Kombinasi Waktu Sinkronisasi dan Nilai *Salt* untuk Peningkatan Keamanan pada *Single Sign-On*

Rizal Munadi[1], Zuhar Musliyana[2], Teuku Yuliar Arif[3], Afdhal[4], Syahrial[5]

***Abstract*—Single sign-on (SSO) is a session authentication process that allows a user to login by using user registered identity and password in order to access appropriate applications. The authentication process takes the user in to login for all the applications they have been given rights to and eliminates further prompts when they switch applications during a particular session. Its implementation will provide a reduction of password burden to access many applications for every login process. Ease of access through a single account needs to be addressed carefully to ensure the authentication credentials that are not scattered and known by others. Currently, there are several open source SSO authentication methods available. However, the use of existing authentication methods is still vulnerable to attack, such as Man-In-The-Middle. In this study, SSO authentication algorithm using One-Time Password (OTP) is proposed using a combination of time synchronization and salt value. These combinations are used to verify user session while accessing any application with SSO mechanism. The results show that the proposed OTP algorithm can handle SSO authentication process in good fashion and also protect from Man-In-The-Middle Attack.**

***Intisari*— *Single Sign-On* (SSO) merupakan salah satu proses autentikasi yang mengizinkan pengguna untuk mengakses aplikasi tertentu dengan menggunakan identitas yang membolehkan pengguna untuk masuk ke dalam semua aplikasi yang telah diberikan hak akses dan mengurangi proses login manakala ingin bertukar aplikasi selama sesi yang sedang berlangsung. Penggunaan metode autentikasi ini akan mengurangi beban penggunaan kata sandi yang berbeda untuk beberapa hak akses. Kemudahan melakukan akses melalui penggunaan akun tunggal perlu kehati-hatian untuk menjamin kerahasiaan proses pengesahan dan tidak tersebar ke pihak lainnya. Saat ini, ada beberapa aplikasi SSO yang bersifat terbuka, namun penggunaan metode pengesahan (autentikasi) yang ada masih rentan terhadap berbagai serangan terutama *Man-In-The-Middle Attack*. Pada makalah ini penggunaan metode autentikasi SSO menggunakan algoritme *One-Time Password* (OTP) diajukan dengan kombinasi sinkronisasi waktu dengan nilai *salt*. Kombinasi ini digunakan untuk memverifikasi proses sesi pengguna saat mengakses berbagai aplikasi pada mekanisme SSO. Hasil pengujian memperlihatkan algoritme OTP yang diajukan dapat menangani proses pengesahan SSO dengan baik dan juga dapat memproteksi serangan *Man-In-The-Middle Attack*.**

Kata Kunci*— *Single Sign-On*, *One-Time Password*, Autentikasi, Keamanan Jaringan, *Man-In-The-Middle Attack

[1]*Wireless and Networking Research Group (Winner), Jurusan Teknik Elektro, Jalan Tgk. Syech Abdurrauf No. 7, Darussalam, Banda Aceh 23111, Indonesia (tlp: 0651-7554336; fax: 0651-7552222; e-mail: rizal.munadi@unsyiah.ac.id)*

[2]*ProgramStudi Teknik Informatika, Fakultas Ilmu Komputer, Universitas Ubudiyah Indonesia, Jalan Teuku NyakArief, Banda Aceh 23111, Indonesia Darussalam, Banda Aceh 23111, Indonesia*

[3, 4, 5]*Jurusan Teknik Elektro, Jalan Tgk. Syech Abdurrauf No. 7, Darussalam, Banda Aceh 23111, Indonesia*

I. PENDAHULUAN

Sistem informasi merupakan sajian data yang dikumpulkan, dikelola yang dapat digunakan secara individual maupun komunitas/organisasi, terus berkembang dalam bentuk aplikasi *web-based* yang diakses melalui komputer desktop hingga aplikasi yang dapat diakses secara *mobile*, seperti *smartphone* atau *gadget* lainnya. Aplikasi yang dijalankan ada yang bersifat terbuka dan tertutup dengan menggunakan pembatasan hak akses dengan fitur penggunaan kata sandi (*password*). Untuk aplikasi yang terbuka dan tidak sensitif terhadap akses data, maka penggunaan kata sandi bukanlah merupakan bagian yang penting. Namun sebaliknya, untuk aplikasi yang memerlukan pengesahan (autentikasi) sehingga diperlukan kata sandi sebagai bagian dari proses verifikasi, memberikan pengaruh pada kenyamanan pengguna (*user*) yang harus mengingat banyak kata sandi untuk aplikasi yang berbeda.

Pada berbagai aplikasi, sistem pengesahan menjadi faktor utama yang mejadi pembatas akses *user*. *Single Sign-On* (SSO) merupakan salah satu model pengesahan independen yang memungkinkan pengguna dapat mengakses berbagai layanan aplikasi hanya dengan menggunakan satu *account* tunggal [1]. Akan tetapi, kemudahan akses melalui SSO memerlukan perhatian cermat untuk menjamin bukti-bukti pengesahan tidak mudah tersebar dan diketahui pihak lain.

Saat ini, terdapat beberapa metode autentikasi SSO *open source* yang tersedia, di antaranya seperti *Central Authentication Service (CAS)* dan *Security Assertion Markup Language (SAML)* [2]. Penelitian terkait SAML menjelaskan mengenai penggunaan enkripsi W3C XML sebagai metode autentikasi SSO. Namun penggunaan SAML dengan W3C XML memiliki celah keamanan yang berpotensi terhadap serangan *Cross Site Scripting (XSS)* dan *Cross Site Resource Forgery (CSRF)* [3]. XSS merupakan salah satu jenis serangan injeksi kode (*code injection attack*) yang dilakukan penyerang dengan cara memasukkan kode *Hyper Text Markup Language* (HTML) atau *client script code* lainnya ke suatu situs, sedangkan CSRF adalah jenis serangan eksploitasi satu *website* dengan perintah yang tidak sah dan dikirimkan dari pengguna [3].

Metode autentikasi lain yang telah tersedia yaitu menggunakan *Central Authentication Service (CAS)* menggunakan autentikasi berbasis *Lightweight Directory Access Protocol (LDAP)*. LDAP merupakan salah satu protokol *client-server* yang dapat digunakan untuk mengakses suatu *directory service*. Namun hasil penelitian terkait menunjukkan bahwa jalur komunikasi melalui protokol ini masih rentan serangan *Man-In-The-Middle (MITM)* [4].

Salah satu solusi pengamanan terhadap serangan MITM yang telah banyak diterapkan adalah melalui penggunaan protokol HTTPS yang menyediakan fitur enkripsi melalui *Secure Socket Layer (SSL)* atau *Transport Layer Security*

(TLS). Namun penelitian terkait menunjukkan penggunaan algoritme *Rivest Shamir Adleman (RSA)* pada HTTPS masih memiliki kerentanan untuk dipecahkan [5].

Paper ini mengusulkan metode pengesahan baru pada SSO dengan menggunakan algoritme *One-Time Password (OTP)* berbasis sinkronisasi nilai waktu. Pada imple-mentasinya, dalam algoritme yang diajukan ini penggunaan kombinasi sikronisasi waktu dan nilai *salt* ditujukan untuk membangkitkan *sesi_id user* secara acak dan hanya dapat digunakan untuk satu kali proses pengesahan sehingga dapat mengamankan dari penyalahgunaan akses dan dari kemungkinan aksi *intercept* atau serangan MITM. Penerapan metode ini dapat menjadi salah satu alternatif metode pengamanan autentikasi SSO publik *open source* yang sudah tersedia.

Bagian selanjutnya dari paper ini dapat dijelaskan sebagai berikut. Pada bagian II dijelaskan beberapa studi kepustakaan terkait sistem SSO dan algoritme *One-Time Password*. Pada bagian III diuraikan metode bahan, peralatan penelitian, serta prosedur yang digunakan dalam paper ini. Pada bagian IV dibahas hasil penelitian dan diskusi. Kemudian kesimpulan hasil penelitian akan disajikan pada bagian V.

II. SINGLE SIGN-ON, ONE-TIME PASSWORD, DAN *SALT*

A. *Single Sign-On (SSO)*

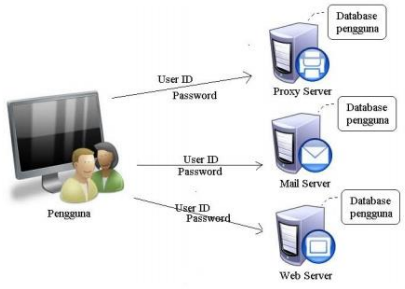
SSO merupakan sebuah sistem autentikasi yang mengizinkan pengguna mengakses berbagai aplikasi dengan menggunakan satu *credential* tanpa harus *login* di masing-masing aplikasi [1]. SSO memiliki dua bagian utama yaitu *single sign-on* di mana user hanya perlu *login* di satu aplikasi, maka aplikasi lain yang didefinisikan ikut dalam SSO otomatis akan dapat diakses, dan *single sign out* yaitu *user* hanya perlu *logout* di satu aplikasi, maka semua aplikasi yang didefinisikan ikut dalam SSO akan *logout* secara otomatis [1]. Sistem ini tidak memerlukan interaksi yang manual, sehingga memungkinkan pengguna melakukan proses sekali *login* untuk mengakses seluruh layanan aplikasi tanpa berulang kali menginputkan *password*-nya.

Teknologi SSO sangat diminati dalam jaringan yang sangat besar dan bersifat heterogen, di mana sistem operasi serta aplikasi yang digunakan berasal dari banyak vendor, dan pengguna diminta untuk mengisi informasi dirinya ke dalam setiap *multi-platform* yang hendak diakses. Perbedaan sistem SSO dengan sistem *login* biasa dengan memasukkan *user name* dan *password* secara berbeda-beda pada setiap sesi *login* dan sistem *single sign-on* dapat dilihat pada Gbr. 1 dan Gbr. 2.

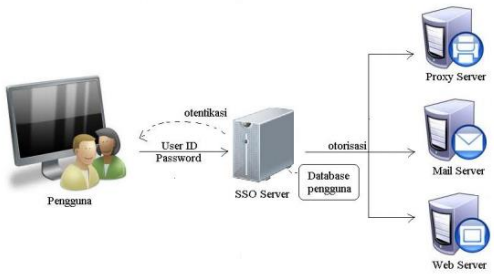
Arsitektur sistem SSO memiliki dua bagian utama yaitu *agent* dan *server* SSO [4]. Kedua bagian tersebut dapat dijelaskan sebagai berikut:

1) Agent: Permintaan setiap HTTP yang masuk ke *web server* akan diterjemahkan oleh *agent*. Di tiap-tiap *web server* ada satu *agent* sebagai *host* dari layanan aplikasi [4]. *Agent* ini akan berinteraksi pada *server* SSO dari sisi lain aplikasi dan berinteraksi dengan *web browser* dari sisi pengguna.

2) SSO server: Dalam menyediakan fungsi manajemen sesi *cookies temporer* (sementara) menggunakan *server* SSO di mana *user-id*, *session creation time*, *session expiration time* dan lain sebagainya adalah informasi ada pada *cookies* [4].



Gbr.1 Sistem *login* biasa [4].



Gbr. 2 Sistem *single sign-on* [4].

B. *One-Time Password (OTP)*

OTP merupakan metode autentikasi yang menggunakan *password* selalu berubah setelah setiap kali *login*, atau berubah setiap interval waktu tertentu [6]. OTP dapat dibedakan atas dua kategori utama sebagai berikut.

1) OTP Berbasiskan Algoritme Matematika: OTP jenis ini merupakan tipe lainnya dari OTP yang menggunakan algoritme matematika kompleks seperti fungsi hash kriptografi untuk membangkitkan *password* baru berdasarkan *password* sebelumnya dan dimulai dari kunci *shared* rahasia [6]. Contoh algoritme matematika yang digunakan dalam OTP ini adalah algoritme *open source* OATH yang telah distandarkan dan algoritme-algoritme lainnya yang telah dipatenkan. Beberapa produk aplikasi yang menggunakan autentikasi ini adalah sebagai berikut:

a) CRYPTOCard
   CRYPTOCard menghasilkan OTP baru setiap kali tombolnya ditekan. Sistem komputer akan menerima beberapa nilai balasan jika tombolnya ditekan lebih dari sekali secara tidak sengaja atau jika *client*-nya gagal mengautentikasi.
b) Verisign
   Verisign unified authentication menggunakan standar dari OATH.
c) E-token Aladdin *Knowledge System* NG-OTP
   E-token Aladdin *knowledge system* NG-OTP merupakan *hybrid* antara USB dan *token OTP*. *E-token Aladdin knowledge system NG-OTP* mengombinasikan fungsionalitas dari token autentikasi yang berbasiskan

smart card dan teknologi autentikasi *user one-time password* dalam mode terpisah.

2) OTP Berbasiskan Sinkronisasi Waktu: OTP jenis ini berbasiskan sinkronisasi waktu yang berubah secara konstan pada setiap satuan interval waktu tertentu [6]. Proses ini memerlukan sinkronisasi antara token milik *client* dengan server autentikasi. Pada jenis token yang terpisah (disebut dengan *disconnected token*), sinkronisasi waktu dilakukan sebelum token diberikan kepada *client* [6]. Tipe token lainnya melakukan sinkronisasi saat token dimasukkan dalam suatu alat *input*. Di dalam token terdapat sebuah jam akurat yang telah disinkronisasikan dengan waktu yang terdapat pada *server* autentikasi. Pada sistem OTP ini, waktu merupakan bagian yang penting dari algoritme kata sandi, karena pembangkitan kata sandi baru didasarkan pada waktu saat itu dan bukan pada kata sandi sebelumnya atau sebuah kunci rahasia.

Pada penelitian terkait [7], OTP jenis ini sudah mulai diimplementasikan terutama pada remote *Virtual Private Network* (VPN), dan keamanan jaringan Wi-Fi dan juga pada aplikasi berbagai aplikasi *Electronic Commerce* (*E-commerce*). Ukuran standar penggunaan waktu pada algoritme ini adalah 30 detik [7]. Nilai ini dipilih sebagai keseimbangan antara keamanan dan kegunaan.

C. Salt

Salt adalah data atau teks yang dipakai untuk menyulitkan penyerang *password* [8]. Biasanya *salt* digunakan pada proses algoritma *hash* untuk dimasukkan kedalam proses *hash* sebagai tambahan *input*. Hal ini menyebabkan nilai *hash* akan berubah jauh dari *hash* sebelumnya tanpa *salt*. *Salt* dapat dipilih tetap atau acak. Dengan *salt* maka penyerangan tidak dapat dilakukan secara paralel dengan *lookup password* dalam satu tabel, tapi penyerang harus terlebih dahulu membangkitkan tabel untuk tiap-tiap *salt*.

Selain menyulitkan penyerang kata sandi, penerapan *Salt* pada fungsi *hash* juga diklaim dapat menjaga karakter kata sandi yang disimpan karena akan selalu memiliki panjang karakter yang sama [9]. Namun pada kasus ini, *user* tidak akan dapat mengetahui teks asli saat kata sandinya terlupa karena algoritme *hash* yang digunakan bersifat satu arah.

III. METODOLOGI

A. Metode

Penelitian dalam paper ini menggunakan metode penelitian kuantitatif dengan menganalisis penerapan algoritma OTP sebagai metode autentikasi SSO.

B. Bahan

Pada penelitian ini digunakan beberapa perangkat lunak pendukung di antaranya:

- Notepad ++ sebagai perangkat lunak *editor* untuk memudahkan pengetikan kode program.
- Linux Apache MySQL dan PHP (LAMP) sebagai perangkat lunak untuk menjalankan aplikasi autentikasi SSO berbasis OTP.

C. Alat

Peralatan pendukung yang digunakan pada penelitian ini adalah seperangkat komputer dengan spesifikasi cukup untuk menjalankan *software* LAMP, Notepad++ dan Wireshark.

D. Prosedur Pengujian

Pada penelitian ini, pengujian dilakukan dengan melakukan autentikasi SSO berbasis OTP, pengujian *response time* dan pengujian kerentanan terhadap serangan MITM.

IV. PEMBAHASAN

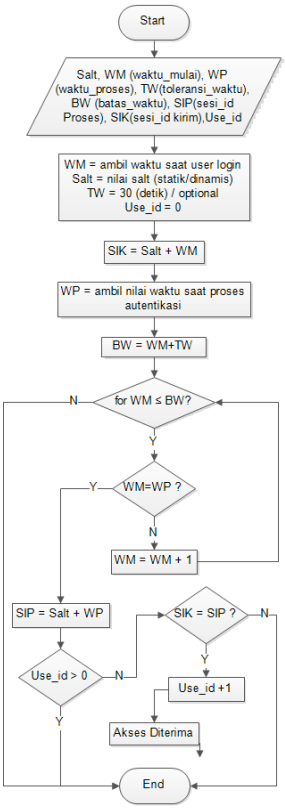
Bagian ini membahas mengenai penerapan OTP sebagai metode autentikasi SSO. Bagian yang akan dijelaskan terdiri atas perancangan *flowchart*, dan *pseudocode*.

Seperti yang telah dibahas pada bagian pendahuluan bahwa jenis OTP yang akan digunakan adalah OTP berbasis sinkronisasi waktu. Pada paper ini, OTP dikembangkan dengan kombinasi nilai *salt*. Penerapan kombinasi OTP dan *salt* ini dilakukan pada proses autentikasi di level aplikasi sehingga tidak mengganggu kenyamanan di sisi pengguna (*user*). Kombinasi OTP dan *salt* akan menghasilkan *sesi_id user* yang selalu berubah setiap proses autentikasi. Setiap aplikasi yang terhubung dengan SSO akan mendapatkan *nilai sesi_id* yang berbeda-beda. Tidak hanya itu, *sesi_id* yang dibangkitkan juga memiliki masa aktif tertentu. Setelah *sesi_id* berhasil terverifikasi, *sesi_id* tersebut tidak dapat digunakan kembali. Secara lebih detil, *flowchart* algoritme OTP dengan kombinasi nilai *salt* dapat dilihat pada Gbr. 3.

Proses pada pengesahan dimulai pada sesi *login user* SSO. Nilai *Salt*, dan nilai waktu mulai (WM) yaitu nilai waktu saat *user* mengakses aplikasi merupakan data masukan awal yang akan digunakan pada proses pembangkitan *sesi_id user*. Tahapan pertama yang dilakukan adalah pencampuran nilai *Salt* dan WM. Hasil pencampuran tersebut digunakan sebagai nilai *sesi_id* yang akan dikirim dari *server* melalui variabel *Sesi_send* kirim (SIK). Selanjutnya dilakukan penambahan nilai WM dengan nilai toleransi waktu (TW) untuk mendapatkan nilai batas waktu (BW). Nilai standar TW ditetapkan sebesar 30 detik, namun nilai ini bersifat *optional*. Penentuan ini merupakan nilai paling ideal dengan mempertimbangkan keamanan dan kegunaan. Ini juga bermakna nilai SIK hanya dapat digunakan dalam tenggang waktu 30 detik untuk satu kali proses autentikasi. Setelah itu, dilakukan pengecekan nilai BW terhadap nilai WP untuk memastikan proses pengecekan *sesi_id user* berada dalam *range* toleransi waktu yang telah tentukan.

Tahapan berikutnya beranjak ke proses *looping* nilai WM hingga memenuhi kondisi nilai BW untuk mendapatkan kecocokan nilai WM dengan WP. Setelah ditemukan kecocokan, dilakukan pencampuran nilai *Salt* dengan nilai WP untuk membangkitkan nilai *sesi_id* proses (SIP) yang akan dibandingkan dengan nilai SIK. Kemudian dilakukan verifikasi dengan mengecek nilai variabel *Use_id* tidak lebih besar dari 0 (nilai *default* 0) agar *sesi_id user* hanya dapat digunakan untuk satu kali proses autentikasi. Jika kondisi ini terpenuhi sistem akan memberikan nilai umpan balik ke variabel *Use_id*. Namun jika tidak, maka proses akan

dihentikan. Tahapan akhir adalah proses pencocokan nilai SIK dengan SIP. Jika terpenuhi maka pengecekan *sesi_id user* berhasil dan sistem memberikan izin akses.



Gbr. 3 Flowchart algoritme OTP dengan kombinasi nilai *salt*.

V. HASIL PENGUJIAN

Pada bagian ini akan dipaparkan mengenai hasil penerapan algoritme OTP pada autentikasi SSO. Pada bagian ini, algoritme OTP dengan kombinasi *salt* ditampilkan pada Gbr. 4. Proses pengujian autentikasi SSO dengan OTP, pengamatan *response time*, dan pengujian terhadap serangan MITM diuraikan dalam bagian ini.

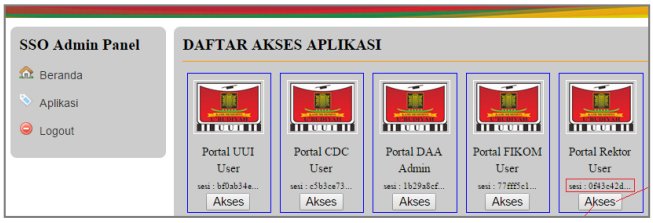
Algoritma One-Time Password SSOdengan Kombinasi Salt

```
1: Input:  Salt, WM (waktu_mulai), WP(waktu_proses), BW(batas_waktu), TW (toleransi_waktu),
   SIK(Sesi_id Kirim), SIP(Sesi_id Proses), Use_id
2: WM = ambil waktu saat user login
3: Salt = nilai salt (statik/dinamis)
4: TW = 30 (detik)  / optional
5: Use_id = 0
6: Output: SI (Sesi_id) {Akses Diterima / Ditolak}
7: SIK= Salt + WM
8: WP = ambil nilai waktu saat proses autentikasi
9: BW = WM+TW
10: while (WM  ≤ BW) do
11: if WM = WP
12: SIP = Salt + WP
13: if Use_id ≤ 0 and SIP = SI
14: Use_id++
15: out: Sesi_id valid akses diterima
16: end if
17: end if
18: WM = WM+1
19: end while
```

Gbr. 4 *Pseudocode* algoritme *One-Time Password* dengan kombinasi *salt.*

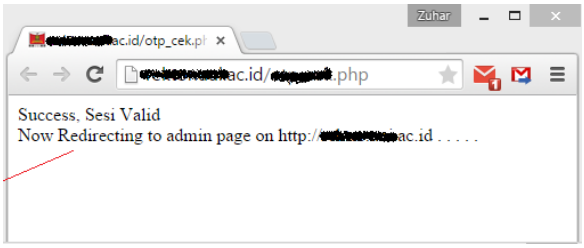
A. Pengujian Autentikasi SSO dengan OTP

Pada Gbr. 5 ditunjukkan percobaan akses yang dilakukan terhadap salah satu aplikasi yang terhubung dengan SSO. Pada percobaan ini, dilakukan kustomisasi *interface* SSO untuk dapat menampilkan kode *sesi_id user* pada masing-masing aplikasi.



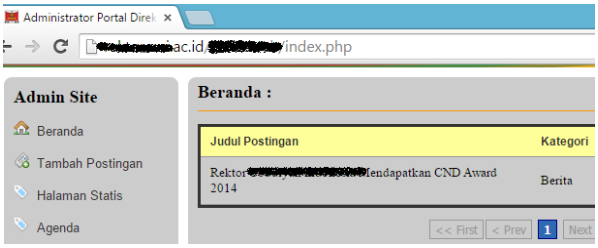
Gbr. 5 Percobaan autentikasi SSO.

Pengujian pada Gbr. 5 memperlihatkan SSO menggunakan autentikasi OTP dengan kombinasi nilai *salt* menghasilkan kode autentikasi *user* yang dinamis (masing-masing aplikasi mempunyai kode *sesi_id* yang berbeda). Hasil proses autentikasi di atas dapat dilihat pada Gbr. 6.



Gbr. 6 Proses autentikasi SSO dengan OTP.

Gbr. 6 memperlihatkan proses autentikasi SSO berhasil dilakukan. Selanjutnya *user* diarahkan ke halaman *admin* dari aplikasi yang diakses seperti diperlihatkan Gbr. 7.



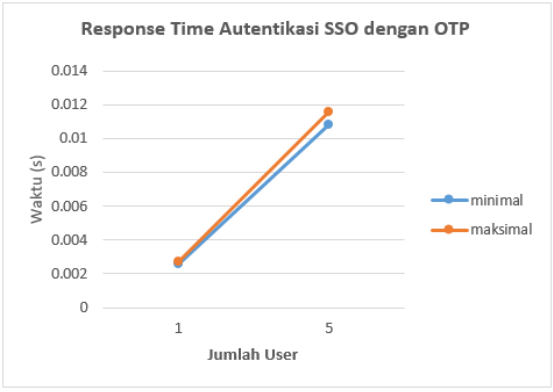
Gbr. 7 Halaman admin aplikasi.

B. Pengujian Response Time

Hasil percobaan pada Gbr. 7 menunjukkan proses autentikasi SSO menggunakan algoritme OTP berhasil dilakukan.

Pengujian *response time* dilakukan dengan mengukur waktu komputasi proses autentikasi SSO dengan OTP. Untuk mengukur waktu komputasi secara akurat digunakan fungsi *microtime* yang ada pada bahasa pemrograman PHP. Pengujian ini terdiri atas dua pengujian utama. Pengujian pertama dilakukan percobaan autentikasi sebanyak 25 kali untuk satu *account user* yang telah diberi hak akses. Penentuan ukuran *sample* pengujian ini berdasarkan jumlah *sample* minimum yaitu 15 untuk penelitian eksperimen [10]. Selanjutnya untuk melihat perbandingan *response time* dari

pengujian pertama, pengujian kedua dilakukan menggunakan lima *account user* yang melakukan autentikasi pada waktu yang bersamaan. Hasil pengujian ini dapat lihat pada Gbr. 8.

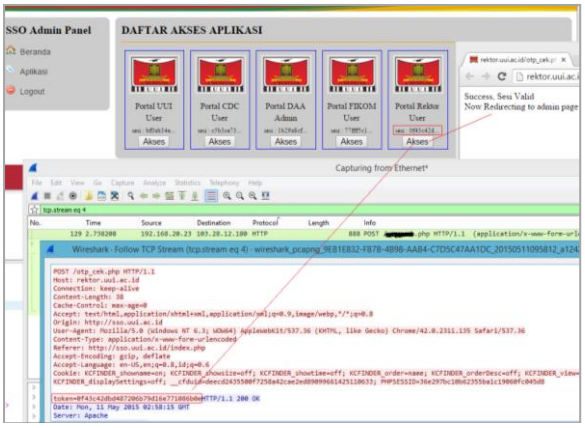


Gbr. 8 *Response time* autentikasi SSO dengan OTP.

Berdasarkan data hasil pengujian pada Gbr. 8, nilai *response time* rata-rata autentikasi SSO pada pengamatan dengan satu *user* sebesar 0,0052s atau 15,31% lebih cepat dibandingkan dengan *response time* rata-rata lima *user* dengan nilai 0,0061s.

C. Pengujian Serangan Man-In-The-Middle Attack (MITM)

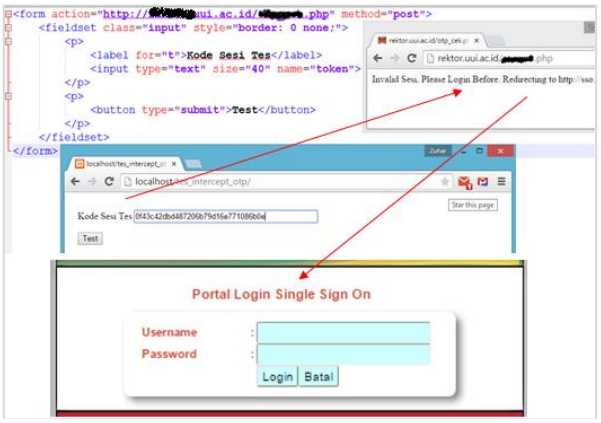
Pengujian ini dilakukan dengan menganalisis paket data saat autentikasi SSO berlangsung. Analisis ini dilakukan menggunakan *software wireshark* untuk mendapatkan *sesi_id user*. Hasil percobaan ini dapat dilihat pada Gbr. 9.



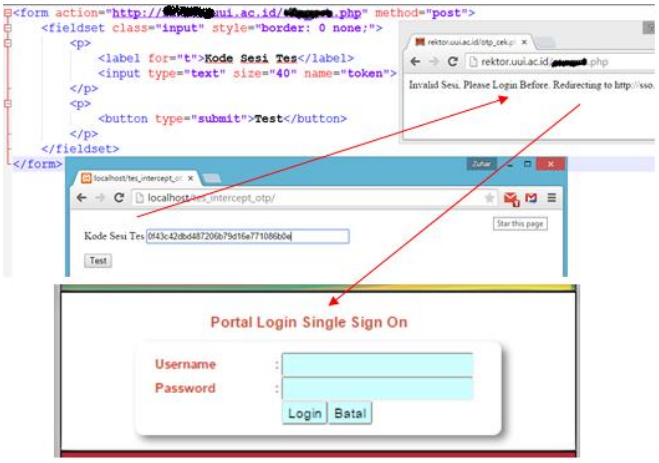
Gbr. 9 Pengujian MITM pada autentikasi SSO.

Pada Gbr. 10 diperlihatkan serangan MITM berhasil menangkap *sesi_id user* saat proses autentikasi SSO berlangsung. Pada tahapan selanjutnya dilakukan percobaan *intercept* menggunakan *sesi_id* yang didapatkan pada Gbr. 9. Percobaan ini dilakukan dengan mengirim nilai kode *sesi_id* melalui sebuah *form* menggunakan *method _POST* dengan target *action form* menuju ke *file* autentikasi pada aplikasi yang diakses. Hasil percobaan ini ditunjukkan pada Gbr. 11.

Seperti ditunjukkan pada Gbr. 11, proses *intercept* gagal dilakukan karena kode *sesi_id user* yang dibangkitkan melalui OTP hanya dapat digunakan untuk satu kali proses autentikasi.



Gbr. 10 *Response time* autentikasi SSO dengan OTP.



Gbr. 11 Pengujian *intercept* autentikasi SSO.

VI. KESIMPULAN

Hasil penelitian ini menunjukkan algoritme OTP dapat digunakan sebagai metode autentikasi SSO dan dapat menjadi salah satu alternatif metode pengamanan autentikasi SSO publik *open source* yang sudah tersedia. Dari sisi keamanan, penerapan OTP dapat memproteksi serangan MITM karena *sesi_id user* yang dibangkitkan hanya dapat digunakan untuk satu kali proses autentikasi.

REFERENSI

[1] K.D. Lewis, "Web Single Sign-On Authentication using SAML," *International Journal of Computer Science Issues* (IJCSI), Vol. 2, Aug. 2009.

[2] S. Lawton. (2015, Jan.). Secure Authentication With Single Sign-On (SSO) Solutions. Tom's IT Pro, California, USA. [Online]. Available : http://www.tomsitpro.com/articles/single-sign-on-solutions,2-853.html

[3] P. Telnoni, R.Munir, Y. Rosmansyah, "Pengembangan Protokol Single Sign-On SAML Dengan Kombinasi Speech dan Speaker Recognition," *Jurnal Cybermatika ITB*, Vol. 2, Dec. 2014

[4] G. Ramadhan, "Analisis teknologi Single Sign On (SSO) dengan penerapan Central Authentication Service (CAS) pada Universitas Bina Darma," Skripsi, Lab. Komputer, UBD, Palembang, Indonesia, 2012.

[5] J. Kirk (2007, Mei.). Researcher: RSA 1024-bit Encryption Not Enough. IDG Consumer & SMB, San Francisco, USA. [Online]. Available: http://www.pcworld.com/article/132184/article.html

[6] Hyun-Chul Kim; Lee, H.-W.; Young-Gu Lee; Moon-Seog Jun, "A Design of One-Time Password Mechanism Using Public Key

Infrastructure," *Fourth International of Networked Computing and Advanced Information Management*, Sep. 2008

[7] D. M'Raihi, S. Machani, M. Pei, J. Rydell. (2011, Mei.). TOTP: Time-Based One-Time Password Algorithm. The Internet Engineering Task Force (IETF), California, USA. [Online]. Available: https://tools.ietf.org/html/rfc6238

[8] Gauravaram, P., "Security Analysis of salt || password Hashes," *International Conference Advanced Computer Science Applications and Technologies* (ACSAT), 26-28 Nov. 2012.

[9] P. Ducklin. (2013, Nov.). Anatomy of a password disaster - Adobe's giant-sized cryptographic blunder. Sophos Ltd, Boston, USA. [Online]. Available:nakedsecurity.sophos.com/2013/11/04/anatomy-of-a-password-disaster-adobes-giant-sized-cryptographic-blunder/

[10] Roscoe, J.T, Fundamental Research Statistics for the Behavioural Sciences 2nd edition, New York, USA: Holt Rinehart & Winston, 1975.