bersama oleh mahasiswa yang bisa menyebabkan terjadinya pencurian data *session* di komputer, jika kita hanya menggunakan *client side session.*

Untuk lebih meningkatkan keamanan dari segi sesi dalam aplikasi Simple-O, maka sesi akan menggunakan pewaktuan. Pewaktuan yang dimaksud guna untuk menghindari pengguna lupa *logout* dari aplikasi Simple-O sehingga dapat mengamankan identitas pengguna. Algoritma pewaktuan dapat dilihat pada Gambar 3.1. Pada *cookie* dari program ini juga diberikan waktu, sehingga *cookie* dapat terhapus secara otomatis jika user lupa *logout* dari program.

```
Proc Pewaktuan ()
[memulai sesi]

function login_validate() { //fungsi validasi
   $timeout = 3600;   //waktu selama 1 jam
   $_SESSION["expires_by"] = time() + $timeout;
}
Efunc
function login_check() {  // fungsi cek login
   $exp_time = $_SESSION["expires_by"];
   if (time() < $exp_time)
         {
         login_validate();
         return true;
         }
   else
         {
         unset($_SESSION["expires_by"]);
         return false;
         }
}
Efunc
Eproc
```

Gambar 3.1 Algoritma pewaktuan [9]

3.3 *Password Strength Meter*

Pada aplikasi Simple-O sudah terdapat sistem autentifikasi, namun tidak menutup kemungkinan terjadinya kebocoran keamanan yang berasal dari proses autentifikasi tersebut. Pada aplikasi Simple-O ini autentifikasi berupa permintaan id pengguna dan kata sandi dari pengguna. Dalam pemilihan kata sandi pada umumnya terbagi menjadi dua hal yang utama yaitu kata sandi yang terpilih secara acak untuk pengguna dan kata sandi yang khusus dibuat sendiri oleh

pengguna**.** Pada aplikasi Simple-O ini, menggunakan sistem kata sandi yang dibuat sendiri oleh pengguna dengan tujuan memudahkan pengguna untuk mengingat kata sandi yang dibuatnya. Namun, ketika kata sandi tersebut dibuat sendiri oleh pengguna, pada umumnya bersifat kurang aman bagi pengguna. Pengguna cenderung membuat kata sandi yang mudah diingat sehingga kombinasinya mudah ditebak oleh *hacker*.

Serangan yang dapat terjadi pada sistem autentifikasi adalah *dictionary attack*. *Dictionary attack* merupakan serangan otomatis yang digunakan untuk melawan sistem otentikasi. *Hacker* pada umumnya menggunakan kata sandi yang sangat potensial dan mencoba untuk masuk ke akun pengguna yang telah diberikan. Serangan jenis ini tidak dapat dilakukan pada kata sandi yang diberikan secara *random/*acak kepada pengguna tetapi berfungsi untuk menyerang kata sandi yang dibuat oleh pengguna sendiri.

Solusi dari permasalahan ini bisa dilakukan dengan memberitahukan kekuatan kata sandi dengan *Password Strength* Meter. Permasalahan yang umum adalah dititikberatkan pada kata sandi yang "bagus", yang dimiliki pengguna dengan menggunakan aturan yang sederhana. Aplikasi Simple-O ini memperbolehkan pengguna untuk menentukan kata sandinya sendiri tetapi kata sandi tersebut harus melalui suatu tes apakah memenuhi tingkat kualitas kata sandi yang "bagus". Pengertian kata "bagus" disini dapat diartikan sebagai kombinasi kata sandi seperti, kombinasi huruf besar dan kecil, angka, dan batas minimal karakter untuk kata sandi. Algoritma dari *password strength meter* dapat dilihat pada Gambar 3.2. Dengan bantuan dari program *password strength meter*, diharapkan pengguna dapat membuat kata sandi yang "bagus".

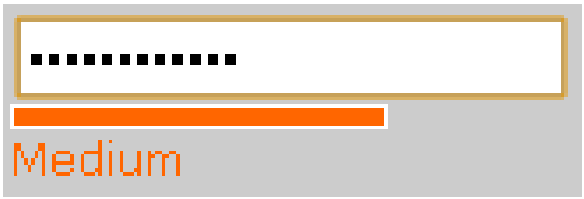
Proc Password Strength Meter ()

[masukkan karakter untuk kata sandi]

[menghitung tingkat kekuatan kata sandi dengan meninjau kombinasi dari huruf besar dan kecil, angka, tanda baca, dan karakter khusus]

[Setelah 6 karakter diisi, maka akan menampilkan parameter kekuatan password]

Gambar 3.2 Algoritma dari password strength meter [11]



Gambar 3.3 Salah satu tampilan dari password strength meter [11]

3.4 *Hash* Kata Sandi dengan MD5

Pada program awal Simple-O, kata sandi yang tersimpan dalam basis data masih berbentuk kata sandi asli. Hal ini masih sangat tidak aman jika ada seseorang sampai bisa melihat isi basis data, maka orang tersebut sudah memiliki semua akun pengguna maupun *admin* dari program. Guna menanggulangi masalah ini, maka dilakukan pengaman pada kata sandi dengan menggunakan metode *hash* MD5(Message Digest 5).

Message digest adalah suatu nilai hasil transformasi string kata sandi dengan menggunakan *class message digest* yang nantinya akan disimpan dalam basis data untuk dicocokkan ketika user melakukan autentifikasi.[10] MD5 adalah suatu algoritma yang dapat memastikan bahwa pesan yang dikirim akan sama dengan pesan yang diterima dengan membandingkan inti sari dari pesan tersebut dan bersifat satu arah. Satu arah dalam hal ini berarti tidak mempunyai fungsi untuk melakukan pengembalian nilai yang telah mengalami proses *hash*.

Pada penyimpanan kata sandi di dalam basis data, kata sandi akan tersimpan dalam bentuk kata sandi yang sudah mengalami proses *hash.* Pada proses *hash* kata sandi, perlu ditambahkan kata pengacak untuk kombinasi kata sandi. Tujuan dari kata pengacak ini guna meningkatkan kesulitan dalam memecahkan kata sandi yang telah mengalami proses *hash.* Algoritma kombinasi *hash* MD5 dengan pengacak dapat dilihat pada Gambar 3.4.

```
Proc kombinasi_hashMD5()
$pass= input kata sandi;
$pengacak = "awdrgyjilpzdcvgnjm";
$password = md5($pengacak . md5($pass) . $pengacak);
Eproc
```

Gambar 3.4 Algoritma kombinasi *hash* MD5 dengan pengacak. [12]

Penjelasan dari algoritma diatas adalah:

a. Pertama-tama, mengidentifikasi kata sandi yang dimasukkan oleh pengguna ke dalam variabel $pass.
b. Tentukan variabel pengacak yang ingin digunakan.
c. Selanjutnya, kombinasikan password asli yang sudah di *hash* MD5 dengan variabel pengacak. Setelah dikombinasikan, maka diproses *hash* MD5 kembali, yang hasilnya disimpan ke dalam basis data.

3.5 Lupa Kata Sandi dan Ubah Kata Sandi

Lupa kata sandi adalah masalah yang biasa bagi manusia. Dengan banyaknya akun yang dimiliki pengguna, memungkinkan pengguna untuk tidak memberikan kata sandi yang sama untuk setiap akun. Hal ini dikarenakan supaya menghindari kebocoran salah satu kata sandi akun, yang dapat mengakibatkan seseorang tahu kata sandi akun yang lain yang berhubungan dengan pengguna. Namun dengan cara memberikan kata sandi yang berbeda-beda untuk beberapa akun yang dimiliki, memungkinkan seorang pengguna lupa kata sandi pada salah satu akun yang dia miliki. Untuk mengatasi masalah ini, maka setiap pengembang dari sebuah *website* akan memberikan layanan lupa kata sandi / *forgot password*. Pada aplikasi Simple-O ini masih belum ada layanan lupa kata sandi, sehingga yang dapat mengetahui dan mengubah kata sandi dari pengguna, hanya yang memiliki kekuasaan penuh terhadap basis data Simple-O, yaitu *admin*. Namun akan dirasakan sedikit kesulitan jika banyak yang mengalami lupa kata sandi dalam satu waktu dan harus meminta bantuan *admin* untuk mengubah kata sandi pengguna. Maka, dari permasalahan tersebut akan dibuat layanan lupa kata sandi

pada aplikasi Simple-O untuk mengatasi masalah tersebut. Adapun algoritma untuk lupa kata sandi terlihat pada Gambar 3.5.

```
Proc reset_katasandi()
$email = input email;

function randomPassword()   //  function untuk membuat kata sandi
{
$digit = 6;
$karakter =
"ABCDEFGHJKLMNPQRSTUVWXYZabcdefhijkmnpqrstuvwxyz23456789";
srand((double)microtime()*1000000);  //random generator
$i = 0;
$pass = "";
while ($i <= $digit-1)
{
$num = rand() % 55;
$tmp = substr($karakter,$num,1);
$pass = $pass.$tmp;
$i++;
}
return $pass;
}
Efunc
// variabel katasandi baru >> memanggil function randomPassword
        $newPassword = randomPassword();
        $pengacak  = "awdrgyjilpzdcvgnjm";
        $newPasswordEnkrip = md5($pengacak . md5($newPassword) .
$pengacak);
        [cek email dalam tabel tbuser di basis data]
        [cek baris dalam basis data tbuser]
```

Gambar 3.5 Algoritma lupa kata sandi.[13]

(telah diolah kembali)

```
        [cek baris dalam basis data tbuser]
        if (baris dalam basis data = 0)
        {
                [tampilkan “email tidak valid”]
        }
        Else  // jika ada maka akan update kata sandi dalam basis data
        {
                [ update kata sandi dalam basis data berdasarkan email;]
        }
Eproc
```

Gambar 3.5 Algoritma lupa kata sandi [13] (Lanjutan)
(telah diolah kembali)

Tidak menutup kemungkinan juga bahwa, kata sandi seseorang diketahui orang lain dengan disengaja ataupun tak disengaja. Hal ini sangat tidak baik bagi seorang pengguna yang ketahuan kata sandi akunnya oleh pihak lain karena akun aplikasi Simple-O ini adalah program penilaian ujian esei dari seorang pengguna dimana data-data di dalamnya sangat penting untuk penilaian kegiatan belajar pengguna. Untuk membantu seorang pengguna yang kata sandi miliknya diketahui orang lain, maka akan ditambahkan fasilitas pada aplikasi Simple-O berupa ubah kata sandi. Pada layanan ini pengguna diharapkan dapat mengganti kata sandi secara berkala, untuk menghindari pencurian terhadap kata sandi pengguna. Algoritma untuk ubah kata sandi dapat terlihat pada Gambar 3.6.

```
Proc ubah_katasandi ()
$pass_lama = input kata sandi lama;
$pengacak = "awdrgyjilpzdcvgnjm";
$password_lama = md5($pengacak . md5($pass_lama) . $pengacak );

$pass = input kata sandi baru;
$pengacak = "awdrgyjilpzdcvgnjm";
$password_baru = md5($pengacak . md5($pass) . $pengacak );
```

Gambar 3.6 Algoritma ubah kata sandi

```
$pass2 = input kembali kata sandi baru;
$pengacak = "awdrgyjilpzdcvgnjm";
$password_baru1 = md5($pengacak . md5($pass2) . $pengacak );

$userid = $_COOKIE['ses_userid']; //userid saat login

        //koneksi ke basis data
        [select password from tbuser where userid='$userid']
if ($pass_lama=="")
        {
                [tampilkan “Password lama kosong"]
        }
        else if ($pass == '')
        {
                [tampilkan "Password baru kosong"]
        }
        else if ($pass != $pass2)
        {
                [tampilkan "Password baru tidak sama"]
        }
        else if ($password_lama != mysql_result($result, 0))
        {
           [tampilkan “Masukkan password lama dengan benar"]
        }
       else
        {
       [update kata sandi yang baru ke dalam basis data berdasarkan userid]
      [tampilkan “ubah password berhasil”]
        }
Eproc
```

Gambar 3.6 Algoritma ubah kata sandi (Lanjutan)

3.6 Proses *Login* Dengan Sistem Terkunci

Dengan melihat kondisi dari aplikasi Simple-O ini, tidak memungkinkan seseorang dapat menembus akun orang lain pada saat mengerjakan soal ujian karena pentingnya keluaran yang dikerjakan oleh pengguna. Untuk mengantisipasi terjadinya masalah tersebut, maka akan dibuat suatu proses *login* dengan sistem terkunci sehingga seorang pengguna yang masuk ke dalam aplikasi Simple-O tidak dapat ditembus orang lain sampai pada saat waktu diam yang ditentukan habis atau sampai pada saat pengguna logout dari aplikasi Simple-O. Algoritma untuk proses *login* dengan sistem terkunci seperti Gambar 3.7.

```
Proc login_terkunci()
    [cek userid, kata sandi, kode CAPTCHA]
    [cek status sign dalam basis data bernilai 0 atau 1]
    If (sign=0)
        [update basis data nilai sign menjadi 1]
    Else
        [proses login kembali]
Eproc
```

Gambar 3.7 Algoritma proses login terkunci

3.7 CAPTCHA Pada Proses *Login*

Pada perancangan CAPTCHA di Simple-O ini, akan dipasang pada bagian form *login*, untuk keamanan pada sistem autentifikasi aplikasi Simple-O. Dengan adanya program CAPTCHA ini, diharapkan terhindar dari program yang dijalankan komputer untuk dapat masuk ke dalam sistem Simple-O. Ada beberapa hal yang harus diperhatikan dalam penggunaan CAPTCHA agar fungsi dari CAPTCHA tersebut maksimal yaitu :

a. Gambar teks CAPTCHA harus terdistorsi secara acak sebelum disajikan kepada pengguna.
b. CAPTCHA mudah dibaca oleh manusia dan tidak terbaca oleh komputer, yaitu dibuat hanya dengan berdasarkan teks bacaan atau persepsi visual lainnya. Hal ini dilakukan untuk mencegah pengguna

yang tidak diinginkan mampu mengakses sumber daya yang dilindungi.

c. Tingkat keamanan dari *script* CAPTCHA itu sendiri harus bersifat aman, sehingga sistem pun harus dapat memastikan bahwa tidak ada cara yang mudah untuk mengganggu *script* CAPTCHA tersebut.
d. Penciptaan *script* CAPTCHA sendiri merupakan hal yang kurang baik karena dikhawatirkan akan terdapat banyak kemungkinan celah yang dapat ditembus oleh penyerang. Oleh sebab itu, untuk meningkatkan keamanan website lebih baik menggunakan script CAPTCHA yang telah diuji implementasinya dengan baik.

BAB 4
PENERAPAN, PENGUJIAN, DAN ANALISIS SISTEM KEAMANAN PADA SIMPLE-O

4.1 Penerapan Sistem Keamanan Pada Simple-O

Aplikasi Simple-O merupakan aplikasi yang berbasiskan *web*, sehingga dalam penerapannya dibutuhkan *web server*. Dalam sebuah *web server* dibutuhkan PHP *engine* dan sistem basis data. Aplikasi Simple-O ini akan diterapkan pada sebuah komputer yang memiliki spesifikasi sebagai berikut:

Motherboard : ECS 945GZT-M v: 1.0
Prosesor : Pentium(D) 2.80 GHz
Memory (RAM) : 2 GB
Sistem Operasi : Ubuntu 10.4
Web *Server* : Apache 2.2.14
Sistem Basis Data : MySQL 5.1.41
PHP Engine : PHP 5.3.2

Proses pengujian dilakukan dengan sistem *server-client*, dimana terdapat satu komputer *server* dan beberapa komputer *client* yang terhubung dengan menggunakan *internet*. Dengan menggunakan *web browser*, pengguna dapat menggunakan aplikasi Simple-O dengan mengetikkan alamat http://www.ee.ui.ac.id/logic/simple-o untuk mengakses aplikasi Simple-O yang sudah diterapkan sistem keamanan dan alamat http://www.ee.ui.ac.id/logic/bobotsama untuk mengakses aplikasi Simple-O sebelum penerapan sistem keamanan.

4.1.1 Analisis Penerapan Session

Session/sesi merupakan suatu mekanisme untuk menyimpan suatu data tertentu seperti nama pengguna atau kata sandi pada saat mengakses suatu *website*. Data *session* tersimpan pada *web server*, yang berfungsi untuk membentuk interaksi antara *client* dan *web server*. Pada aplikasi Simple-O ini diterapkan sesi pada saat autentifikasi dengan tujuan menyimpan identitas (*user*

id) dari pengguna untuk memasuki halaman-halaman berikutnya. Dengan adanya sesi pada aplikasi Simple-O, hal ini dapat menutup kemungkinan terjadinya pengguna yang masuk ke program tanpa melalui proses autentifikasi.

Setiap *file* html yang merupakan tampilan yang berhubungan dengan pengguna, diawali dengan *session_start()* untuk mengecek sesi dari *id* pengguna aktif atau tidak. Pada sesi di aplikasi Simple-O diberikan sebuah pewaktuan untuk keluar dari program. Hal ini dilakukan untuk mengantisipasi pengguna yang lupa keluar dari program. Pengaturan waktu dari aplikasi Simple-O ini terdapat dalam file timer.php.

Function login_validate berfungsi untuk membuat validasi dari sesi yang ada. Perhitungannya adalah waktu saat sesi itu aktif ditambahkan dengan waktu dari *timeout* sebesar 3600 detik atau 1 jam. Sedangkan, pada *function login_check* dilakukan pengecekan pada waktu yang telah berlangsung, jika waktu sekarang lebih kecil dibandingkan dengan waktu habis/*expired* maka akan kembali ke *function login_validate* dimana waktu yang ada sekarang ditambahkan lagi dengan waktu *timeout* sebesar 3600 detik. Jika nilai dari waktu sekarang lebih besar dari waktu habis, maka akan keluar dari program.

Dengan menggunakan pewaktuan tersebut, maka pada bagian awal skrip dari setiap file yang akan ditampilkan kepada pengguna, diawali dengan algoritma pada Gambar 4.1 untuk menglihat sesi yang aktif.

```
Proc cek_sesi()
if (ISSET($_SESSION['username']))
        {
                if (!login_check()) //func dari timer.php
                        {
                         header("Location: logout.php");
                         exit(0);
                         }
        }
Eproc
```

Gambar 4.1 Algoritma cek sesi. [9]

(telah diolah kembali)

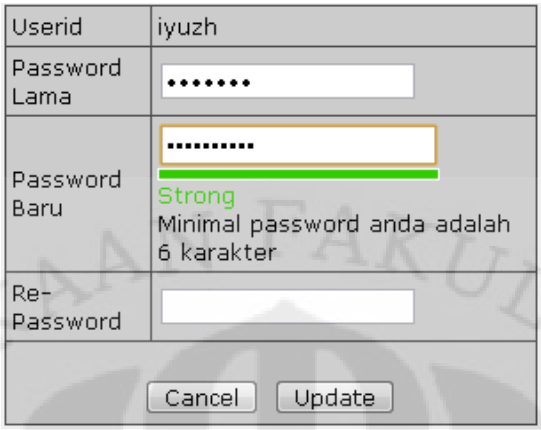
Pada sistem pewaktuan ini dibutuhkan pula suatu metode untuk *auto refresh* tampilan halaman admin.php. Tujuan dilakukan *auto refresh* tersebut adalah untuk mewaspadai saat pengguna lupa untuk keluar dari program. Waktu *auto refresh* halaman dilakukan dalam kurun waktu setiap 1 jam karena aplikasi Simple-O masih menggunakan *frame* pada tampilan setiap halaman *web*. Jika dilakukan *refresh* pada halaman tampilan *web* Simple-O, maka akan kembali ke halaman awal saat pertama kali *login* (*home page*). Dengan kondisi tersebut, maka pewaktuan *auto refresh* diberikan jarak waktu yang cukup panjang dengan tujuan tidak mengganggu pengguna yang sedang menyelesaikan soal ujian. Untuk menutup sesi, maka pada file logout.php dituliskan *destroy_session()* yang berguna untuk menghapus semua sesi yang ada saat pengguna keluar dari aplikasi Simple-O.

4.1.2 Analisis Penerapan Password Strength Meter

Penerapan *password strength meter* pada aplikasi Simple-O terletak pada saat pengguna dalam mengubah kata sandi ataupun pada saat *admin* melakukan proses registrasi pengguna. Saat mengetikkan kata sandi, maka akan terdapat parameter yang memberitahukan kepada pengguna ataupun *admin*, tentang kekuatan dari kata sandi yang dituliskan. Hal ini dilakukan karena program dari Simple-O memberikan wewenang kepada pengguna untuk membuat kata sandinya sendiri, serta memiliki tujuan agar terjadi interaksi antara sistem dari aplikasi Simple-O kepada pengguna. Dengan memberitahukan kualitas kekuatan dari kata sandi yang dibuat, diharapkan pengguna dapat membuat kata sandi yang baik, sehingga mengurangi tindakan pencurian id pengguna oleh orang lain.

Pada program *password strength meter* ini, terdiri dari 5 parameter, yaitu *very weak, weak, medium, strong, dan very strong*. Pengguna harus memasukkan minimal 6 karakter, baru akan terlihat parameter *very weak*. Penilaian terhadap parameter dari program *password strength meter* ini, menggunakan perhitungan yang menilai dari kombinasi huruf besar, huruf kecil, angka, tanda baca, serta karakter khusus seperti @,#,$,%,^,&, dsb. Dengan mengkombinasikan kata sandi dengan unsur-unsur tersebut, maka tingkat kekuatan dari kata sandi akan semakin

kuat. Tampilan dari password strength meter dapat dilihat pada Gambar 4.2 dan Gambar 4.3.



Gambar 4.2 Tampilan ubah kata sandi

Gambar 4.3 Tampilan menambah pengguna oleh admin

4.1.3 Analisis Penerapan Hash Kata Sandi Dengan MD5

Pada program awal dari Simple-O, kata sandi yang tersimpan dalam basis data masih merupakan kata sandi yang asli, tidak mengalami perubahan, seperti pada Gambar 4.4. Jika terdapat seorang yang tidak bertanggung jawab berhasil masuk ke dalam sistem basis data Simple-O, maka orang tersebut sudah dapat mencuri setiap identitas dari para pengguna Simple-O karena kata sandi dari setiap pengguna masih dapat dilihat dan dimengerti oleh manusia.

			userid	password	nama	email	idgroup	matkul
			user1	user1	user1	user1@user.com	2	1
			user2	user2	user2	user1@user.com	2	1
			user3	user3	user3	user1@user.com	2	1
			user4	user4	user4	user1@user.com	2	1
			user5	user5	user5	user1@user.com	2	1
			user6	user6	user6	user1@user.com	2	1
			user7	user7	user7	user1@user.com	2	1
			user8	user8	user8	user1@user.com	2	1
			user9	user9	user9	user1@user.com	2	1
			user10	user10	user10	user10@user.com	2	1

Gambar 4.4 Kata sandi sebelum menggunakan *hash* MD5

Hash dengan MD5 merupakan program yang terdapat dalam PHP *engine*, sehingga dalam penggunaannya dapat langsung menuliskan md5 pada skrip PHP. Dengan demikian PHP *engine* sudah dapat mengerti perintah tersebut. Dengan menggunakan proses hash MD5 pada kata sandi, maka kata sandi tersebut akan mengalami proses *hash* sehingga kata sandi yang tersimpan di dalam basis data bukanlah kata sandi yang sebenarnya. Algoritma dari hash MD5 ini adalah menerima input dari sejumlah karakter yang bebas, kemudian mengubahnya dan menghasilkan output heksadesimal sepanjang 32 karakter. Hasil dari penggunaan *hash* MD5 pada kata sandi dapat dilihat pada Gambar 4.5.

			userid	password	nama	email	idgroup	matkul
			user1	af957b6f697a839d2094ef88fd954174	user1	user1@user.com	2	
			user2	062fa54668ffc484cd2a328a4003c090	user2	user2@user.com	2	
			user3	0ca52c6e50161aaae44be185c9dda34a	user3	user3@user.com	2	
			user4	f0b3952d0f777ac247617a5e79d068ce	user4	user4@user.com	2	
			user5	35e9bbd4ca087d90e57c775b75b80c2d	user5	user5@user.com	2	
			user6	8ce9af9081d14107761b1bea422278b5	user6	user6@user.com	2	
			user7	f2e1a426e56c0ba1350b6a12a410210f	user7	user7@user.com	2	
			user8	22ef8f4d16c1296f9aaa678275ffc7b2	user8	user8@user.com	2	
			user9	c0658c9c651e8d870fc9df784752bec0	user9	user9@user.com	2	
			user10	78dfe0a38fa4385eae0447b349d16405	user10	user10@user.com	2	

Gambar 4.5 Kata sandi sesudah menggunakan *hash* MD5

Sebaiknya, saat menggunakan *hash* MD5 tidak langsung menggunakan skrip tersebut untuk sebuah kata sandi, melainkan dibutuhkan suatu variabel

pengacak untuk mengkombinasikan kata sandi yang akan mengalami proses *hash*. Alasan dari pemberian variabel pengacak pada proses *hash* ini dikarenakan terdapat banyak layanan yang berfungsi untuk mengembalikan nilai hasil dari proses *hash* MD5 ke nilai aslinya. Sehingga, jika terjadi seseorang yang berhasil masuk ke sistem basis data dan ingin mencuri kata sandi dari seseorang, maka orang tersebut tidak akan mendapatkan kata sandi yang asli.

4.1.4 Analisis Penerapan Lupa dan Ubah Kata Sandi

Penerapan lupa kata sandi pada aplikasi Simple-O ini, bukanlah memberitahukan kata sandi dari pengguna, melainkan memberikan kata sandi baru yang dibuat oleh sistem secara acak dengan panjang 6 karakter. Pemberian kata sandi yang dibuat oleh sistem terdiri dari huruf besar, huruf kecil, dan angka. Pada saat masuk ke form lupa kata sandi, maka pengguna akan ditanyakan alamat kotak suratnya (*email*). Dengan memasukkan alamat kotak surat, maka sistem akan mengecek alamat kotak surat tersebut terdaftar dalam basis data atau tidak. Jika alamat kotak surat dari pengguna terdaftar dalam sistem basis data, maka proses reset kata sandi akan berlangsung.

Penerapan dari ubah kata sandi, dapat dilakukan setelah pengguna melakukan proses *login*. Hal ini dikarenakan pada aplikasi Simple-O ini menggunakan sistem *login* terkunci dimana pengguna yang sudah masuk ke dalam aplikasi Simple-O tidak dapat ditembus oleh orang lain, seperti yang telah dijelaskan pada sub bab 3.6. Dengan mengasumsikan sebuah kondisi dimana terdapat seorang admin yang siaga pada saat ujian berlangsung untuk mengantisipasi terjadinya salah seorang pengguna yang tidak bisa masuk ke dalam aplikasi Simple-O melalui akunnya.

4.1.5 Analisis Penerapan Proses *Login* dengan Sistem Terkunci

Penerapan proses *login* dengan sistem terkunci ini memerlukan penambahan sebuah *field* pada basis data aplikasi Simple-O yang terletak pada tabel tbuser. Penambahan satu *field* ini bertujuan untuk memberikan sebuah tanda apabila seorang pengguna sedang *login* atau tidak. Nama *field* yang ditambahkan pada tbuser tersebut adalah sign dengan tipe *boolean*, dimana nilai dari data yang

masuk berupa 0 dan 1. Nilai 0 yang dimaksud adalah kondisi dimana sebuah akun sedang tidak aktif, sedangkan nilai 1 mengartikan bahwa sebuah akun sedang aktif. Pada program yang berlangsung diperlukan sebuah penambahan algoritma pada bagian cekuser.html dan logout.html. Pada bagian cekuser.html perlu ditambahkan cek keadaan sign pada basis data dimana nilai dari sign=0. Setelah dicek pada basis data dan nilai sign dari suatu akun pengguna bernilai 0, maka pengguna dapat masuk ke program, dan sistem mengupdate basis data dimana nilai sign diubah menjadi 1 sebagai tanda bahwa akun tersebut sedang aktif. Tidak lupa pada bagian logout.html ditambahkan suatu kode yang mengembalikan nilai dari sign menjadi posisi semula yaitu bernilai 0 ketika pengguna keluar dari program. Hasil penerapan proses *login* dengan sistem terkunci, dapat dilihat pada Gambar 4.6.

			userid	password	nama	email	idgroup	matkul	sign
☐	✎	✕	user1	af957b6f697a839d2094ef88fd954174	user1	user1@user.com	2		0
☐	✎	✕	user2	062fa54668ffc484cd2a328a4003c090	user2	user2@user.com	2		0
☐	✎	✕	user3	0ca52c6e50161aaae44be185c9dda34a	user3	user3@user.com	2		0
☐	✎	✕	user4	f0b3952d0f777ac247617a5e79d068ce	user4	user4@user.com	2		0
☐	✎	✕	user5	35e9bbd4ca087d90e57c775b75b80c2d	user5	user5@user.com	2		0
☐	✎	✕	user6	8ce9af9081d14107761b1bea422278b5	user6	user6@user.com	2		0
☐	✎	✕	user7	f2e1a426e56c0ba1350b6a12a410210f	user7	user7@user.com	2		0
☐	✎	✕	user8	22ef8f4d16c1296f9aaa678275ffc7b2	user8	user8@user.com	2		0
☐	✎	✕	user9	c0658c9c651e8d870fc9df784752bec0	user9	user9@user.com	2		0
☐	✎	✕	user10	78dfe0a38fa4385eae0447b349d16405	user10	user10@user.com	2		0

Gambar 4.6 Basis data tabel tbuser dengan sign

4.1.6 Analisis Penerapan CAPTCHA

Penerapan CAPTCHA pada aplikasi Simple-O ini diterapkan pada bagian *form login*. CAPTCHA yang diterapkan disini menggunakan program CAPTCHA yang memiliki kriteria CAPTCHA yang baik dengan lisensi GNU/GPL karena gambar teks CAPTCHA mudah terbaca oleh manusia serta terdistorsi secara acak sebelum disajikan kepada pengguna sehingga meminimalisir terjadinya celah keamanan disini. Pada *script form login* perlu ditambahkan sebuah kode untuk memanggil *script* CAPTCHA tersebut. Sedangkan pada file cekuser.html diperlukan sebuah algoritma, dimana akan mengecek pengguna memasukkan data yang sesuai atau tidak sesuai dengan gambar dari CAPTCHA tersebut.

```
Proc captcha_cek()
$img = [tampilan CAPTCHA];
$valid = $img->check([kode yang dimasukkan pengguna]);
If ($valid == true)
{
        login;
}
Else
{
        echo “Login Failed. Input the right code”
}
Eproc
```

Gambar 4.7 Algoritma cek CAPTCHA pada cekuser.html [14]

(telah diolah kembali)

Gambar 4.7 merupakan algoritma yang menerangkan proses memanggil fungsi dari Securimage() dengan mengidentifikasikan ke dalam variabel img. Kemudian mencocokkan antara variabel img dengan masukkan dari pengguna yang menuliskan teks pada gambar CAPTCHA. Nilai hasil cek tersebut, dimasukkan ke dalam variabel valid. Kemudian dilakukan kondisi *if else*, dimana jika variabel valid bernilai benar/cocok maka pengguna dapat *login* ke dalam program Simple-O. Namun jika salah, maka pengguna diminta untuk memasukkan kembali teks pada gambar CAPTCHA dengan benar.

4.2 Skenario Pengujian Aplikasi

Untuk mengetahui pengaruh implementasi pengamanan pada aplikasi Simple-O maka dilakukan beberapa skenario pengujian. Pengujian dilakukan pada saat proses autentifikasi. Hal ini dikarenakan perubahan proses yang paling besar terletak pada proses autentifikasi pengguna karena melibatkan proses cek kata sandi dan CAPTCHA. Sedangkan pada penerapan sistem keamanan seperti lupa kata sandi, ubah kata sandi, serta *password strength meter* merupakan program tambahan dimana tidak ada parameter untuk perbandingan performasi. Tujuan pengujian ini dilakukan untuk menganalisis perbandingan tingkat kecepatan

proses sistem yang telah diterapkan beberapa sistem pengamanan dengan program yang belum diterapkan sistem keamanan sebagai parameternya. Proses pengujian dilakukan dengan sistem *server-client*, dimana terdapat satu komputer *server* dan beberapa komputer *client* yang terhubung dengan *internet*. Skenario pengujian yang dilakukan adalah:

a. Pengujian dilakukan dengan melakukan proses *login* ke dalam aplikasi Simple-O yang belum diterapkan sistem keamanan dan aplikasi Simple-O yang sudah diterapkan sistem keamanan dengan jumlah 10 pengguna yang *login*.
b. Pengujian dilakukan dengan melakukan proses *login* ke dalam aplikasi Simple-O yang belum diterapkan sistem keamanan dan aplikasi Simple-O yang sudah diterapkan sistem keamanan dengan jumlah 20 pengguna yang *login*.
c. Pengujian dilakukan dengan melakukan proses *login* ke dalam aplikasi Simple-O yang belum diterapkan sistem keamanan dan aplikasi Simple-O yang sudah diterapkan sistem keamanan dengan jumlah 30 pengguna yang *login*.

4.2.1 Skenario Pengujian 1

Pada skenario pengujian 1, dilakukan pengujian dari pengguna yang masuk ke dalam 2 aplikasi, yaitu aplikasi Simple-O yang belum diterapkan keamanan dan aplikasi Simple-O yang sudah diterapkan keamanan. Aplikasi Simple-O yang belum diterapkan sistem keamanan akan menjadi parameter pengujian ini. Skenario ini dilakukan dengan menggunakan 10 komputer *client* yang melakukan proses *login* secara bersamaan dalam satu waktu. Pada saat pengguna *login*, dilakukan perhitungan waktu untuk kecepatan proses *login*. Waktu mulai terhitung dari saat *submit* sampai sebelum tampilan awal dari aplikasi Simple-O. Algoritma perhitungan waktu kecepatan proses *login* seperti pada Gambar 4.8. Hasil pengujian yang tersimpan dalam database, dapat dilihat pada Gambar 4.9 untuk pengujian pada simple-o lama dan 4.10 untuk pengujian pada simple-o baru.

```
Proc delay_login()
$starttime = microtime(true);

        [Proses autentifikasi]

$endtime = microtime(true);
$delay = $starttime - $endtime;
Eproc
```

Gambar 4.8 Algoritma perhitungan waktu delay

(PHP manual)

			userid	delay	time
☐	✎	✕	user8	0.0212469100952	2010-06-04 14:38:10
☐	✎	✕	user1	0.0207591056824	2010-06-04 14:38:10
☐	✎	✕	user2	0.0225100517273	2010-06-04 14:38:10
☐	✎	✕	user5	0.0229249000549	2010-06-04 14:38:10
☐	✎	✕	user10	0.0174119472504	2010-06-04 14:38:10
☐	✎	✕	user4	0.0231258869171	2010-06-04 14:38:12
☐	✎	✕	user3	0.0459852218628	2010-06-04 14:38:13
☐	✎	✕	user7	0.0342361927032	2010-06-04 14:38:13
☐	✎	✕	user6	0.0384151935577	2010-06-04 14:38:13
☐	✎	✕	user9	0.0228350162506	2010-06-04 14:38:13

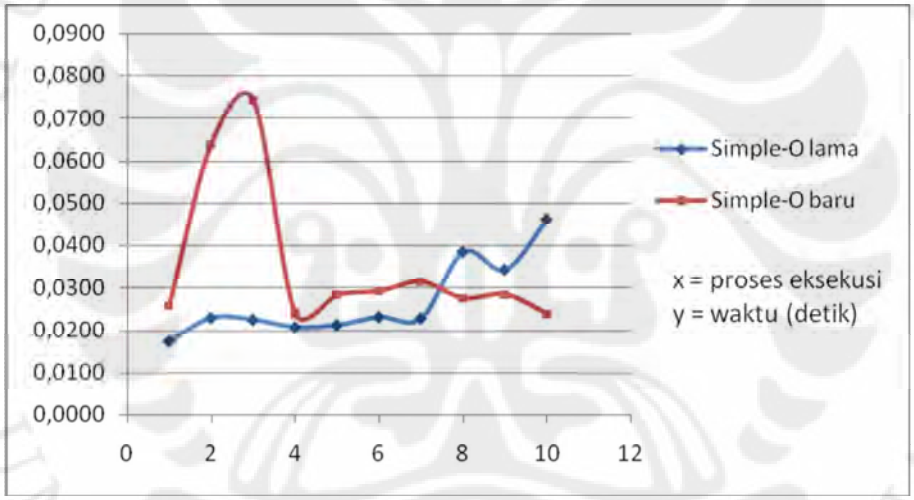
Gambar 4.9 Hasil pengujian 1 pada simple-o lama

			userid	delay	time
☐	✎	✕	user5	0.0257658958435	2010-06-04 14:45:53
☐	✎	✕	user6	0.0639798641205	2010-06-04 14:45:53
☐	✎	✕	user7	0.074385881424	2010-06-04 14:45:53
☐	✎	✕	user1	0.0239272117615	2010-06-04 14:45:55
☐	✎	✕	user8	0.0282781124115	2010-06-04 14:45:56
☐	✎	✕	user9	0.0291290283203	2010-06-04 14:45:56
☐	✎	✕	user10	0.0313470363617	2010-06-04 14:45:57
☑	✎	✕	user2	0.0274660587311	2010-06-04 14:46:20
☑	✎	✕	user4	0.0283589363098	2010-06-04 14:46:32
☑	✎	✕	user3	0.0237910747528	2010-06-04 14:47:01

Gambar 4.10 Hasil pengujian 1 pada simple-o baru

Tabel 4.1: Tabel hasil pengujian 1

10 peserta			
Id Pengguna	Waktu simple-o lama(s)	Id Pengguna	Waktu simple-o baru(s)
user10	0,0174	user5	0,0258
user5	0,0229	user6	0,0640
user2	0,0225	user7	0,0744
user1	0,0208	user1	0,0239
user8	0,0212	user8	0,0283
user4	0,0231	user9	0,0291
user9	0,0228	user10	0,0313
user6	0,0384	user2	0,0275
user7	0,0342	user4	0,0284
user3	0,0460	user3	0,0238
Rata-rata	0,0269	Rata-rata	0,0356



Gambar 4.11 Grafik proses eksekusi pengujian 1

Dengan menggunakan algoritma perhitungan waktu diatas, maka didapatkan data perbandingan waktu proses eksekusi pada tabel 4.1 dan Gambar 4.11. Perhitungan rata-rata waktu kecepatan proses *login,* dilakukan dengan pembulatan 4 angka di belakang koma(,). Seperti contoh 0,01839207860291 maka dibulatkan menjadi 0,0184. Berdasarkan data dari 10 peserta yang *login* pada alamat web simple-o lama, maka rata-rata kecepatan proses adalah 0,0269 detik. Sedangkan rata-rata waktu kecepatan proses *login* pada simple-o baru adalah 0,0356.

Namun, pada proses *login* aplikasi Simple-O yang sudah diterapkan keamanan, terdapat 3 orang pengguna dari 10 pengguna yang salah dalam memasukkan kode CAPTCHA. Hal ini mengakibatkan hanya 7 pengguna yang berhasil menjalankan proses *login* secara bersamaan. Berdasarkan analisis terhadap program CAPTCHA yang diterapkan untuk proses autentifikasi, tingkat kesalahan dari pengguna sebesar 30 % dalam membaca kode CAPTCHA tersebut.

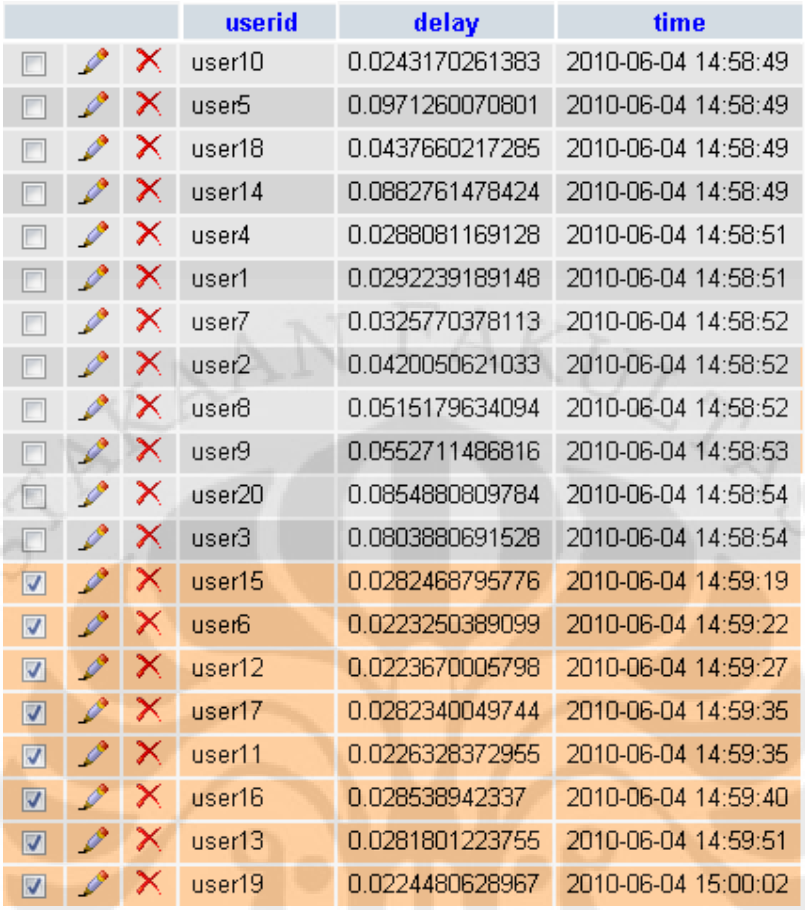
Pada proses perhitungan yang didapat, beberapa proses yang dijalankan secara bersamaan akan mengalami waktu proses yang lebih lama. Berdasarkan data yang didapat, proses terlama pada alamat web simple-o lama sebesar 0,0460 detik oleh user3 dengan urutan proses *login* ke-7. Sedangkan proses terlama pada alamat web simple-o baru sebesar 0,0744 detik oleh user7 dengan urutan proses *login* ke-3. Jika dibandingkan dengan rata-rata waktu proses pada kedua alamat web tersebut, maka perbandingan 2 proses terlama mencapai ± 2x rata-rata waktu proses. Dari pengujian 1, selisih nilai rata-rata proses *login* aplikasi Simple-O yang lama dan aplikasi Simple-O yang sudah diterapkan sistem keamanan sebesar 0,0087 detik.

4.2.2 Skenario Pengujian 2

Pada skenario pengujian 2, dilakukan dengan menggunakan 20 komputer *client* yang melakukan proses *login* secara bersamaan dalam satu waktu. Pada saat pengguna *login*, dilakukan perhitungan waktu untuk menghitung kecepatan proses *login*, dari saat *submit* sampai sebelum tampilan awal dari aplikasi Simple-O. Hasil pengujian yang tersimpan dalam database, dapat dilihat pada Gambar 4.12 untuk pengujian pada simple-o lama dan 4.13 untuk pengujian pada simple-o baru.

			userid	delay	time
☐			user13	0.0197069644928	2010-06-04 14:51:40
☐			user5	0.0234398841858	2010-06-04 14:51:40
☐			user4	0.0170819759369	2010-06-04 14:51:40
☐			user15	0.01997590065	2010-06-04 14:51:40
☐			user10	0.0226349830627	2010-06-04 14:51:40
☐			user2	0.0171849727631	2010-06-04 14:51:41
☐			user16	0.0316939353943	2010-06-04 14:51:43
☐			user18	0.0360159873962	2010-06-04 14:51:43
☐			user8	0.023766040802	2010-06-04 14:51:43
☐			user14	0.0231540203094	2010-06-04 14:51:43
☐			user6	0.0482001304626	2010-06-04 14:51:44
☐			user1	0.0433268547058	2010-06-04 14:51:44
☐			user17	0.0472588539124	2010-06-04 14:51:44
☐			user9	0.0519330501556	2010-06-04 14:51:44
☐			user19	0.0564379692078	2010-06-04 14:51:45
☐			user7	0.0341899394989	2010-06-04 14:51:45
☐			user11	0.0550630092621	2010-06-04 14:51:45
☐			user20	0.0188400745392	2010-06-04 14:51:45
☐			user12	0.0188429355621	2010-06-04 14:51:45
☐			user3	0.0231969356537	2010-06-04 14:51:45

Gambar 4.12 Hasil pengujian 2 pada simple-o lama



	userid	delay	time
	user10	0.0243170261383	2010-06-04 14:58:49
	user5	0.0971260070801	2010-06-04 14:58:49
	user18	0.0437660217285	2010-06-04 14:58:49
	user14	0.0882761478424	2010-06-04 14:58:49
	user4	0.0288081169128	2010-06-04 14:58:51
	user1	0.0292239189148	2010-06-04 14:58:51
	user7	0.0325770378113	2010-06-04 14:58:52
	user2	0.0420050621033	2010-06-04 14:58:52
	user8	0.0515179634094	2010-06-04 14:58:52
	user9	0.0552711486816	2010-06-04 14:58:53
	user20	0.0854880809784	2010-06-04 14:58:54
	user3	0.0803880691528	2010-06-04 14:58:54
	user15	0.0282468795776	2010-06-04 14:59:19
	user6	0.0223250389099	2010-06-04 14:59:22
	user12	0.0223670005798	2010-06-04 14:59:27
	user17	0.0282340049744	2010-06-04 14:59:35
	user11	0.0226328372955	2010-06-04 14:59:35
	user16	0.028538942337	2010-06-04 14:59:40
	user13	0.0281801223755	2010-06-04 14:59:51
	user19	0.0224480628967	2010-06-04 15:00:02

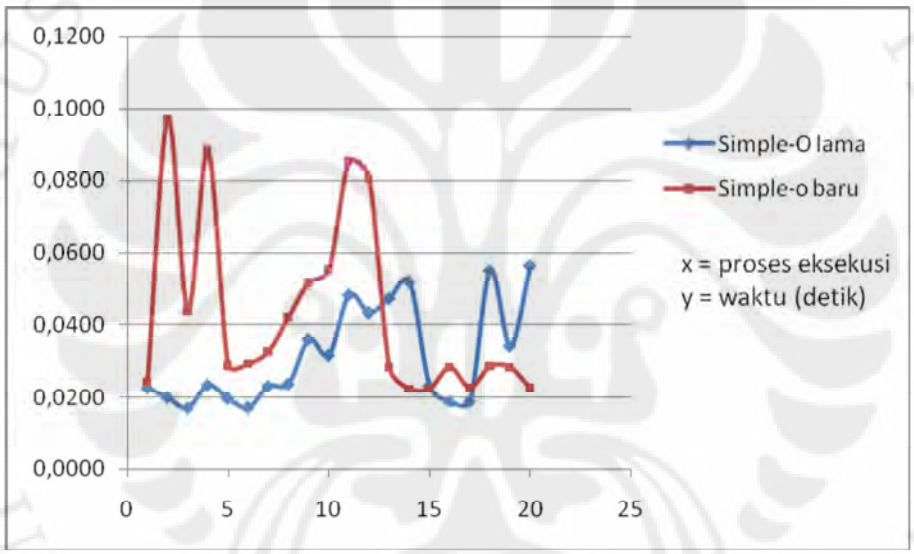
Gambar 4.13 Hasil pengujian 2 pada simple-o baru

Tabel 4.2: Tabel hasil pengujian 2

20 peserta			
Id Pengguna	waktu simple-o lama (s)	Id Pengguna	waktu simple-o baru (s)
user10	0,0226	user10	0,0243
user15	0,0200	user5	0,0971
user4	0,0171	user18	0,0438
user5	0,0234	user14	0,0883
user13	0,0197	user4	0,0288
user2	0,0172	user1	0,0292
user14	0,0232	user7	0,0326
user8	0,0238	user2	0,0420
user18	0,0360	user8	0,0515
user16	0,0317	user9	0,0553
user6	0,0482	user20	0,0855
user1	0,0433	user3	0,0804

Tabel 4.2: Tabel hasil pengujian 2 (Lanjutan)

20 peserta			
Id Pengguna	Waktu simple-o lama(s)	Id Pengguna	Waktu simple-o baru(s)
user17	0,0473	user15	0,0282
user9	0,0519	user6	0,0223
user3	0,0232	user12	0,0224
user12	0,0188	user17	0,0282
user20	0,0188	user11	0,0226
user11	0,0551	user16	0,0285
user7	0,0342	user13	0,0282
user19	0,0564	user19	0,0224
Rata-rata	0,0382	Rata-rata	0,0254



Gambar 4.14 Grafik proses eksekusi pengujian 2

Dengan menggunakan algoritma perhitungan waktu kecepatan proses pada Gambar 4.8, maka didapatkan data perbandingan seperti Tabel 4.2 dan Gambar 4.14. Perhitungan rata-rata kecepatan proses *login*, dilakukan dengan pembulatan 4 angka di belakang koma(,). Berdasarkan data dari 20 peserta yang *login* pada alamat web simple-o lama, maka rata-rata kecepatan proses *login* adalah 0,0316 detik. Sedangkan, rata-rata kecepatan proses *login* pada simple-o baru adalah 0,0431 detik.

Dari data yang didapatkan pada proses *login* aplikasi simple-o baru, terdapat 8 orang pengguna dari 20 pengguna yang salah dalam memasukkan kode

CAPTCHA. Pada pengujian kedua ini tingkat kesalahan dalam membaca kode CAPTCHA meningkat menjadi 40 %. Namun, dilihat pada Gambar 4.12, peningkatan banyak terjadi pada pengguna 11-20, sebanyak 7 pengguna. Sedangkan dari pengguna 1-10 mengalami penurunan dari 3 pengguna menjadi 1 pengguna yang salah. Dari 2 pengujian yang telah dilakukan, dapat ditarik analisis sementara bahwa pada pengguna yang baru masih belum terbiasa dengan tampilan kode CAPTCHA yang terdapat pada proses *login* simple-o.

Pada proses perhitungan yang didapat, beberapa proses yang dijalankan secara bersamaan akan mengalami waktu proses yang lebih lama. Berdasarkan data yang didapat, proses terlama pada alamat web simple-o lama sebesar 0,0564 detik oleh user19 dengan urutan proses *login* ke-15. Sedangkan proses terlama pada alamat web simple-o baru sebesar 0,0971 detik oleh user5 dengan urutan proses *login* ke-2. Jika dibandingkan dengan rata-rata waktu proses pada kedua alamat web tersebut, maka perbandingan 2 proses terlama mencapai ± 2x rata-rata waktu proses.

Dari pengujian 2, selisih nilai rata-rata proses *login* aplikasi Simple-O yang lama dengan aplikasi Simple-O yang sudah diterapkan sistem keamanan sebesar 0,0115 detik. Dapat disimpulkan bahwa penerapan beberapa sistem keamanan tidak memberikan pengaruh yang cukup besar pada pengujian 2.

4.2.3 Skenario Pengujian 3

Pada skenario pengujian 3, dilakukan pengujian dari pengguna yang masuk ke dalam aplikasi Simple-O yang telah diterapkan sistem keamanan. Skenario ini dilakukan dengan menggunakan 30 komputer *client* yang melakukan proses *login* secara bersamaan dalam satu waktu. Pada saat pengguna *login*, dilakukan perhitungan waktu untuk kecepatan proses terhitung dari saat *login* sampai sebelum tampilan awal dari aplikasi Simple-O. Algoritma perhitungan waktu proses *login* seperti pada Gambar 4.8. Hasil pengujian yang tersimpan dalam basis data, dapat dilihat pada Gambar 4.15 untuk pengujian pada simple-o lama dan 4.16 untuk pengujian pada simple-o baru.

			userid	delay	time
			user6	0.0225110054016	2010-06-04 15:05:17
			user10	0.0176541805267	2010-06-04 15:05:17
			user16	0.0246858596802	2010-06-04 15:05:17
			user15	0.0173888206482	2010-06-04 15:05:17
			user4	0.0172169208527	2010-06-04 15:05:18
			user17	0.0225360393524	2010-06-04 15:05:19
			user19	0.0305938720703	2010-06-04 15:05:20
			user2	0.0237009525299	2010-06-04 15:05:20
			user28	0.0226650238037	2010-06-04 15:05:20
			user27	0.0407068729401	2010-06-04 15:05:21
			user20	0.0489032268524	2010-06-04 15:05:21
			user24	0.0445120334625	2010-06-04 15:05:21
			user22	0.0472187995911	2010-06-04 15:05:21
			user1	0.0227310657501	2010-06-04 15:05:21
			user14	0.0228021144867	2010-06-04 15:05:21
			user30	0.188269853592	2010-06-04 15:05:23
			user7	0.172524929047	2010-06-04 15:05:23
			user9	0.184772014618	2010-06-04 15:05:23
			user3	0.169307947159	2010-06-04 15:05:23
			user5	0.0256969928741	2010-06-04 15:05:23
			user29	0.0315539836884	2010-06-04 15:05:23
			user18	0.171971082687	2010-06-04 15:05:23
			user8	0.19172501564	2010-06-04 15:05:23
			user25	0.173745155334	2010-06-04 15:05:23
			user12	0.167018175125	2010-06-04 15:05:23
			user26	0.188627958298	2010-06-04 15:05:23
			user11	0.186696052551	2010-06-04 15:05:23
			user23	0.195245027542	2010-06-04 15:05:23
			user13	0.183716773987	2010-06-04 15:05:23
			user21	0.183497905731	2010-06-04 15:05:23

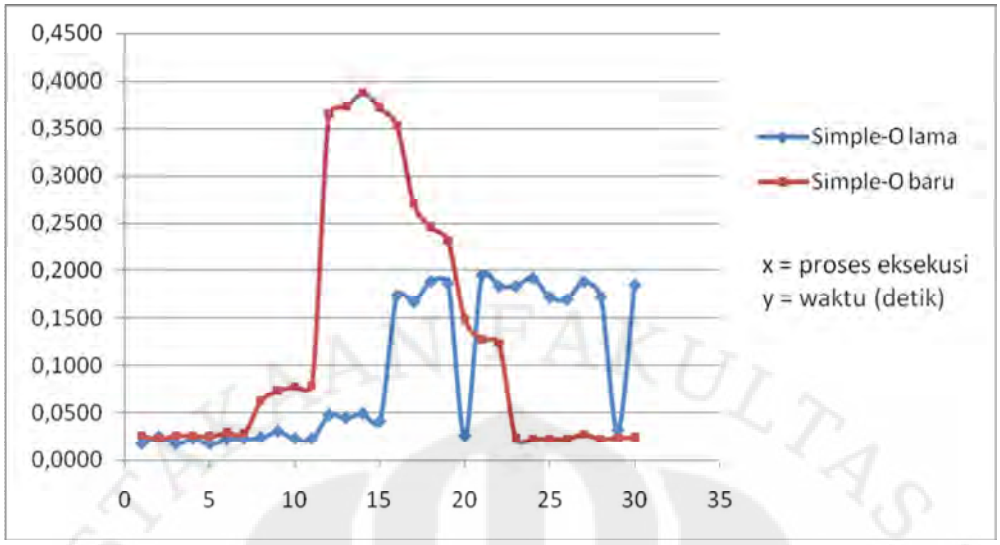
Gambar 4.15 Hasil pengujian 3 pada simple-o lama

			userid	delay	time
☐			user17	0.0255830287933	2010-06-04 15:17:00
☐			user15	0.0227138996124	2010-06-04 15:17:00
☐			user16	0.0258011817932	2010-06-04 15:17:00
☐			user10	0.0259327888489	2010-06-04 15:17:00
☐			user4	0.0250661373138	2010-06-04 15:17:00
☐			user14	0.0288398265839	2010-06-04 15:17:02
☐			user20	0.0286200046539	2010-06-04 15:17:02
☐			user19	0.0617010593414	2010-06-04 15:17:04
☐			user8	0.0729310512543	2010-06-04 15:17:04
☐			user30	0.077201128006	2010-06-04 15:17:04
☐			user29	0.0781760215759	2010-06-04 15:17:04
☐			user12	0.365118980408	2010-06-04 15:17:05
☐			user11	0.372539997101	2010-06-04 15:17:05
☐			user2	0.387544870377	2010-06-04 15:17:05
☐			user21	0.371546983719	2010-06-04 15:17:05
☐			user1	0.352483034134	2010-06-04 15:17:05
☐			user18	0.270061016083	2010-06-04 15:17:05
☐			user6	0.245478153229	2010-06-04 15:17:05
☐			user7	0.230849981308	2010-06-04 15:17:05
☐			user9	0.148694992065	2010-06-04 15:17:06
☐			user25	0.123599052429	2010-06-04 15:17:06
☐			user26	0.127867937088	2010-06-04 15:17:06
☑			user5	0.0227279663086	2010-06-04 15:17:14
☑			user27	0.022243976593	2010-06-04 15:17:33
☑			user24	0.0224459171295	2010-06-04 15:17:35
☑			user3	0.0223541259766	2010-06-04 15:17:35
☑			user13	0.0273921489716	2010-06-04 15:17:37
☑			user22	0.0224170684814	2010-06-04 15:17:44
☑			user28	0.0236580371857	2010-06-04 15:17:48
☑			user23	0.0239260196686	2010-06-04 15:18:12

Gambar 4.16 Hasil pengujian 3 pada simple-o baru

Tabel 4.3: Tabel hasil pengujian 3

30 peserta			
Id Pengguna	waktu simple-o lama(s)	Id Pengguna	waktu simple-o baru(s)
user15	0,0174	user17	0,0256
user16	0,0247	user15	0,0227
user10	0,0177	user16	0,0258
user6	0,0225	user10	0,0259
user4	0,0172	user4	0,0251
user17	0,0225	user14	0,0288
user28	0,0227	user20	0,0286
user2	0,0237	user19	0,0617
user19	0,0306	user8	0,0729
user14	0,0228	user30	0,0772
user1	0,0227	user29	0,0782
user22	0,0472	user12	0,3651
user24	0,0445	user11	0,3725
user20	0,0489	user2	0,3875
user27	0,0407	user21	0,3715
user25	0,1737	user1	0,3525
user12	0,1670	user18	0,2701
user26	0,1886	user6	0,2455
user11	0,1867	user7	0,2308
user5	0,0257	user9	0,1487
user23	0,1952	user26	0,1279
user13	0,1837	user25	0,1236
user21	0,1835	user5	0,0227
user8	0,1917	user27	0,0222
user18	0,1720	user24	0,0224
user3	0,1693	user3	0,0224
user30	0,1883	user13	0,0274
user7	0,1725	user22	0,0224
user29	0,0316	user28	0,0237
user9	0,1848	user23	0,0239
Rata-rata	0,0947	Rata-rata	0,1219



Gambar 4.17 Grafik proses eksekusi pengujian 3

Dengan menggunakan algoritma perhitungan waktu kecepatan proses pada Gambar 4.8, maka didapatkan data perbandingan pada Tabel 4.3 dan Gambar 4.17. Perhitungan rata-rata kecepatan proses *login*, dilakukan dengan pembulatan 4 angka di belakang koma(,). Berdasarkan data dari 30 peserta yang *login* pada alamat web simple-o lama, maka rata-rata kecepatan proses *login* adalah 0,0947 detik. Sedangkan, rata-rata kecepatan proses *login* pada simple-o baru adalah 0,1219 detik.

Dari data yang didapatkan pada proses *login* aplikasi simple-o, terdapat 8 orang pengguna dari 30 pengguna yang salah dalam memasukkan kode CAPTCHA. Pada pengujian ketiga ini tingkat kesalahan dalam membaca kode CAPTCHA turun menjadi 26,67 %. Kesalahan banyak terjadi pada pengguna 21-30, sebanyak 5 pengguna. Dari pengguna 11-20 mengalami penurunan drastis dari 7 pengguna menjadi 1 pengguna yang salah. Pada pengguna 1-10, mengalami penaikan 1 angka, menjadi 2 pengguna yang mengalami kesalahan. Dari 3 pengujian yang telah dilakukan, dapat ditarik analisis pada sistem CAPTCHA bahwa pada pengguna baru, masih belum terbiasa dengan tampilan kode CAPTCHA yang terdapat pada proses *login* simple-o baru.

Pada proses perhitungan yang didapat, beberapa proses yang dijalankan secara bersamaan akan mengalami waktu proses yang lebih lama. Berdasarkan data yang didapat, proses terlama pada alamat web simple-o lama sebesar 0,1953

detik oleh user23 dengan urutan proses *login* ke-28. Sedangkan proses terlama pada alamat web simple-o baru sebesar 0,3875 detik oleh user2 dengan urutan proses *login* ke-14. Jika dibandingkan dengan rata-rata waktu proses pada kedua alamat web tersebut, maka perbandingan 2 proses terlama mencapai 2x - 3x rata-rata waktu proses.

Dari pengujian 3, selisih nilai rata-rata proses *login* aplikasi Simple-O yang lama dengan aplikasi Simple-O yang sudah diterapkan sistem keamanan sebesar 0,0272 detik. Secara garis besar dapat disimpulkan bahwa penerapan beberapa sistem keamanan tidak memberikan pengaruh yang cukup besar pada pengujian 3 karena perbedaan waktu rata-rata tidak mencapai 1 detik.

4.3 Analisis Kecepatan Proses

Pada pengujian yang berhasil menjalankan proses *login* secara bersamaan, didapatkan data bahwa proses *login* yang dapat dilayani oleh *server* berbeda-beda. Bisa dilihat pada bagian waktu, bahwa tiap-tiap pengguna tercatat dalam waktu yang berbeda. Perbedaan waktu ini dikarenakan banyaknya permintaan yang masuk ke komputer *server* secara bersamaan. Pada komputer *server* menggunakan *processor dual core* dimana proses yang dapat dikerjakan oleh *server* hanya 2 proses dalam waktu yang bersamaan. Hal ini menyebabkan waktu pada beberapa pengguna untuk melakukan *login* terasa lebih lama karena terdapat waktu tunggu untuk dapat diproses oleh *processor server*. Pada data terlihat juga daftar pengguna yang tidak urut sehingga dapat disimpulkan bahwa proses ini tidak bisa diduga siapa yang akan didahulukan untuk dilayani.

Pada proses perhitungan yang didapat, beberapa proses yang dijalankan secara bersamaan akan mengalami waktu proses yang lebih lama. Dari 3 pengujian yang dilakukan, maka didapatkan data bahwa waktu *login* terlama lebih besar 2-3x dibandingkan dengan rata-rata waktu *login* pada setiap pengujian. Hal ini dipengaruhi oleh beberapa faktor diantaranya, yaitu :

a. kecepatan koneksi *internet* dari pengguna,
b. kecepatan proses dari *server*, dan
c. tingkat kesibukan dari *server*.

Berdasarkan hasil dari pengujian 1, 2, dan 3 maka didapatkan hasil rata-rata waktu proses *login* dari masing-masing pengujian. Dengan hasil pengujian diatas maka dapat dilihat pengaruh dari penerapan sistem keamanan pada aplikasi Simple-O, yaitu:

a. 32,34 % pada pengujian 1, dengan rata-rata peningkatan 0,0087 detik.
b. 36,39 % pada pengujian 2, dengan rata-rata peningkatan 0,0115 detik.
c. 28,72 % pada pengujian 3, dengan rata-rata peningkatan 0,0272 detik.

Dari peningkatan waktu yang didapatkan, secara garis besar dapat disimpulkan bahwa persentase dari penerapan sistem keamanan cukup besar yang mencapai angka 36,39 %. Namun, jika dilihat nilai peningkatan dalam satuan detik, secara garis besar peningkatan ini masih terlihat kecil karena rata-rata peningkatan yang terjadi masih dalam kurun waktu dibawah 1 detik.

Pada pengujian ini, masih ditemukan kendala pengguna yang gagal dalam melakukan proses *login* karena salah memasukan kode CAPTCHA. Untuk mengetahui lebih jauh pengaruh dari jumlah pengguna yang melakukan *login* secara bersamaan terhadap kinerja proses *login*, maka diperlukan penelitian lebih lanjut dengan skenario pengujian yang lebih banyak dan memastikan semua pengguna berhasil melakukan proses *login* dengan baik.

4.4 Analisis Kuisioner

Setelah proses pengujian dilakukan, tidak lupa untuk memberikan kuisioner kepada para pengguna untuk menilai aplikasi Simple-O yang sudah diterapkan beberapa sistem keamanan. Kuisioner yang diberikan kepada para pengguna berupa pilihan ganda dengan pertanyaan:

a. Apakah aplikasi Simple-O ini mudah untuk digunakan?
b. Bagaimana pendapat Anda tentang kenyamanan penggunaan aplikasi Simple-O?

Sedangkan tingkat jawaban yang diberikan adalah nilai A-E dimana:

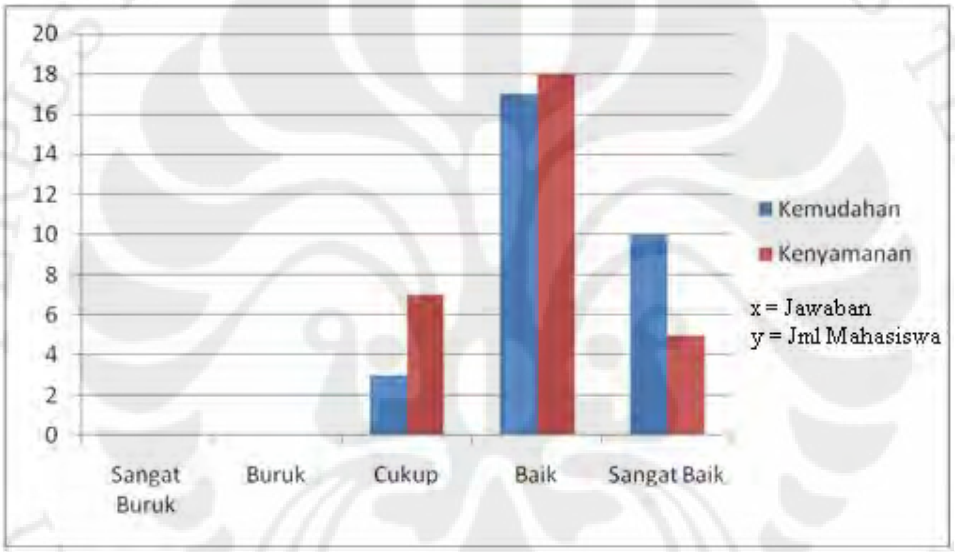
a. A merupakan jawaban sangat buruk;
b. B merupakan jawaban buruk;

c. C merupakan jawaban cukup;
d. D merupakan jawaban baik;
e. E merupakan jawaban sangat baik.

Hasil jawaban dari kuisioner yang diberikan pengguna dapat dilihat pada Tabel 4.4 dan Gambar 4.18.

Tabel 4.4: Tabel jawaban kuisioner dari para pengguna

	Sangat Buruk	Buruk	Cukup	Baik	Sangat Baik
Kemudahan			3	17	10
Kenyamanan			7	18	5



Gambar 4.18 Grafik jawaban kuisioner dari para pengguna

Berdasarkan Tabel 4.4 dan Gambar 4.18 dapat kita tarik sebuah kesimpulan untuk masing-masing pertanyaan, yaitu:

a. Untuk soal nomor 1, mayoritas dari pengguna berpendapat bahwa aplikasi Simple-O ini mudah untuk digunakan dengan perolehan nilai sebesar 56,67%.
b. Untuk soal nomor 2, mayoritas dari pengguna berpendapat bahwa aplikasi Simple-O yang sudah diterapkan beberapa sistem keamanan, memiliki tingkat kenyamanan yang baik dengan perolehan nilai sebesar 60% dari para pengguna.

Dengan kesimpulan yang didapatkan dari masing-masing pertanyaan, dapat ditarik kesimpulan secara menyeluruh bahwa aplikasi Simple-O ini baik diterapkan sebagai penilaian ujian esei dan tidak mengganggu kenyamanan dari para pengguna. Namun diperlukan pengembangan dari segi tampilan aplikasi Simple-O agar lebih menarik untuk dilihat para pengguna.

BAB 5
KESIMPULAN

Dari hasil uji coba dan analisis yang telah dilakukan dapat diambil beberapa kesimpulan, yaitu:

a. Penerapan sesi dengan pewaktuan berhasil diterapkan pada aplikasi Simple-O, merupakan pengamanan dari identitas pengguna yang lupa keluar dari program.
b. Penerapan *password strength meter* pada aplikasi Simple-O memberikan respon kepada pengguna saat membuat kata sandi. Dengan memberitahukan tingkat kekuatan dari kata sandi tersebut, diharapkan pengguna tidak membuat kata sandi dengan tingkat kekuatan rendah.
c. Penerapan hash MD5 pada kata sandi berhasil diterapkan pada aplikasi Simple-O yang berfungsi untuk mengamankan kata sandi pengguna di dalam database.
d. Penerapan lupa dan ubah kata sandi berhasil diterapkan pada aplikasi Simple-O. Penerapan ini merupakan tahap antisipasi bila seorang pengguna lupa atau dicuri kata sandinya.
e. Penerapan proses login terkunci berhasil diterapkan pada aplikasi Simple-O yang bertujuan membuat seseorang yang sudah masuk program, tidak dapat ditembus oleh orang lain. Mengasumsikan kondisi saat mengerjakan ujian.
f. Penerapan CAPTCHA berhasil diterapkan pada aplikasi Simple-O. Penerapan ini dilakukan untuk mengantisipasi pembobolan proses autentifikasi dengan menggunakan program komputer.
g. Penerapan beberapa sistem keamanan menyebabkan sistem bekerja lebih lama terutama dalam proses autentifikasi. Peningkatan terjadi berkisar antara 28% - 37% yang telah dilakukan pada pengujian 1 sampai pengujian 3. Namun, tambahan waktu yang diperlukan untuk proses autentifikasi tidak terlalu signifikan untuk dapat dirasakan oleh manusia yaitu berkisar 0,0087 – 0,0272 detik.

h. Pengembangan sistem keamanan yang dilakukan pada aplikasi Simple-O ini, tidak mengganggu kenyamanan pengguna dalam menggunakan aplikasi Simple-O.

DAFTAR PUSTAKA

1. Anak Agung Putri Ratna, Bagio Budiardjo dan Djoko Hartanto, "SIMPLE: Sistem Penilaian Esei Otomatis untuk Menilai Ujian dalam Bahasa Insonesia", Jurnal Makara Seri Teknologi, volume 11, April 2007, ISSN : 1693-6698
2. Anak Agung Putri Ratna, Adhe W. Astato, Bagio Budiardjo, Djoko Hartanto, "Simple-O: Web Based Automated Essay Grading System Using Latent Semantic Analysis method for Indonesian Language Considering Weight Word and Word Synonym", The 10th International Conference on Quality in Research, Faculty of Engineering, University of Indonesia, 4 – 6 December 2007, Depok, Indonesia.
3. Schlossnagle, George. Advanced PHP Programming. Indianapolis: Developer's Library, 2004.
4. Hermawandi, Dudi. Impementasi Skema Pembobotan SICBI Pada Aplikasi Essay Grading Metode Latent Symantec Analysis. Depok: Departemen Teknik Elektro Universitas Indonesia, 2008.
5. PHP Manual (n.d.). 26 Juli 2009. http://php.net/download-docs.php

DAFTAR ACUAN

[1] Jeni, "Bab 11 Keamanan WEB". http://poss.ipb.ac.id/files/JENI-Web%20Programming-Bab%2011-Keamanan%20WEB.pdf (1 Juni 2010, 20:07)

[2] CAPTCHA: Telling Humans and Computers Apart Automatically. (n.d) http://www.captcha.net/ (6 Mei 2010, 10:29)

[3] "Prosedur Penilaian", 19 Januari 2010. http://www.docstoc.com/docs/22787096/Prosedur-Penilaian (1 Juni 2010, 21:34)

[4] Napitupulu, Ester Lince, "Soal Pilihan Ganda Menjerumuskan", Jakarta: Kompas. November 1, 2009.

http://nasional.kompas.com/read/xml/2009/11/01/19445564/soal.pilihan.ganda.menjerumuskan (4 Des 2009, 14:31)

[5] Anak Agung Putri Ratna, Bagio Budiardjo dan Djoko Hartanto, "SIMPLE: Sistem Penilaian Esei Otomatis untuk Menilai Ujian dalam Bahasa Insonesia", Jurnal Makara Seri Teknologi, volume 11, April 2007, ISSN : 1693-6698

[6] Hermawandi, Dudi. Impementasi Skema Pembobotan SICBI Pada Aplikasi Essay Grading Metode Latent Symantec Analysis. Depok: Departemen Teknik Elektro Universitas Indonesia, 2008.

[7] Kadir, Abdul, "Belajar Database menggunakan MySQL", Penerbit ANDI Yogyakarta, 2008.

[8] Schlossnagle, George. Advanced PHP Programming. Indianapolis: Developer's Library, 2004.

[9] "Script pengatur waktu Session, Logout Otomatis", 30 Mei 2009. http://www.ilmuwebsite.com/tutorial-php/script-pengatur-waktu-session-logout-otomatis (28 April 2010, 23:52)

[10] Abrari, "Algoritma hash MD5", 2 April 2010, 11:25. http://abrari.wordpress.com/2010/04/02/algoritma-hash-md5/ (24 Mei 2010, 01:12)

[11] Resig, John, "jQuery 1.2.1 - New Wave Javascript", 16 Sep 2007. www.jquery.com (1 Mei 2010, 12:06)

[12] Ari, Rosihan, "Tips Membuat Script PHP Pengolah Password dengan MD5", 13 September 2008. http://blog.rosihanari.net/tips-membuat-script-php-pengolah-password-dengan-md5 (30 April 2010, 01: 03)

[13] Ari, Rosihan, "Membuat Fasilitas Lost Password dengan PHP", 23 September 2008. http://blog.rosihanari.net/membuat-fasilitas-lost-password-dengan-php (30 April 2010, 01: 19)

[14] Phillips, Drew, "Securimage: A PHP class for creating and managing form CAPTCHA images", 23 Maret 2008. http://www.phpcaptcha.org (30 Mei 2010, 11:47)