TUGAS AKHIR – TE 141599

ANALISIS POTENSI CELAH KEAMANAN PADA *WEBSITE SINGLE SIGN ON* YANG MENGGUNAKAN PROTOKOL *OPENID*

Akhirul Hajri
NRP 2210100093

Dosen Pembimbing
Christyowidiasmoro, ST.,MT
Reza Fuad Rachmadi, ST., MT

JURUSAN TEKNIK ELEKTRO
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2015

TUGAS AKHIR – TE 141599

ANALISIS POTENSI CELAH KEAMANAN PADA *WEBSITE SINGLE SIGN ON* YANG MENGGUNAKAN PROTOKOL *OPENID*

Akhirul Hajri
NRP 2210100093

Dosen Pembimbing
Christyowidiasmoro, ST.,MT
Reza Fuad Rachmadi, ST., MT

JURUSAN TEKNIK ELEKTRO
Fakultas Teknologi Industri
Institut Teknologi Sepuluh Nopember
Surabaya 2015

FINAL PROJECT – TE 141599

VULNERABILITY ANALISYS ON SINGLE SIGN ON WEBSITE THAT USE OPENID PROTOCOL

Akhirul Hajri
NRP 2210100093

Advisors
Christyowidiasmoro, ST.,MT
Reza Fuad Rachmadi, ST., MT

Departement of Electrical Engineering
Faculty of Industrial Technology
Sepuluh Nopember Institute of Technology
Surabaya 2015

ANALISIS POTENSI CELAH KEAMANAN PADA *WEBSITE SINGLE SIGN ON* YANG MENGGUNAKAN PROTOKOL *OPENID*

TUGAS AKHIR

Diajukan untuk Memenuhi Sebagian Persyaratan
Untuk Memperoleh Gelar Sarjana Teknik
Pada
Bidang Studi Teknik Komputer dan Telematika
Jurusan Teknik Elektro
Institut Teknologi Sepuluh Nopember

Menyetujui

Dosen Pembimbing I	Dosen Pembimbing II
Christowidiasmoro, ST., MT	Reza Fuad R., ST.,MT
NIP. 19830127 2009121004	NIP 19850403 2012121001

SURABAYA
JULI 2015

ABSTRAK

Single sign on adalah metode yang memungkinkan pengguna hanya perlu melakukan satu kali proses otentikasi untuk menggunakan beberapa layanan. Salah satu protokol dalam *single sign on* adalah protokol *OpenID*. Proses otentikasi pada *OpenID* yang menggunakan satu akun untuk beberapa *website*, serta tanpa menggunakan sebuah kode sandi dan proses otentikasi berada diluar sistem/*website* penyedia layanan, timbul pertanyaan bagaimana tingkat keamanan pada protokol *OpenID* ini Untuk itu dilakukan pengujian keamanan pada protokol *OpenID*.Tugas Akhir ini bertujuan untuk menguji dan menemukan potensi celah keamanan yang terdapat pada protokol *OpenID* serta menemukan solusi untuk mengatasi dan menghindarkan pengguna dari bahaya celah keamanan tersebut. Berdasarkan pengujian yang telah dilakukan proses pengalihan memungkinkan pengguna menjadi korban *Phishing*. Cara terbaik untuk mencegah tindakan *phishing* adalah dengan cara tidak menampilkan halaman *login* pada proses pengalihan jika pengguna belum diotentikasi di Provider. Selain itu ketika akun pengguna di *Relying Party* dicuri, maka berpotensi kode token yang tersimpan pada *Relying Party* juga dapat dicuri dan ini akan membahayakan data-data pengguna di Provider. Untuk mengatasi celah keamanan ini, Provider harus membuat sebuah regulasi untuk membatasi *website* yang bisa menggunakan protokol *OpenID*.

Kata kunci: *Single sign on, OpenID, phishing,* kode token,dan keamanan data.

(Halaman ini sengaja dikosongkan)

ABSTRACT

Single sign on is a method which allow users to gain access to several services with just one authentication. One of the protocols in a single sign-on is the OpenID protocol. Authentication process in OpenID use one account for multiple websites, and without using a password and authentication processes are outside the system / website Relying Party. Raises the question of how the level of security in this OpenID protocol. Therefore dilakukan penetration testing to determine the quality of security in the OpenID protocol. The purpose of this penetration testing is to find the vulnerability in OpenID Protocol and find the solution to fix that and prevent users from the dangers of the vulnerability. Based on the penetration testing that was done the process of allowing users become victims of Phishing. The best way to prevent phishing action is to not display the login page in the transfer process if the user has not been authenticated in Provider. Additionally, when a user account on Relying Party is stolen, then potentially token code stored on a Relying Party can also be stolen and this will harm the user data in Provider. To resolve this vulnerability, Providers must make a regulation to limit the websites that can use the OpenID protocol.

***Keywords**: Single Sign On, OpenID, phishing, session hijacking dan data security.*

(Halaman ini sengaja dikosongkan)

KATA PENGANTAR

Puji syukur kehadirat ALLAH SWT atas segala rahmat dan karunia-Nya, penulis dapat menyelesaikan penelitian ini dengan judul : **Analisis Potensi Celah Keamanan pada *Website Single Sign On* yang menggunakan Protokol *OpenID***

Penelitian ini disusun dalam rangka pemenuhan bidang riset di Jurusan Teknik Elektro ITS, Bidang Studi Teknik Komputer dan Telematika, serta digunakan sebagai persyaratan menyelesaikan pendidikan S-1. Penelitian ini dapat terselesaikan tidak lepas dari bantuan berbagai pihak. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. Ayah dan ibu yang sangat banyak membantu dalam semua hal, terutama dalam hal spiritual dan material. Karena berkat doa dari kedua orang tualah semua proses penelitian tugas akhir ini dapat berjalan dengan lancar.
2. Bapak Dr. Tri Arief Sardjono, ST., MT. selaku Ketua Jurusan Teknik Elektro, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember.
3. Bapak Christyowidiasmoro, ST., MT dan Bapak Reza Fuad Rachmadi, ST., MT, selaku dosen pembimbing yang selalu memberikan saran serta bantuan dalam penelitian ini.
4. Bapak Ibu dosen pengajar Jurusan Teknik Elektro,
5. Seluruh teman-teman asisten bidang Studi Teknik Komputer dan Telematika Teknik Elektro ITS.

Kesmpurnaan hanya milik ALLAH SWT, untuk itu penulis memohon segenap kritik dan saran yang membangun serta meminta maaf atas segala kekurangan yang ada dalam penulisan buku ini. Semoga penelitian ini dapat memberikan manfaat bagi kita semua. Amin.

Surabaya. Juli 2015

Penulis

(Halaman ini sengaja dikosongkan)

DAFTAR ISI

Daftar Tabel

(Halaman ini sengaja dikosongkan)

Daftar Gambar

Daftar Kode

(Halaman ini sengaja dikosongkan)

BIOGRAFI PENULIS

Akhirul Hajri lahir di Lubuk Jambi pada tanggal 30 Juni 1992. Menyelesaikan pendidikan SD di SDN 001 Pasar Lubuk Jambi pada tahun 2004, kemudian melanjutkan pendidikan SMP di SMPN1 Kuantan Mudik dan lulus pada tahun 2007 dan menyelesaikan pendidikan SMA di SMAN Plus Provinsi Riau pada tahun 2010. Setelah lulus SMA, penulis melanjutkan pendidikan S1 di jurusan Teknik Elektro Institut Teknologi Sepuluh Nopember (ITS). Penulis sangat tertarik dengan jaringan komputer dan pernah mengikuti pelatihan serta sertifikasi jaringan komputer yang diadakan oleh salah satu perusahaan telekomunikasi PT Huawei, yaitu Huawei Ceritified Datacom Associate (HCDA) pada tahun 2014.

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Single *Sign On* merupakan sebuah metode dimana pengguna hanya sekali melakukan *login* untuk dapat mengakses *network resources* yang ada. Metode *single sign on* menggunakan pihak ketiga sebagai pengelola *username* dan *password* dari pengguna aplikasi. Pada dasarnya otentikasi *single sign on* hanya melibatkan satu *credential user* untuk *login* ke banyak aplikasi berbasis *website* setelah sebelumnya melakukan otentikasi pada salah satu aplikasi yang terintegrasi dengan sistem *Single Sign On*.

Salah satu protokol yang digunakan dalam *single sign on* adalah protokol *OpenID*. Pada *OpenID* ini terdapat 3 pihak yang terlibat yaitu pengguna, pihak client atau lebih dikenal sebagai *Relying Party* dan pihak *provider* sebagai otentikator/penyedia jasa otentikasi dan penyimpan data pengguna. Proses otentikasi protokol *OpenID* menggunakan jasa pihak ketiga dan berada diluar sistem penyedia layanan. Jadi ketika proses otentikasi, pengguna akan dialihkan menuju sistem *provider* untuk diotentikasi. Ketika pengguna telah diotentikasi, *provider* akan memberitahukan kepada *Relying Party* bahwa pengguna telah diotentikasi dan bisa menggunakan layanannya.

Pada protokol *OpenID* terjalin kerja sama antara *Relying Party* dan *Provider* dalam hal identitas dan data-data pengguna. dari kerja sama yang terjalin antara *provider* dan *Relying Party*, *Relying Party* akan mendapatkan hak untuk mengakses data-data pengguna yang ada di *provider*. Dan untuk keamanan dan kenyamanan pengguna, seharusnya tidak semua data-data pengguna bisa diakses oleh *Relying Party*, harus ada regulasi untuk membatasi *Relying Party* untuk mengakses data pengguna.

Karena protokol *OpenID* ini adalah protokol untuk proses otentikasi, faktor keamanana adalah hal yang paling utama dalam proses otentikasi ini. Data-data untuk proses otentikasi harus dalam kondisi aman dan tidak boleh diketahui orang lain. Untuk itu dibutuhkan kredibilitas yang tinggi untuk bisa menjalin kerja sama antara *Relying*

Party dan *Provider*. Serta jaminan sistem yang benar-benar aman, dikarenakan hal ini berkaitan dengan keamanan data pengguna. Selain itu, proses otentikasi pada *OpenID* yang menggunakan 1 akun untuk beberapa website, serta tanpa menggunakan sebuah kode sandi dan proses otentikasi berada diluar sistem/*website* penyedia layanan. menimbulkan pertanyaan bagaimana tingkat keamanan pada protokol *OpenID* ini. Untuk itu dilakukan pengujian tingkat keamanan dan analisa mendalam pada protokol *OpenID* untuk mencari tahu potensi celah keamanan yang bisa terjadi. Dengan mengetahui potensi yang terdapat dalam protokol *OpenID* kemudian dapat dicari solusi untuk mengatasi celah keamanan tersebut. Dan diharapkan pengguna aplikasi dapat terhindar dari tindakan kriminal (*cyber crime*) dan berbagai tindakan *hacking* lainnya.

1.2 Permasalahan

Pengunaan protokol *HTTP (Hyper Text Transfer protocol)* dalam proses otentikasi berpotensi menimbulkan celah keamanan yang besar. Potensi celah ini akan menimbulkan celah-celah keamanan yang lain. Akibat dari penggunaan protokol *HTTP*, dan tidak menggunakan saluran yang aman seperti *SSL (Secure Socket Layer)*, semua data otentikasi sangat terbuka dan tidak terenkripsi. Dan ini sangat berbahaya, karena orang lain dapat melihat parameter-parameter yang digunakan dalam proses otentikasi. Dan parameter-parameter tersebut dapat dimanfaatkan untuk membajak akun dari pengguna dengan berbagai metode *hacking*, seperti *session hijacking, parameter injection*, dan lain sebagainya.

Tidak hanya itu, proses *redirect*/pengalihan ke pihak *provider* juga mempunyai kerentanan terhadap tindakan *phishing*, yaitu dengan memalsukan halaman pihak *provider*. *Phishing* tentu sangat berbahaya karena jika pengguna terjebak didalamnya, maka peretas akan mendapatkan *username* dan *password* pengguna aplikasi.

Jika akun pengguna di *Relying Party* diretas orang lain akan berpotensi menimbulkan bahaya baru, yaitu ada potensi pencurian kode token yang diberikan *provider* kepada *Relying Party*. Dimana kode token ini berfungsi sebagai alat untuk mendapatkan hak untuk mengakses data pengguna di *provider*.

Setiap *provider* mempunyai regulasi sendiri tentang hak akses yang diberikan kepada *Relying Party*. Kelemahan *provider* bisa mengakibatkan ternjadinya pembobolan hak akses yang seharusnya tidak dimiliki oleh *Relying Party* menjadi bisa di akses.

1.3 Tujuan

Penelitian tugas akhir ini bertujuan menganilisis potensi celah keamanan pada aplikasi website yang menggunakan metode SSO, khususnya yang menggunakan protokol *OpenID*. Dengan ditemukannya celah keamanan yang terdapat pada penggunaan protokol OpenID ini, dapat dicari solusi untuk mengatasinya. Dengan begitu diharapkan dapat meminimalisir tindakan kriminal (*cyber crime*) dan tindakan pencurian akun lainnya, serta diharapkan juga dapat membantu administrator *website* mengetahui celah keamanan pada sistem yang dikelolanya dan kemudian meningkatkan kualitas keamanan pada sistemnya.

1.4 Batasan Masalah

Batasan masalah dalam pengerjaam tugas akhir ini adalah proses analisis yang hanya terbatas pada:

1. *Website single sign on* yang menggunakan protkol *OpenID*
2. Hasil pengujian merupakan hasil yang didapat selama penelitian tugas akhir ini, jadi jika setelah laporan Tugas Akhir ini hasil pengujian tidak berfungsi lagi, kemungkinan sistem telah diperbaiki dan kualitas keamanan sistem telah ditingkatkan
3. Sampel yang digunakan untuk objek pengujian hanya untuk *provider* besar, seperti Google, Facebook dan Wordpress.

1.5 Sistematika Penulisan

Sistematika penulisan laporan Tugas Akhir ini dibagi dalam 4 bab, masing-masing bab diuraikan sebagai berikut:

1.Bab 1

Merupakan pendahuluan yang berisi latar belakang, tujuan permasalahan, batasan masalah, metodelogi dan sistematika penulisan.

2.Bab 2

Teori penunjang dan kepustakan yang digunakan dalam proses tugas akhir ini.

3.Bab 3

Desain sistem dan sistematika pengujian keamananan.

4.Bab 4

Hasil analisa dan pengujian keamanan sistem serta solusi untuk mengatasi permasalahan keamanan pada sistem.

BAB 2

DASAR TEORI

2.1 Otentikasi

Otentikasi adalah sebuah metode untuk memverifikasi seorang pengguna yang mempunyai hak untuk mengakses suatu sistem terntentu. Proses otentikasi dibutuhkan untuk memperkuat keamanan sistem, untuk itu ada beberapa faktor dalam proses otentikasi seorang pengguna, yaitu:

a. Sesuatu yang diketahui oleh pengguna, misalnya seperti kata sandi dan kode *PIN(Personal Idetification Number).*
b. Sesuatu yang dimiliki pengguna, misalnya seperti kode token atau sebuah sertifikat perangkat lunak.
c. Sesuatu yang ada pada diri pengguna, misalnya seperti sidik jari atau retina dari pengguna.

Jika semua faktor tersebut diterapkan dengan benar, maka seorang peretas akan lebih sulit untuk melakukang penyerangan/peretasan terhadap suatu sistem, karena terdapat satu faktor yang tidak dimiliki oleh seorang peretas, seperti sesuatu yang ada pada diri pengguna, yaitu sidik jari. Dengan demikian informasi rahasia mennjadi lebih aman.

Ada beberapa kombinasi factor-faktor dalam proses otentikasi yang biasa digunakan untuk meningkatkan kualitas keamanan sistem, seperti :

a. Sebuah kartu pintar yang digunakan bersama dengan pembaca kartu pintar (*smartcard reader*) dan kode *PIN*.
b. Sidik jari yang digunakan bersama kata sandi atau kode *PIN*
c. Sertifikat perangkat lunak yang tersimpan dikomputer yang hanya dapat diakses dengan sebuah kata sandi atau kode *PIN*. [11]

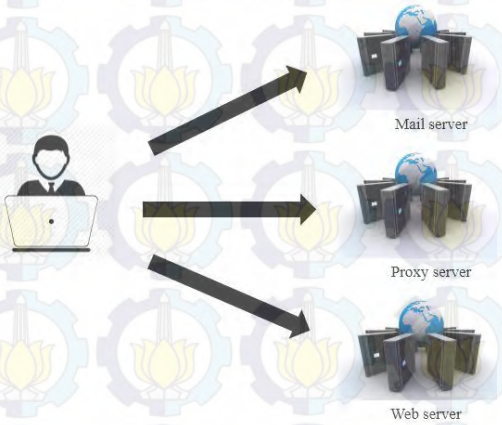
Semakin banyak kombinasi faktor yang digunakan akan membuat peluang peretas untuk melakukan pencurian data semakin kecil.

2.2 Single sign on

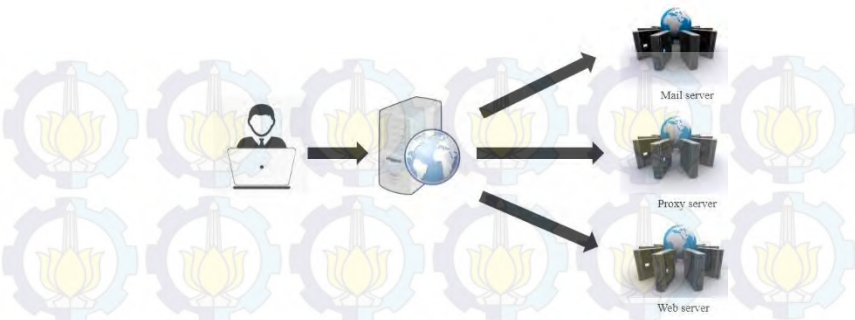
Single sign on adalah sebuah metode yang memungkinkan pengguna dapat menggunakan beberapa layanan sekaligus hanya dengan mengguna satu *username*/*userid* dan *password* saja. Pada *single sign on* ini, proses otentikasi hanya sekali, kemudian proses otentikasi akan dilakukan dengan otomatis ketika pengguna membuat layanan lain melalui sebuah *session* [1]

Jadi, jika pengguna menggunakan metode *single sign on* pada *website*, maka pengguna hanya perlu melakukan satu kali *login* dan kemudian pengguna dapat menggunakan beberapa layanan yang lain tanpa perlu memasukan *password* lagi.

Untuk perbandingan antara metode *single sign on* dengan metode *sign on* dapat dilihat pada gambar 2.1 dan gambar 2.2



Gambar 2.1 Metode *Sign On*



Gambar 2.2 Metode *Single sign on*

2.3 *OpenID*

OpenID adalah sebuah protokol terbuka yang memungkin pengguna untuk dapat menggunakan akun yang telah ada untuk *login* pada beberapa *website* tanpa perlu membuat *password* baru.[13] Jadi, dengan *OpenID*, pengguna dapat *login* pada *website* lain dengan akun yahoo miliknya. [12]

Pada protokol *OpenID* ini, terdapat 3 pihak yang terlibat, yaitu:

a. *Identity Provider*
   Identity *provider* adalah *website* yang bertugas sebagai penyedia identitas pengguna, misalnya Google, Yahoo, Facebook, dan Twitter.
b. *Relying Party*
   Relying Party adalah *website* yang menggunakan jasa identity *provider*.
c. Pengguna

Pengguna dapat memilih informasi apa saja yang akan diberikan atau informasi apa saja yang dapat diakses oleh *website* yang dikunjugi oleh pengguna. Sedangkan untuk *password* pengguna hanya diketahui oleh *provider*, dan *provider* yang mengelola identitas dari pengguna, selain itu tidak *website* yang mengetahui *password* pengguna. [13]

Jadi, dengan *OpenID* ini, pengguna dapat *login* pada suatu *website* dengan menggunakan akun di*website* lain. Misalnya, pengguna A dapat *login website* livejournal.com dengan menggunakan akun Google yang

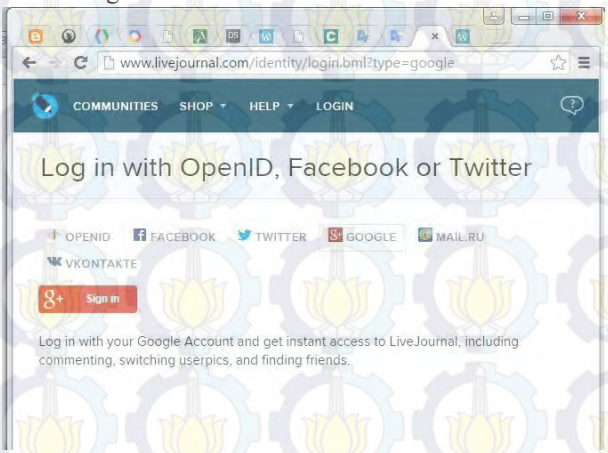
dimilikinya. Tapi dalam hal ini, *website* livejournal.com hanya mendapatkan beberapa hak untuk mengakses data pengguna dan hak tersebut harus sudah disetujui oleh pengguna A. livejournal.com tidak mengetahui *password* akun Google pengguna A. Dengan metode *login* seperti ini, seorang pengguna tidak perlu lagi mengingat banyak *userid* dan *password* untuk *login* pada beberapa *website* yang berbeda.

Saat ini *OpenID* memiliki lebih dari satu miliar pengguna *OpenID* dan lebih dari 50.000 *website* menggunakan *OpenID* untuk *login*. Beberapa perusahaan besar menggunakan *OpenID*, termasuk Google, Facebook, Yahoo !, Microsoft, AOL, MySpace, Sears, Universal Music Group, France Telecom, Novell, Sun, Telecom Italia, dan masih banyak lagi. [13]

2.3.1 Prosedur Penggunaan Protokol *OpenID*

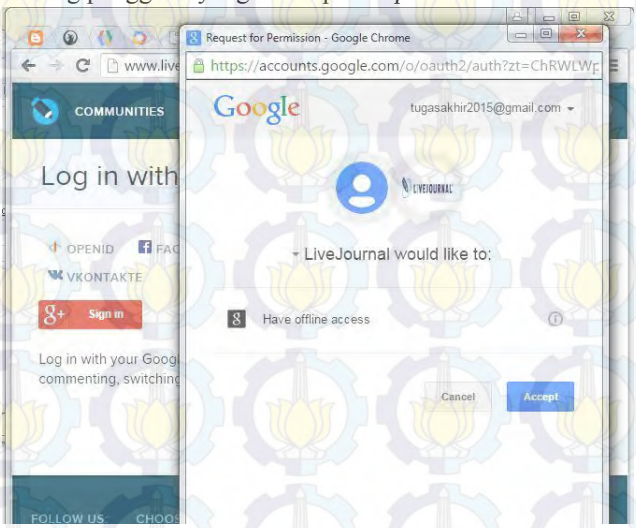
Berikut adalah proses ketika pengguna *login* pada *Website* livejournal.com dengan menggunakan *OpenID* (menggunakan *provider* Google dan wordpress)

1. Menggunakan Google
   a. Pengguna memilih *login* ke *Website* livejournal.com dengan akun Google+.



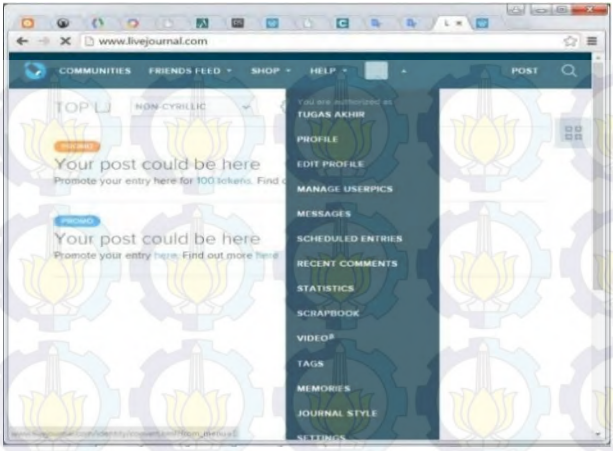
Gambar 2.3 Tombol *Login* Dengan Goole

b. Kemudian pengguna akan dialihkan ke *website* Google untuk otentikasi, serta meminta persetujuan pengguna bahwa *website* livejournal.com akan mengakses informasi tentang pengguna yang tersimpan di *provider*.



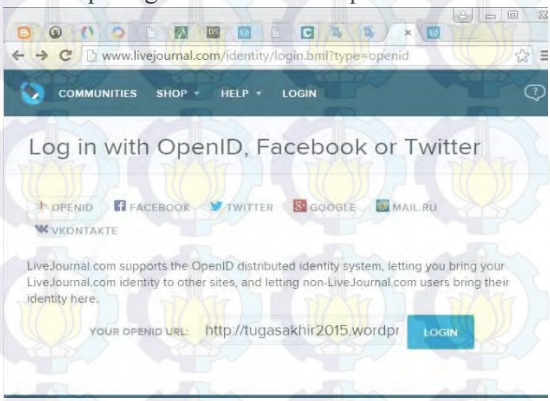
Gambar 2.4 Permintaan izin mengakses data pengguna

c. Setelah pengguna memberikan persetujuan kepada *Relying Party*, selanjutnya pengguna bisa menggunakan layanan pada *Website* livejournal.com



Gambar 2.5 Pengguna Berhasil *login* dengan Google

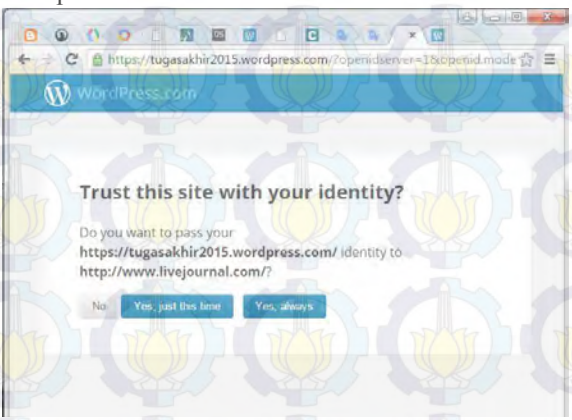
2. Menggunakan url wordpress.
   a. Pengguna memilih *login* ke *Website* livejournal.com dengan *URL* http://tugasakhir2015.wordpress.com



Gambar 2.6 Kolom *login* dengan Wordpress

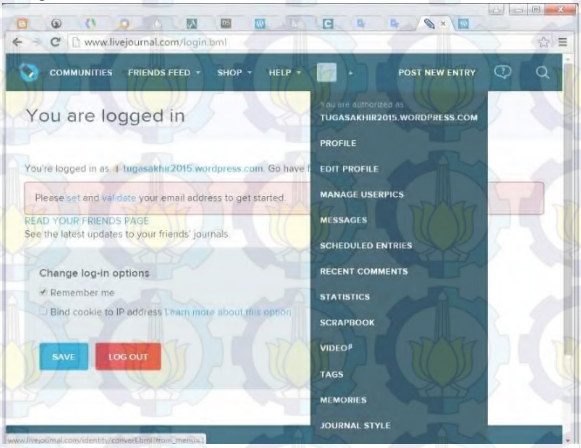
   b. Kemudian, pengguna akan dialihkan ke *website* wordpress.com untuk otentikasi dan meminta persetujuan pengguna bahwa *website* livejournal.com akan mengakses

informasi tentang pengguna yang tersimpan di wordpress.com



Gambar 2.7 Verifikasi data pengguna

c. Setelah itu, pengguna akan dialihkan kembali ke *website* livejournal.com



Gambar 2.8 Pengguna Berhasil *Login* dengan Wordpress

2.3.2 Proses Otentikasi dan Cara Kerja Protokol *OpenID*

Adapun prinsip kerja *OpenID* secara teknis dan proses otentikasi di lakukan adalah sebagai berikut:

a. Pengguna memasukan *URL OpenID* pada kolom yang tersedia (misalnya: http://alice.myopenid.com) di *website Relying Party*.
b. Untuk mengetahui *Provider* yang digunakan oleh pengguna, *Relying Party* menggunakan *URL* yang dikirimkan oleh Pengguna. Pada umumnya *URL identifier OpenID* terletak pada bagian <head> dari kode html *Website Relying Party*, lebih spesifiknya terletak pada tag <*link*>. Didalam tag <*link*> ini terdapat atribut rel="`openid2.provider`" serta link dari *OpenID provider* yang digunakan.
c. Setelah *Relying Party* mengetahui *OpenID provider* yang digunakan, maka *Relying Party* akan membuat sebuah prosedur keamanan dengan *OpenID Provider* menggunakan algoritma Diffie-Hellman yaitu dengan membuat *security key*. *Security key* ini akan digunakan untuk memverifikasi dalam pertukaran data antara keduanya.
d. Kemudian *Relying Party* akan mengalihkan pengguna ke *Website OpenID* menggunakan sebuah *URL identifier* yang terdiri dari semua parameter yang dibutuhkan untuk proses otentikasi (`openid.mode=chceckid setup`). Parameter-parameter tersebut adalah identias dari pengguna *OpenID* (`openid.identity`), asosiasi (`openid.assoc`) dan *URL* yang akan mengalihkan pengguna kembali ke *Website Relying Party* (`openid.return to`). Selain itu, biasanya *Relying Party* juga akan meminta parameter tambahan kepada *OpenID provider* seperti nama lengkap, jenis kelamin, dan *email*.
   Berikut adalah contoh parameter yang digunakan:

```
GET     /openid-auth?OpenID.mode=
checked_setup&OpenID.identity
=http://alice.myid.org&openid
.return_to=http://www.example.org
/openid-login&openid.assoc_handle
=xxxxxxxxxxxxxxx
HTTP/1.1
Host: myid.org
```

Kode 2.1 Parameter dalam otentikasi

e. Selanjutnya pengguna akan melakukan proses otentikasi di *Website OpenID provider* (memasukan penggunaname dan *password*) kemudian memberikan persetujuan dari permintaan *Relying Party* untuk mengakses informasi pengguna tersebut. Demi kenyamaan pengguna, biasanya *Relying Party* akan membuat sebuah *HTTP Cookies* untuk memudahkan pengguna kedepannya jika akan kembali menggunakan *OpenID*.

f. Setelah pengguna melakukan proses otentikasi dan pemberian izin akses (otorisasi), kemudian pengguna akan di alihkan kembali ke *Website Relying Party* melalui sebuah *URL* yang terdiri dari parameter *authentication response* (`openid.mode=id res`), indentitas pengguna *OpenID* (`openid.identity`), alamat *Relying Party* (`openid.return to`), parameter *nonce* (`OpenID.response nonce`) dan *cryptographic signature* (`openid.sig`) [6]

Jadi, hampir semua proses otentikasi pada *OpenID* dapat dilihat langsung, karena semua parameter tersimpan di dalam *redirect-URL*. untuk menghindari hal yang tidak diinginkan, *Provider* akan memvalidasi *redirect-URL* yang di terimanya dari *Relying Party* berdasarkan *redirect-URL* yang sudah terdaftar sebelumnya.

Saat ini *OpenID* masih dalam tahap adopsi dan akan semakin populer seiring dengan banyaknya perusahaan besar yang mulai menerima dan menjadi penyedia layanan (*provider*) *OpenID*.

Berdasarkan data pada Desember 2009 diperkirakan sudah lebih dari 1 Miliar pengguna *OpenID* menggunakan *URL* (*URL enabled*) dengan sekitar kurang lebih 9 juta situs yang mendukung *login* melalui *OpenID*.

Berikut beberapa penyedia layanan *OpenID*

Tabel 2.1 Daftar *Provider OpenID*

Penyedia Layanan	Format URL
Google	Google.com/account/me
Yahoo	*OpenID*.yahoo.com
My*OpenID*	username.my*OpenID*.com
LiveJournal	username.livejournal.com
AOL	*OpenID*.aol.com/username
Wordpress	username.wordpress.com
Blogspot	username.blogspot.com
Verisign	username.pip.verisignlabs.com
Google Profile	google.com/profiles/username

2.4 Hypertext Transfer Protocol (HTTP)

Pengertian *HTTP* adalah Singkatan dari *Hyper Text Transfer Protocol*, protokol yang mendasari oleh *World Wide Web*. Dalam pengertian *HTTP* menetapkan bagaimana pesan diformat dan ditransmisikan, dan apa tindakan dari *webserver* dan *browser* sebagai respon pada berbagai perintah.

Secara khusus *HTTP* dapat diartikan sebagai pesan yang berbentuk format dan dapat dikirim melalui sebuah *server* ke *Client*. *HTTP* juga berfungsi sebagai alat yang mengatur bentuk dan aksi apapun yang dilakukan oleh *Web Server* juga *Web Browser* untuk direspon atas perintah yang ada pada protokol *HTTP*.

2.5 *Packet Sniffing*

Packet sniffing adalah metode penyadapan data-data yang berada di jaringan, dimana, dengan menggunakan metode ini, seorang peretas dapat melihat data pengguna lain yang dikirim dijaringan. Adapun perangkat yang digunakan untuk melakukan *sniffing* disebut dengan

packet sniffer. *Packet sniffer* ini bekerja pada layer *Transmission Control Protocol/Internet Protocol (TCP/IP)* [14].

Paket sniffing dapat dilakukan tidak hanya pada jaringan lokal saja (*LAN*), tetapi juga bisa dilakukan pada jaringan yang lebih luas(*WAN*). *Packet sniffing* ini umumnya digunakan untuk menganalisa paket data yang dikirimkan dijaringan dengan tujuan untuk mengetahui permasalahan yang terjadi dijaringan. Namun dalam perkembangannya, *packet sniffing* lebih sering digunakan untuk tujuan kejahatan, seperti untuk tindakan *hacking*, mencuri akun pengguna, dan mencuri data-data penting pengguna. Misalnya ketika terjadi komunikasi antara pengguna A dan pengguna B, maka dengan metode *packet sniffing*, semua data komunikasi antar A dan B dapat diketahui. Sama halnya jika seorang pengguna *login* pada suatu *website*, dengan metode *packet sniffing* ini, userid dan *password* pengguna dapat diketahui.

Packet sniffing tidak hanya bersifat pasif, dimana peretas hanya bertindak sebagai pendengar tanpa melakukan apa-apa terhadap koneksi antara 2 pengguna. Tetapi *packet sniffing* juga dapat bersifat aktif, dimana peretas dapat mengubah data-data pengguna yang dikirim dijaringan, sehingga data yang diterima oleh pengguna lain menjadi tidak sesuai dengan data asli yang dikirim sebelumnya.

2.6 *Session Hijacking*

Session hijacking adalah sebuah teknik *hacking* dengan tujuan untuk mengambil alih akun pengguna lain dengan mencuri sessi yang tersimpan didalam *cookie*. Ketika pengguna mengunjungi sebuah *website*, maka *website* akan mengirimkan *cookie* dan cookie tersebut akan tersimpan pada *browser* yang digunakan oleh pengguna. *Cookie* yang dikirimkan inilah yang kemudian dicuri peretas melalui *proses packet sniffing*, kemudian *cookie* yang sudah didapatkan disimpan ke *browser* yang digunakan oleh peretas.

Ada banyak cara yang dilakukan untuk mendapatkan cookie, seperti melakukan *packet sniffing* ataupun dengan mencurinya langsung pada *browser* yang digunakan pengguna. Dengan metode *session hijacking* ini, seorang peretas dapat mencuri akun pengguna tanpa perlu mengetahui *username* dan *password* pengguna.

2.7 Cookie

Cookie adalah berkas yang dibuat oleh *website* yang dikunjungi pengguna yang digunakan untuk menyimpan informasi pengguna, seperti informasi profil dari pengguna [15]. *Cookie* ini tersimpan *browser* yang digunakan oleh pengguna. Jadi ketika pengguna membuka *website* yang pernah dikunjungi sebelumnya, maka *browser* akan mengirimkan *cookie* yang sesuai kepada *website* tersebut. Dengan cara ini, *website* dapat menampilkan informasi yang sesuai dengan pengaturan pengguna.

Cookie dapat juga disebut sebagai *ID card* pengguna saat terhubung pada *website* tertentu. [9]. Cookie dapat menyimpan berbagai jenis informasi, termasuk di antaranya informasi pribadi seperti nama, alamat rumah, alamat *email*, atau nomor telepon pengguna. Akan tetapi informasi ini hanya akan disimpan jika pengguna pernah memberikan informasi ini kepada *website* tersebut. *Website* tidak dapat mengakses informasi yang tidak pernah diberikan pengguna kepada *website* tersebut, dan *website* juga tidak dapat mengakses berkas lainnya pada komputer pengguna..

2.8 Phishing

Phishing adalah suatu metode *hacking* yang digunakan untuk mendapatkan informasi pribadi pengguna seperti userid, *password* dan data-data rahasia lainnya dengan cara menyamar sebagai pihak lain. *Phishing* berasalah dari kata bahasa inggris *fishing* yang mempunyai arti memancing, memancing dalam hal ini adalah meancing pengguna untuk memberikan informasi rahasia seperti *password*, kode *PIN* dan lain sebagainya [16]

Ada banyak cara untuk melakukan *phishing*, seperti mengirimkan informasi palsu ke *email* pengguna, mengggunakan sebuah tautan menuju halaman palsu dan lain sebagainya, yang tujuan utamanya adalah untuk mendapatkan *userid* dan *password* pengguna.

2.9 Kode Token Akses

Kode token adalah sebuah kode acak yang mengidentifikasi seorang pengguna atau halaman dan dapat digunakan oleh aplikasi untuk melakukan panggilan *API(Application Programming Interface)*. Kode token ini meyimpan informasi tentang masa berlaku kode token tersebut serta aplikasi yang diakses oleh kode token tersebut.[17]
Dalam penggunaannya, ada beberapa jenis kode token, yaitu

a. Kode token pengguna
   Kode token jenis ini digunakan ketika aplikasi(dalam hal ini *website Relying Party*) ingin mengakses data-data pengguna, seperti membaca, memodifikasi atau menulis data pengguna tertentu atas nama mereka. Kode token ini didapatkan melalui dialog *login*. Untuk alur *Relying Party* mendapatkan kode token dapat dilihat pada gambar 2.9

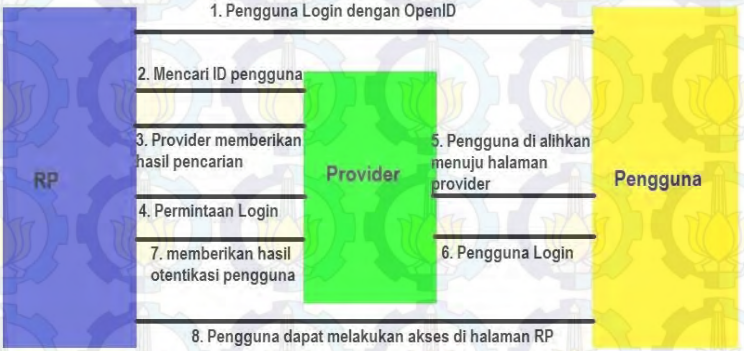
Gambar 2.9 Alur *Relying Party* mendapatkan kode token penguna [17]

b. Kode token aplikasi
   Kode token aplikasi ini diperlukan untuk memodifikasi dan membaca pengaturan aplikasi. Kode token jenis ini dihasilkan secara rahasia oleh aplikasi dan *provider*.
c. Kode token halaman
   Kode token ini hampir sama dengan kode token pengguna. Untuk mendapatkan kode token ini, terlebih dahulu harus mendapatkan kode token pengguna dan hak untuk mengakses *manage_pages*. Setelah itu barulah kemudian mendapatkan kode token halaman ini.
d. Kode token klien
   Kode token klien adalah sebuah kode pengenal yang dapat ditanamkan kedalam kode biner aplikasi *mobile* dan aplikasi desktop yang digunakan untuk mengidentifikasi aplikasi pengguna. Kode token klien digunakan untuk mengakses aplikasi setingkat *API*, tapi dengan hak akses yang terbatas.

BAB 3

DESAIN DAN SISTEMATIKA PENGUJIAN

3.1 Desain Sistem



Gambar 3.1 Arsitektur Protokol *OpenID*

Adapun prinsip kerja *OpenID* secara teknis dan proses otentikasi di lakukan adalah sebagai berikut:

a. Pengguna memasukan *URL OpenID* pada kolom yang tersedia (misalnya: http://alice.myopenid.com) di *website Relying Party*.
b. Untuk mengetahui *provider* yang digunakan oleh pengguna, *Relying Party* menggunakan *URL* yang dikirimkan oleh pengguna. Pada umumnya *URL identifier OpenID* terletak pada bagian <*head*> dari kode *html website Relying Party*, lebih spesifiknya terletak pada tag <*link*>. Didalam *tag* <*link*> ini terdapat atribut rel="`openid2.provider`" serta *link* dari *openid provider* yang digunakan.
c. Setelah *Relying Party* mengetahui *OpenID provider* yang digunakan, maka *Relying Party* akan membuat sebuah prosedur keamanan dengan *OpenID Provider* menggunakan algoritma Diffie-Hellman yaitu dengan membuat *security key*.

Security key ini akan digunakan untuk memverifikasi dalam pertukaran data antara keduanya.

d. Kemudian *Relying Party* akan mengalihkan pengguna ke *website OpenID* menggunakan sebuah *URL identifier* yang terdiri dari semua parameter yang dibutuhkan untuk proses otentikasi (*OpenID.mode=chceckid setup*). Parameter-parameter tersebut adalah identias dari pengguna *OpenID* (*openid.identity*), asosiasi (*openid.assoc*) dan *URL* yang akan mengalihkan pengguna kembali ke *website Relying Party* (*openid.return to*). Selain itu, biasanya *Relying Party* juga akan meminta parameter tambahan kepada *OpenID provider* seperti nama lengkap, jenis kelamin, dan *email*.
   Berikut adalah contoh parameter yang digunakan:

```
GET     /openid-auth?OpenID.mode=
checked_setup&OpenID.identity
=http://alice.myid.org&openid
.return_to=http://www.example.org
/openid-login&openid.assoc_handle
=xxxxxxxxxxxxxxx
HTTP/1.1
Host: myid.org
```

Kode 3.1 Parameter dalam otentikasi

e. Selanjutnya pengguna akan melakukan proses otentikasi di *website OpenID provider* (memasukan *username* dan *password*) kemudian memberikan persetujuan dari permintaan *Relying Party* untuk mengakses informasi pengguna tersebut. Demi kenyamaan pengguna, biasanya *Relying Party* akan membuat sebuah *HTTP cookie* untuk memudahkan pengguna kedepannya jika akan kembali menggunakan *OpenID*.

f. Setelah pengguna melakukan proses otentikasi dan pemberian izin akses (otorisasi), kemudian pengguna akan di alihkan kembali ke *website Relying Party* melalui sebuah *URL* yang

terdiri dari parameter *authentication response* (`openid.mode=id res`), indentitas pengguna *OpenID* (`openid.identity`), alamat *Relying Party* (`OpenID.return to`), parameter nonce (`openid.response nonce`) dan *cryptographic signature* (`openid.sig`)[6].

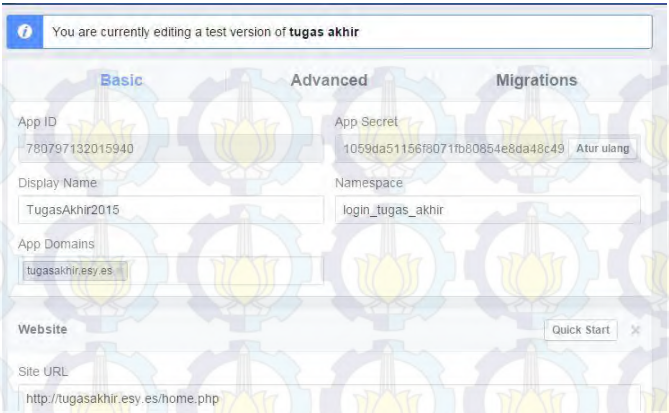
Objek dalam penelitian ini adalah *website Relying Party* yang menggunakan *provider* Google dan Facebook. *Website Relying Party* ini dipasang pada server lokal (*localhost*) dan juga pada *server online*. Selain *website Relying Party* ini, pada penelitian ini juga menguji *website Relying Party* yang sudah ada seperti *website* livejournal.com. idhostinger.com, udemy.com dan mediafire.com.

Dan untuk tahapan pembuatam *website Relying Party* adalah sebagai berikut:

a. Pembuatan Google *API* dan Facebook *API*

Langkah pertama dalam membuat *website Relying Party* adalah dengan membuat aplikasi Google dan Facebook untuk mendapatkan kode *Oauth Client ID/App ID* dan *Client Secret/App Secret* pada *website* console.developers.google.com dan developers.facebook.com. untuk contoh hasil pembuatan aplikasi pada Google dan Facebook dapat dilihat pada gambar 3.2 dan gambar 3.3

Gambar 3.2 Data Aplikasi Google



Gambar 3.3 Data Aplikasi Facebook

b. Membuat basisdata

Setelah membuat aplikasi pada *provider* dan mendapatkan *client id/app id* dan *client secret/app secret*, kemudian dilanjutkan dengan membuat basisdata sebagai media penyimpanan data-data pengguna yang didapat dari *provider*. Tabel untuk setiap *provider* berbeda, ini berdasarkan dari data yang didapat dari *provider* tersebut. Untuk lebih detail bisa dilihat pada tabel 3.1 dan tabel 3.2.

Tabel 3.1 Data pengguna untuk *provider* Google.

No	Nama	Keterangan
1	user_id	Nomor registrasi pengguna
2	name	Nama pengguna
3	email	Alamat email pengguna
4	password	*Password* pengguna
5	social_id	Id pengguna di *provider*
6	picture	Foto pengguna
7	created	Tanggal pengguna mendaftar

Untuk *provider* Facebook, digunakan 2 buah tabel, yaitu untuk menyimpan data pengguna yang didapat dari Facebook dan tabel untuk menyimpan kode token.

Tabel 3.2 Data pengguna untuk *provider* Facebook

No	Nama	Keterangan
1	userid	Id pengguna di *provider*
2	nama	Nama pengguna
3	email	Alamat *email* pengguna
4	gender	Jenis kelamin pengguna

Tabel 3.3 Tabel Kode Token

No	Nama	Keterangan
1	userid	Id pengguna di *provider*
2	akses_token	Kode token

c. Membuat berkas konfigurasi.
Data untuk koneksi ke basisdata serta data *client id, client secret,* alamat *website* dan alamat pengalihan ditambahkan dalam berkas konfigurasi.

d. Menambahkan pustaka *oauth* yang digunakan oleh *provider*
Dalam *website* ini, digunakan 2 buah *Software Development Kit (SDK)* yaitu *PHP SDK* dari Google dan Facebook *Javascript SDK*.

e. Menyimpan data pengguna ke basisdata dan membuat sesi.
Ketika pengguna login dengan *OpenID*, maka *provider* akan memberikan data-data pengguna dan data tersebut akan disimpan dalam basisdata server *website Relying Party*. Dari data tersebut akan dibuat sesi untuk pengguna. Untuk lebih jelas bisa dilihat pada kode 3.2

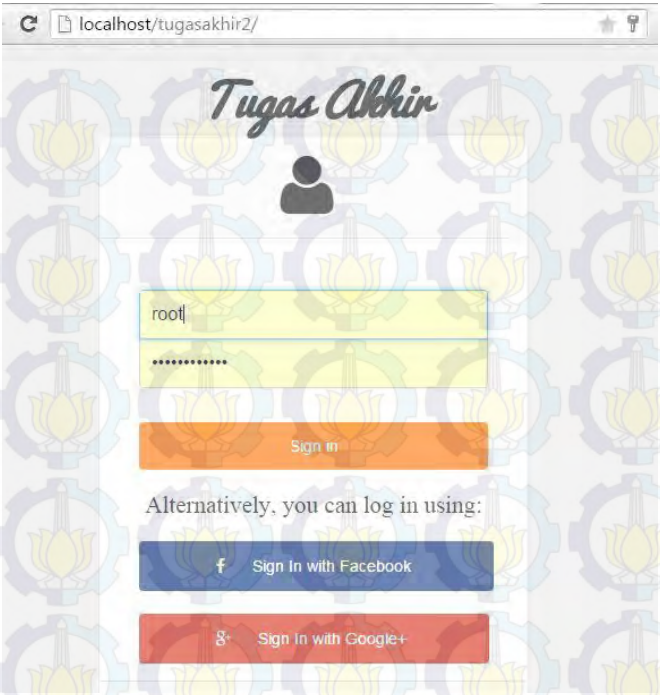
```
$cek_userid=mysql_num_rows(mysql_query
 ("SELECT userid FROM fb_data WHERE
userid='$userid'"));
if ($cek_userid > 0){
    $query=mysql_query("select * from
fb_data where userid='$userid' ") or
die(mysql_error());
    $xxx=mysql_num_rows($query);
      if($xxx==TRUE){
      $_SESSION['logged_in']=$userid;
            $_SESSION['name']=$nama;
      header("location:home.php");
      }
  }
```

Kode 3.2 Kode untuk memeriksa pengguna ketika *Login* dengan *OpenID*

Proses otentikasi terbagi menjadi 2 bagian, pertama ketika pengguna sudah pernah mendaftar sebelumnya pada *website Relying Party* dan kedua ketika pengguna baru pertama kali mendaftar pada *website Relying Party*. Jika pengguna sudah pernah terdaftar, data langsung diambil dari *basisdata* untuk dibuat sesi baru. Tapi jika pengguna belum terdaftar, maka data dari *provider* terlebih dahulu akan disimpan dalam *basisdata* kemudian diambil untuk dibuat sesi baru.

f. Hasil

Setelah menambahkan semua pustaka dan *SDK*, dan semua proses konfigurasi serta penambahan fitur lainnya, maka *website Relying Party* sudah bisa digunakan. Dan untuk hasil dari *website Relying Party* dapat dilihat pada gambar 3.4

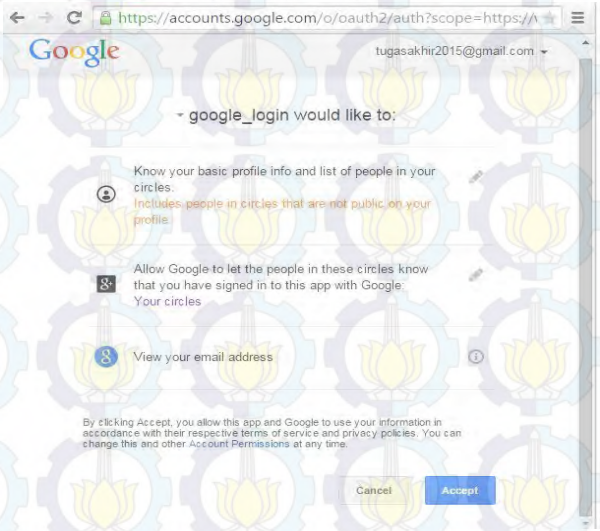


Gambar 3.4 Halaman Awal *website Relying Party*

3.2 Simulasi Sistem

Setelah semua konfigurasi dan pengaturan pada *website Relying Party* selesai, kemudian *website* tersebut akan dicoba untuk *login* dengan menggunakan *provider* Facebook dan Google. Tahapan proses *login* dengan *OpenID* sama dengan langkah-langkah yang dijelaskan pada sub bab 3.1.2. proses permintaan izin aplikasi untuk mengakses data pengguna di *provider* dapat dilihat pada gambar 3.5 dan gambar 3.6

Gambar 3.5 Permintaan izin aplikasi untuk mengakses data pengguna di Facebook



Gambar 3.6 Permintaan izin aplikasi untuk mengakses data pengguna di Google

Setelah pengguna memberi persetujuan kepada aplikasi, maka *provider* akan memberikan data-data pengguna yang diminta oleh aplikasi. Semua data tersebut akan disimpan didalam basisdata *website Relying Party*. Data yang tersimpan dalam *basisdata* dapat dilihat pada gambar 3.7

user_id	name	email	password	social_id	picture
1	ikan koi	goodmonday007@gmail.com		110778121099310401040	https://lh5.googleusercontent

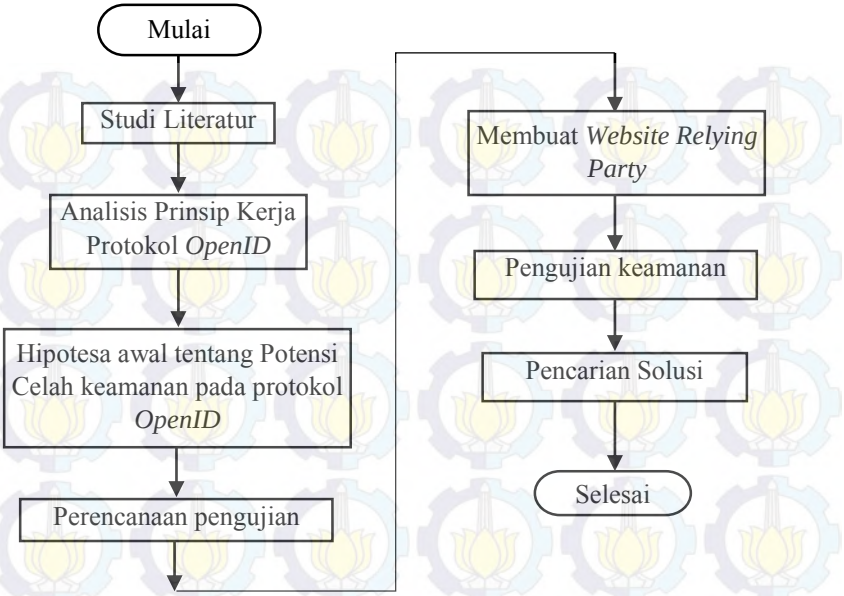
userid	nama	email	gender
998668940156805	Akhirul	jhontheripper414@gmail.com	male

userid	akses_token
998668940156805	CAALDcbteCvsBAMIWldQRzh4PX1JyE1QmCvM0kb9ZACy1IfHpi...

Gambar 3.7 Data pengguna yang tersimpan di basisdata *Relying Party*

3.3 Sistematika Pengujian Sistem

Proses penelitian tugas akhir ini diawali dengan melakukan studi literatur mengenai protokol *OpenID* untuk mengetahui bagaimana prinsip kerja dari protokol *OpenID* ini. Kemudian dilakukan analisis prinsip kerja protokol *OpenID*, dengan tujuan untuk mencari tahu potensi celah keamanan yang ada. Berdasarkan hasil analisis, didapat hipotesa awal tentang potensi celah keamanan yang terdapat pada protokol *OpenID*. Dan dari hipotesa tersebut, dibuat perencanaan pengujian yang akan dilakukan. Pengujian dilakukan untuk membuktikan kebenaran hipotesa sebelumnya. Kemudian pembuatan *website Relying Party* sebagai objek pengujian. *Website Relying Party* tersebut menggunakan *provider* Facebook dan Google. Selain *website Relying Party* yang dibuat sendiri, objek pengujian pada penelitian ini juga menggunakan *website-website* umum yang juga menggunakan protokol *OpenID*, seperti *website* toko *online*, dengan tujuan untuk mengetahui validitas data yang didapat pada *website Relying Party* yang dibuat sendiri. Dari pengujian yang telah dilakukan, didapat beberapa celah keamanan yang ada pada protokol *OpenID*. Berdasarkan celah keamanan yang ditemukan kemudian dicari solusi untuk mengatasi permasalahan akibat celah keamanan tersebut.



Gambar 3.8 Metodelogi penelitian

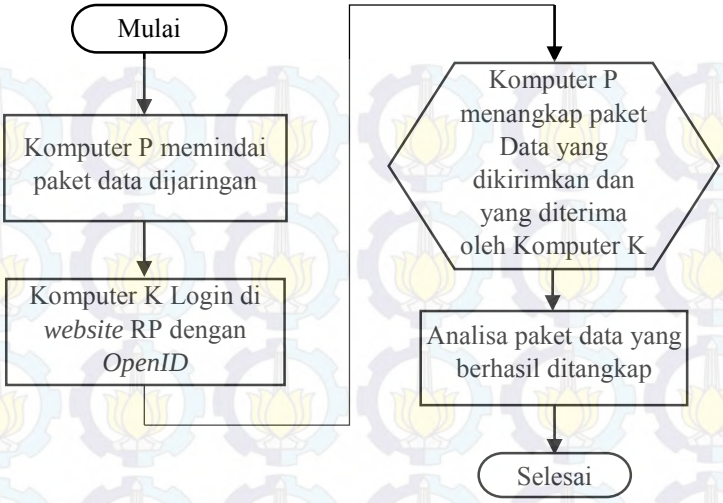
3.3.1 Pengujian Protokol Transfer Paket Data

Dalam komunikasi dan pertukaran data yang terjadi dalam *OpenID* menggunakan protokol *HTTP (Hypertext Transfer Protocol)*. penggunaan protokol *HTTP* tanpa dibungkus dengan jalur yang aman (*SSL*) memungkinkn semua data yang dikirimkan dijaringan bisa dilihat oleh pihak lain. Untuk membuktikan hal tersebut, akan dilakukan pemindaian paket data dijaringan dengan tujuan untuk menangkap paket data yang dikirim dalam otentikasi pada *OpenID*.

Perlengkapan yang digunakan dalam pengujian ini adalah:

a. Komputer penguji P
b. Komputer korban K
c. Jaringan *LAN*
d. Aplikasi *wireshark*

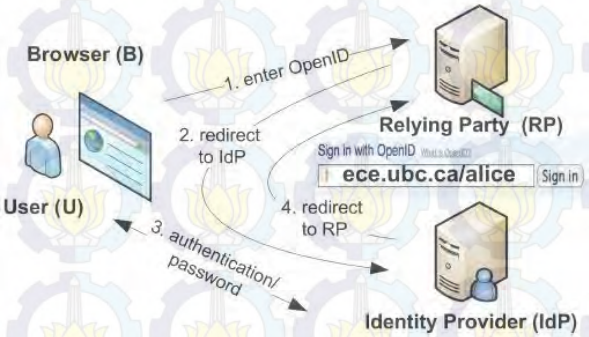
Untuk skema pengujian dapat dilihat pada gambar 3.9



Gambar 3.9 Skema pengujian protokol HTTP

3.3.2 DNS Spoofing

Proses Otentikasi pada *OpenID* melalui sebuah proses pengalihan dari *website Relying Party* menuju *website provider*. Untuk lebih jelas bisa dilihat pada gambar 3.10

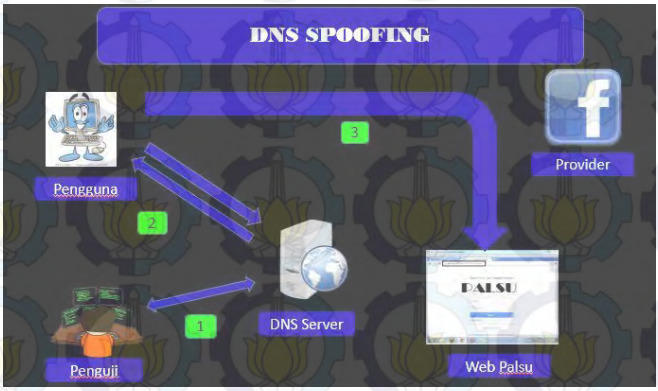


Gambar 3.10 Proses otentikasi *OpenID*[3]

Dari gambar 3.11 terlihat, pada nomor proses nomor 2, pengguna dialihkan menuju *website provider*. Proses pengalihan ini berpotensi

menimbulkan bahaya ketika pengguna dialihkan tidak menuju *website provider*, tetapi *website* lain. Sebuah pengujian akan dilakukan untuk membuktikan potensi celah keamanan pada proses pengalihan ini, yaitu dengan melakukan *DNS Spoofing* pada jaringan yang digunakan pengguna *OpenID*.

DNS Spoofing adalah teknik khusus yang digunakan untuk mengambil alih *DNS server* sehingga *DNS* atau layanan yang menyimpan nama domain dan *IP address* sebuah situs akan dialihkan ke komputer *webserver* lain. Secara umum, proses *DNS Spoofing* dapat dilihat pada gambar 3.11



Gambar 3.11 Cara kerja *DNS Spoofing*

3.3.3 Meretas *Website Relying Party*

Proses otentikasi pada *OpenID* terjadi dipihak ketiga, yaitu *provider*, bukan pada *Relying Party*. Proses ini bisa menjadi ancaman buat pengguna.Ancaman yang bisa terjadi adalah ketika peretas merubah link referensi pada atribut tombol *login* dengan *OpenID*, misalnya jika peretas merubah tautan asli tombol "Sign in with Google" dengan tautan halaman *login* yang dibuat mirip dengan halaman *login* Google. Selain itu, tautan referensi *OpenID* yang sangat panjang juga membuat potensi celah ini semakin besar. Untuk lebih jelas tentang tautan referensi pada tombol "Sign in with Google" bisa dilihat pada kode 3.3

```
https://accounts.google.com/o/oauth2/auth?res
ponse_type=code&redirect_uri=http%3A%2F%2Ftug
asakhirsaya2015.esy.es%2F&client_id=392853343
800-
ad93vos2rbd7erno4skg2j0u89b8olpv.apps.googleu
sercontent.com&scope=https%3A%2F%2Fwww.google
apis.com%2Fauth%2Fplus.login+https%3A%2F%2Fww
w.googleapis.com%2Fauth%2Fuserinfo.email+http
s%3A%2F%2Fwww.googleapis.com%2Fauth%2Fplus.me
&access type=offline&approval prompt=auto
```

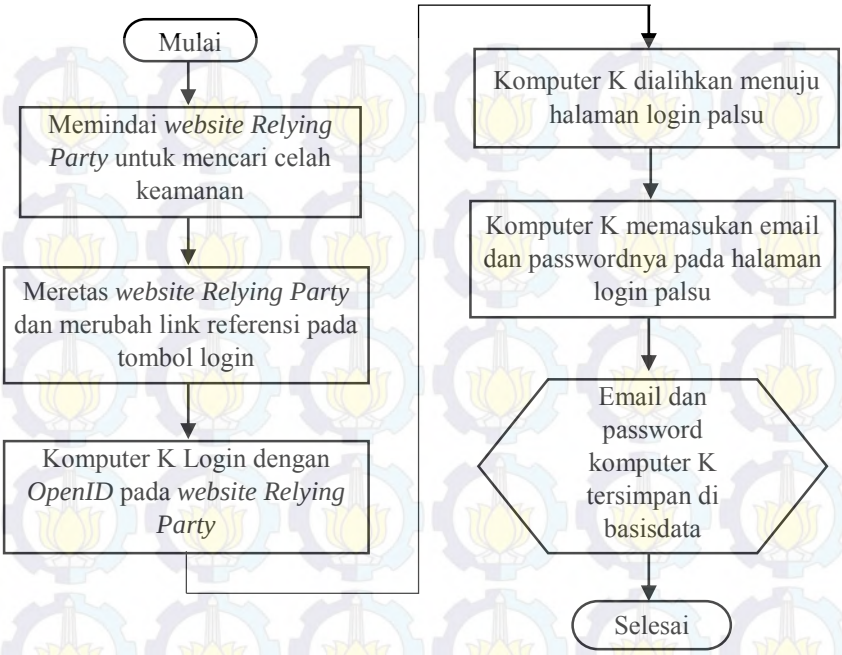
Kode 3.3 Tautan referensi pada tombol "Sign with Google"

Dengan tautan referensi yang sangat panjang tersebut, pengguna lebih cenderung tidak mempedulikannya. Teknik peretasan ini dikenal dengan nama *phishing*.

Pada tahap ini, pengujian dilakukan bertujuan untuk mengambil alih *website Relying Party* dan mengubah tautan referensi pada tombol *login OpenID* serta mencuri *email* dan *password* pengguna yang tersimpan di*provider*. Properti yang digunakan dalam pengujian ini adalah:

a. Komputer penguji P
b. Komputer korban K
c. *Webserver* untuk *Relying Party*

Untuk skema pengujian dapat dilihat pada gambar 3.12



Gambar 3.12 Skema kerja peretasan *website Relying Party*

3.3.4 Pengujian keamanan *AppID/Client ID* dan *App Secret/Client Secret*

App ID/Client ID dan *App Secret/Client Secret* adalah kode yang berfungsi sebagai identitas dari aplikasi yang dibuat *Relying Party* di *provider*. *Provider* mengenali *Relying Party* dari *App ID/Client ID* dan *App Secret/Client Secret*. Pengujian tahap 3 bertujuan untuk mengetahui respon dari *provider* jika *App ID/Client ID* dan *App Secret/Client Secret* digunakan pada *website Relying Party* yang lain.

Adapun yang diperlukan dalam pengujian ini adalah

1. Satu aplikasi dari Facebook dan Google

   a.Facebook

   App ID : 777843195644667

   App Secret : cac60cc4a1ad308f781b8718717e9eb9

b. Google

Client ID :392853343800-ad93vos2rbd7erno4skg2j0u89b8olpv.apps.Googleusercontent.com

Client Secret : RrK7QLi6doFgb7VPOJU2twU2

2. Dua *website Relying Party*

   Website Relying Party A pada *webhosting* dan *website Relying Party* B di *localhost*.

3. 1 Komputer P

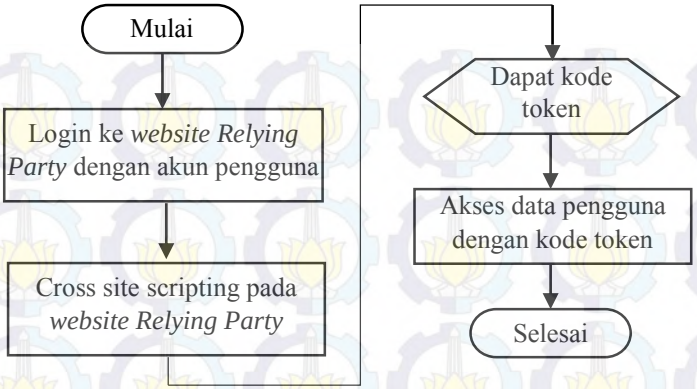
   Komputer P bertindak sebagai penguji.

3.3.5 Pencurian Kode Token

Kode token adalah sebuah kode otorisasi yang diberikan *provider* kepada *Relying Party* supaya *Relying Party* bisa mengakses data-data pengguna di *provider*. Ketika seorang pengguna sudah *login* dengan *OpenID*, maka *provider* akan mengirimkan kode token kepada *Relying Party*, dan kemudian *Relying Party* bisa mengakses data-data pengguna. Kode token ini dikirim bersamaan dengan pengiriman parameter-parameter otentikasi.

Sebuah celah keamanan muncul akibat dari penyalahgunaan kode token ini. Jika akun pengguna di *website Relying Party* berhasil di ambil alih, misalnya melalui *session hijacking*, dan kemudian peretas akan membuka *script website Relying Party* ketika pengguna sudah *login* dan kemudian kode token akan ditemukan. Dan peretas bisa menggunakannya untuk mengakses data-data pengguna di *provider* seperti halnya hak akses yang didapatkan oleh *Relying Party*.

Untuk skema pengujian untuk mendapatkan kode token ini dapat dilihat pada gambar 3.13

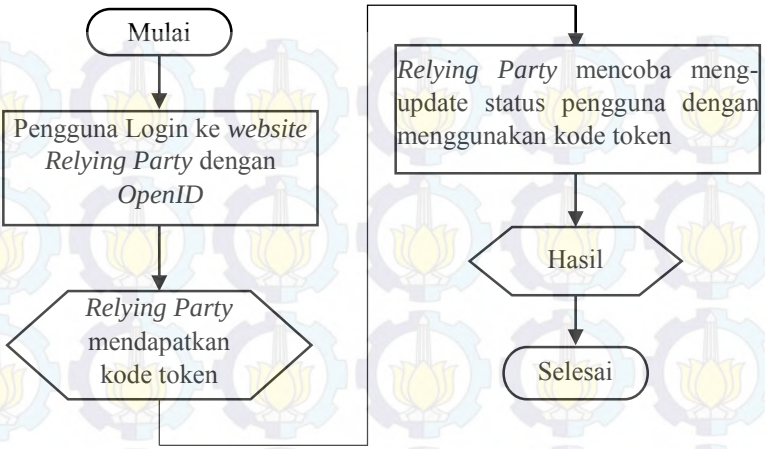


Gambar 3.13 Skema pencurian kode token

3.3.6 Menembus batasan otorisasi *provider*

Setiap *Relying Party* mendapatkan hak untuk mengakses data-data pengguna yang ada di *provider*. Namun, tidak semua data dapat diakses oleh *Relying Party*, hak akses/otorisasi yang didapat oleh *Relying Party* sesuai dengan permintaan *Relying Party* dan telah disetujui oleh pihak *Provider*. Jadi dapat diartikan Relying mempunyai batasan dalam mengakses data pengguna di *provider*.

Pada pengujian ini bertujuan untuk mencoba menembus batasan yang diberikan *provider*. Pengujian ini menggunakan 1 aplikasi *Relying Party* yang tidak mempunyai hak untuk meng-*update status* pengguna di Facebook dan kemudian akan dicoba untuk melakukan *update status* menggunakan kode token.Adapun skema pengujian ini dapat dilihat pada gambar 3.14



Gambar 3.14 Skema menembus batasan otorisasi

(Halaman ini sengaja dikosongkan)

BAB 4

ANALISA DAN PENGUJIAN SISTEM

Dalam bab ini dilakukan pembahasan mengenai analisa dari prinsip kerja protokol *OpenID*. Analisa ini meliputi analisa protokol transfer data (*HTTP*), analisa proses pengalihan pengguna dari *website Relying Party* menuju *website provider*, dan analisa kelemahan *website Relying Party* yang bisa membahayakan data-data pengguna di *provider*.

Selain itu, pada bab ini juga dilakukan pembahasan tentang pengujian dari prinsip kerja protocol *OpenID*. Pengujian dilakukan berdasarkan analisa yang telah dilakukan, dengan menggunakan objek *website Relying Party* pada localhost, menggunakan jasa *webhosting* dan beberapa *website* yang juga menggunakan protokol *OpenID*.

4.1 Analisa Prinsip Kerja *OpenID*

Protokol *OpenID* merupakan protokol yang menggunakan jasa pihak ketiga untuk melakukan otentikasi pengguna. Pada proses otentikasi, pengguna akan dialihkan/redirect dari *website Relying Party* menuju *website OpenID provider*. Proses pengalihan/*redirect* ini terjadi antara 2 jaringan yang berbeda, dan pada tahap ini pengguna sangat rentan terhadap serangan seorang peretas, yaitu melalui *metode man in the middle attack (MITMA).* Berdasarkan prinsip kerja *OpenID*, diketahui bahwa *OpenID* bekerja dengan menggunakan protokol *HTTP GET request*. Penggunaan metode *GET* ini sangat rentan dalam hal keamanan, karena semua paket yang dikirimkan terlihat jelas pada *URL* yang dikirimkan oleh pengguna. Paket yang dikirimkan tersebut merupakan parameter-parameter yang digunakan dalam proses otentikasi seperti parameter identitas dari pengguna *OpenID* (*OpenID.identity*), asosiasi (*OpenID.assoc*) dan *URL* yang akan mengalihkan pengguna kembali ke *website Relying Party* (*OpenID.return to*). Semua parameter ini dapat dilihat oleh pihak lain seperti peretas melalui proses penyadapan (*sniffing*). Dan jika parameter ini didapat oleh seorang peretas, maka peretas bisa menggunakan layanan pada *website Relying Party* seperti menggunakan akun pengguna lain tanpa proses otentikasi.

Selain itu, penggunaan protokol *HTTP* tanpa melalui proses enkripsi seperti menggunakan *secure socket layer (SSL)* memungkinkan seorang peretas bisa mendapatkan *session cookie*s pengguna pada *website Relying Party* dengan cara menyadap di jaringan pengguna atau disebut dengan proses *sniffing*. Dengan memanfaatkan *cookies* ini, peretas juga bisa mengambilalih akun pengguna pada *website Relying Party* tanpa melalui proses otentikasi.

Selanjutnya, potensi celah keamanan yang mungkin terjadi yaitu ketika proses pengalihan pengguna dari *website Relying Party* menuju *website OpenID provider*. Proses yang melibatkan 2 jaringan yang berbeda ini memungkin peretas bisa merubah rute pengalihan tersebut dan pada proses ini pengguna rentan terhadap serangan *phishing*. *Phishing* adalah usaha untuk mendapatkan suatu informasi penting dan rahasia secara tidak sah, seperti pengguna *id, password, pin*, informasi rekening bank, informasi kartu kredit, atau informasi rahasia yang lain. Salah satu teknik yang digunakan adalah dengan mengalihkan pengguna menuju *website* palsu. Proses pengalihan tersebut dapat dilakukan dengan 2 cara, yaitu:

a. *DNS Spoofing*

   Proses yang melibatkan 2 jaringan yang berbeda ini memungkinkan peretas bisa merubah rute pengalihan tersebut. Dengan menggunakan teknik *dns spoofing*, pengguna bisa dialihkan menuju sebuah *website* palsu yang sudah disiapkan oleh peretas, dan jika pengguna terjebak dalam perangkap ini, tentu saja ini sangat berbahaya, karena pengguna *id* dan *password* pengguna bisa di ketahui oleh peretas.

b. Meretas *Website Relying Party*

   Selain melalui dns spoofing, keamanan data pengguna di *provider* juga bisa terancam jika *website Relying Party* menjadi korban hacking, seperti defacing. Ketika *website Relying Party* diambil alih oleh peretas, kemudian peretas merubah link referensi pada tombol "*Login* with Google" menjadi link referensi *website* palsu. Dan ketika pengguna memilih *login* dengan Google, maka pengguna tidak akan

dialihkan menuju *website* Google, tapi menuju *website* palsu dan kemudian pengguna akan diminta memasukan *email* dan *password* nya di *website* palsu tersebut.

Tindakan *phishing* ini tentu sangat berbahaya, karena kelemahan pada *Relying Party*, data-data penting pengguna di *provider* menjadi terancam. Salah satu *provider*, yaitu wordpress.com berusaha mengantisipasi tindakan *phishing* dengan cara tidak menampilkan halaman *login* ketika pengguna menggunakan *OpenID*. Jadi, jika ingin menggunakan *OpenID* wordpress.com, pengguna terlebih dahulu harus *login* di *website* wordpress.com baru bisa *login* dengan URL *OpenID*nya di *website Relying Party*.

Pada protokol *OpenID*, *Relying Party* mempunyai hak untuk mengakses data-data pengguna, tapi tidak semua data dapat pengguna yang ada di *provider* dapat diakses oleh *Relying Party*. Setiap *provider* mempunyai regulasi sendiri dalam mengatur otorisasi kepada *Relying Party*. Seperti pada *provider* Facebook, ketika *Relying Party* mendaftarkan aplikasi *OpenID* kepada Facebook, secara default *Relying Party* akan mendapatkan hak untuk mengakses profil pengguna (user_profiles), *email* dan teman pengguna (*user_friends*). Namun, *Relying Party* juga diberi kesempatan untuk mendapatkan hak otorisasi tambahan dengan memilih hak otorisasi yang diberikan oleh Facebook, dengan syarat *Relying Party* harus membuat sebuah permohonan dan menjelaskan alasan kenapa *Relying Party* memerlukan hak akses tersebut, serta juga harus menjelaskan secara detail bagaimana aplikasi *Relying Party* itu berjalan. Permohonan untuk mendapatkan hak otorisasi akan di pertimbangkan oleh Facebook. jika Facebook menyetujui permohonan tersebut maka *Relying Party* akan mendapatkan hak otorisasi tambahan tersebut. Untuk daftar hak akses yang diberikan oleh Facebook dapat dilihat pada gambar 4.1

Gambar 4.1 Daftar hak akses yang disediakan Facebook

Ketika pengguna *login* dengan menggunakan *OpenID*, maka pengguna akan diminta persetujuan bahwa pengguna akan memberi izin kepada *Relying Party* untuk mengakses data-data pengguna. Pada saat memberi persetujuan ini dibutuhkan ketelitian pengguna, karena data yang akan diakses *Relying Party* tidak hanya identitas pribadi, tapi masih ada fitur lain yang bisa diakses oleh *Relying Party*. Facebook menerapkan regulasi yang cukup ketat tentang otorisasi ini, tapi berbeda dengan Google yang memberi keleluasaan *Relying Party* untuk menambah hak otorisasi mereka tanpa perlu meminta permohonan kepada Google.

Adapun cara *Relying Party* untuk bisa mendapatkan otorisasi terhadap data-data pengguna yaitu dengan menggunakan kode token. Kode token akan didapatkan *Relying Party* setiap pengguna *login* di aplikasi *Relying Party* dengan menggunakan *OpenID*, dan kode token ini juga mempunyai masa berlaku yang terbatas, sehingga tidak setiap waktu *Relying Party* dapat menggunakannya. Namun walaupun begitu, terdapat potensi bahaya ketika kode token ini berhasil dicuri oleh orang lain. Jika kode token ini diketahui orang lain, maka orang tersebut juga akan mendapatkan hak otorisasi yang sama dengan hak otorisasi yang dimiliki oleh *Relying Party*. Misalnya, jika *Relying Party* A

menggunakan *provider* Facebook dan mendapatkan akses yang cukup besar, seperti akses *user_posts* dan *read_mailbox*, maka orang lain yang mendapatkan kode token milik *Relying Party* A juga bisa melakukan hal yang sama dengan yang dilakukan *Relying Party* A.

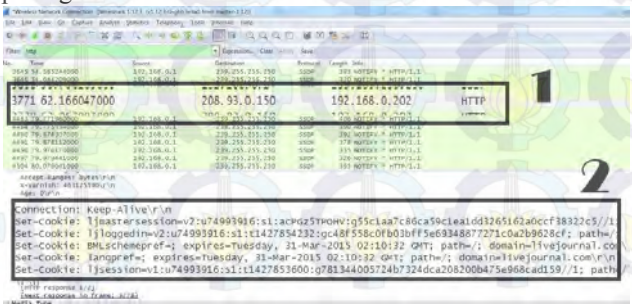
4.2 Pengujian Keamanan Sistem pada *OpenID*

Pengujian keamanan pada protocol *OpenID* berpatokan dari hasil analisa sebelumnya. Pengujian keamanan protokol *OpenID* dilakukan dengan beberapa tahap, yaitu:

1. Pengujian Protokol Transfer Paket Data (*HTTP*)
2. Pencurian kode token
3. Pemalsuan Halaman *Login Provider*
4. Pengujian keamanan *App ID/Client ID* dan *App Secret/Client Secret Relying Party*.
5. Menembus batasan hak akses *Relying Party* yang diberikan *Provider*

4.2.1 Pengujian Protokol Pengiriman Paket Data (*HTTP*)

Untuk mengetahui bagaimana proses pertukaran data dijaringan pada protokol *OpenID* dapat dilakukan dengan cara *sniffing*. Teknik *sniffing* ini dapat dilakukan dengan bantuan aplikasi wireshark. Proses *sniffing* ini dilakukan ketika pengguna A dengan *ip address* 192.168.0.202 melakukan *login* dengan *OpenID* pada *website* livejournal.com dengan *ip address* 208.93.0.150. Hasil *sniffing* dapat dilihat pada gambar 4.2



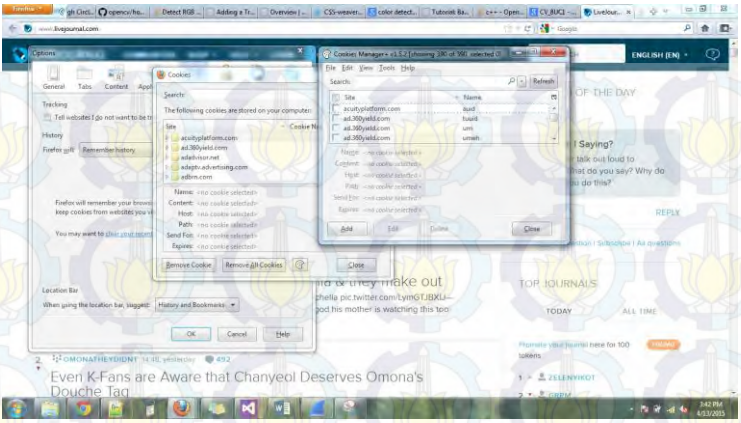
Gambar 4.2 Hasil *Sniffing* pada aplikasi Wireshark

Dari hasil *sniffing* dapat terlihat semua paket data yang terkirim dijaringan antara pengguna A dengan *website* livejournal.com. dan yang paling penting dari hasil *sniffing* ini adalah pada gambar 4.2 nomor 2 terlihat *cookies* yang dikirimkan oleh livejournal.com kepada pengguna A. Ini merupakan celah keamanan yang berbahaya jika *cookies* ini diketahui oleh orang lain atau peretas. *Cookies* dapat digunakan oleh peretas untuk *login* pada akun pengguna A pada *website* livejournal.com tanpa harus melewati proses otentikasi. Teknik pembajakan akun dengan menggunakan *cookies* ini disebut dengan *session hijacking*. Untuk membuktikan celah keamanan pada *cookies* ini, kemudian pengujian dilanjutkan dengan melakukan *session hijacking* pada akun pengguna A.

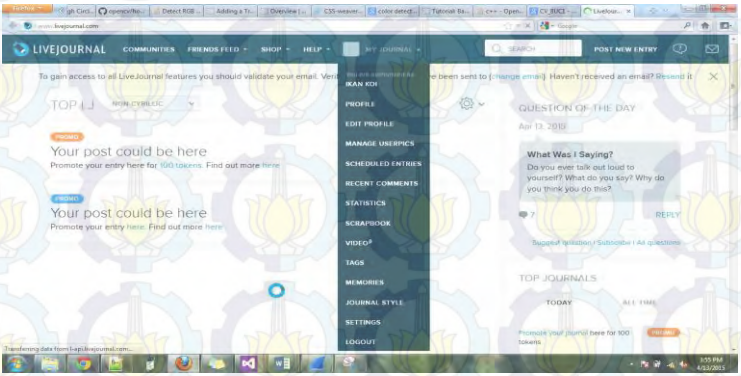
Dari *cookies* yang didapatkan dari *sniffing* tersebut, kemudian *cookies* tersebut disimpan kedalam browser dengan menggunakan *add-on Cookies Manager* pada *browser* mozilla firefox. Daftar *cookies* yang dimasukan kedalam *browser* adalah sebagai berikut:

a. `ljmastersession=v2:u74993916:s1:acPGz5TPOHV:g55c1aa7c86ca59c1ea1dd3265162a0ccf38322c5//1; path=/; domain=www.livejournal.com; HttpOnly`
b. `ljloggedin=v2:u74993916:s1:t1427854232:gc48f558c0fb03bff5e69348877271c0a2b9628cf; path=/; domain=livejournal.com; HttpOnly`
   `BMLschemepref=; expires=Tuesday, 31-Mar-2015 02:10:32 GMT; path=/; domain=livejournal.com`
c. `langpref=; expires=Tuesday, 31-Mar-2015 02:10:32 GMT; path=/; domain=livejournal.com`
d. `ljsession=v1:u74993916:s1:t1427853600:g781344005724b7324dca208200b475e968cad159//1; path=/; domain=livejournal.com; HttpOnly`

Proses dan tahapan pengujian dapat dilihat pada gambar 4.3 dan gambar 4.4



Gambar 4.3 Proses menambahkan *cookies* di browser Mozilla Firefox



Gambar 4.4 Berhasil *login* pada akun Ikan Koi dengan menggunakan *cookies*

Pada gambar 4.4 dapat dilihat bahwa penguji bisa masuk kedalam akun pengguna A tanpa harus melewati proses otentikasi. Hal ini menunjukan salah satu celah keamanan pada protocol *OpenID.*

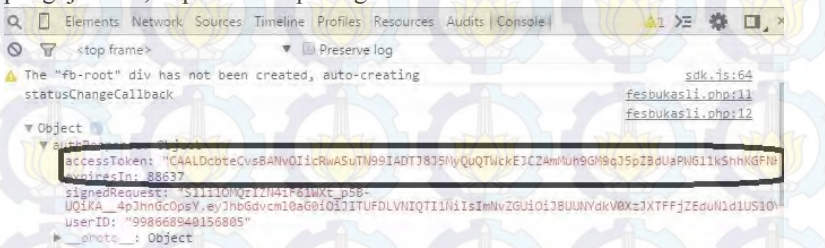
4.2.2 Pencurian Kode Token

Kode token adalah sebuah deretan kode unik yang diberikan *provider* kepada *Relying Party* untuk mengakses data-data pengguna yang ada di *provider*. Kode token ini akan diberikan kepada *Relying*

Party ketika pengguna *login* di *website Relying Party* menggunakan *OpenID*.

Pada pengujian ini, akan dicoba mendapatkan kode token yang diberikan *provider* kepada *Relying Party*. Untuk mendapatkan kode token, pengguna harus *login* terlebih dahulu pada *website Relying Party* dengan *OpenID*. Untuk *provider* yang digunakan adalah Facebook, dengan *SDK (Software Development Kit)* yang digunakan adalah *Javascript SDK* yang disediakan Facebook.

Untuk melihat kode token yang diberikan Facebook, dapat dilakukan dengan cara melihat *element website Relying Party* pada *browser*. Untuk lebih rinci mengenai kode token yang didapat dari pengujian ini, dapat dilihat pada gambar 4.5



Gambar 4.5 Kode Token yang berhasil ditemukan

Dari pengujian ini diketahui bahwa sangat mudah untuk mendapatkan kode token. Potensi bahayanya adalah jika akun pengguna *Relying Party* berhasil diretas melalui *session hijacking* dan peretas menemukan kode token ini, maka peretas juga akan bisa mengakses data-data pengguna di *provider*. Peretas akan memiliki hak otorisasi/akses yang sama dengan hak yang dimiliki oleh *Relying Party*. Celah ini semakin berbahaya jika *Relying Party* mendapatkan hak akses yang cukup besar terhadap data-data pengguna.

4.2.3 Pemalsuan Halaman *Login Provider* (*Phishing*)

Pada saat pengguna memilih *login* dengan menggunakan protokol *OpenID*, maka setelah itu pengguna akan dialihkan ke *website provider*. Pengguna akan diminta *login* dan memasukan *email* dan *password*-nya jika pengguna belum *login* pada *website provider*. Pengujian pada tahap ini adalah mengubah arah pengalihan (*redirect*)

menuju halaman *login* palsu yang mirip dengan halaman *login provider* dan sebagai sampel pada pengujian ini menggunakan 2 buah *provider* yaitu Facebook dan Google.

Tahap awal yang dilakukan adalah dengan membuat sebuah halaman *login* palsu yang mirip dengan halaman *login website provider*. Cara untuk membuat halaman palsu tersebut adalah dengan mengambil *script* asli dari *website provider* dan kemudian menyimpannya pada *webhosting/webserver online*.

Untuk tampilan dari halaman palsu tersebut dapat dilihat pada gambar 4.6 dan 4.7

Gambar 4.6 Halaman *Login* Facebook Palsu

Gambar 4.7 Halaman *Login* Google Palsu

Halaman palsu tersebut berguna untuk menerima data yang dimasukan oleh pengguna, dan ketika pengguna menekan tombol *Sign in*/Masuk, maka *email* dan *password* pengguna akan tersimpan kedalam basisdata.

Tahap selanjutnya adalah membuat basisdata yang berfungsi untuk menyimpan *email* dan *password* pengguna/korban/target. Adapun rincian dari basisdata yang digunakan dapat dilihat pada Tabel 4.1

Tabel 4.1 Tabel penyimpanan *email* dan *password* pengguna dibasisdata

No	Nama	Keterangan
1	*Email*	*Email* pengguna
2	*Password*	*Password* pengguna di *provider*

Setelah membuat halaman palsu dan basisdatanya, tahap selanjutnya adalah merubah alur pengalihan menuju *website provider*. Ada 2 cara untuk mengalihkan pengguna menuju halaman *login* palsu, yaitu:

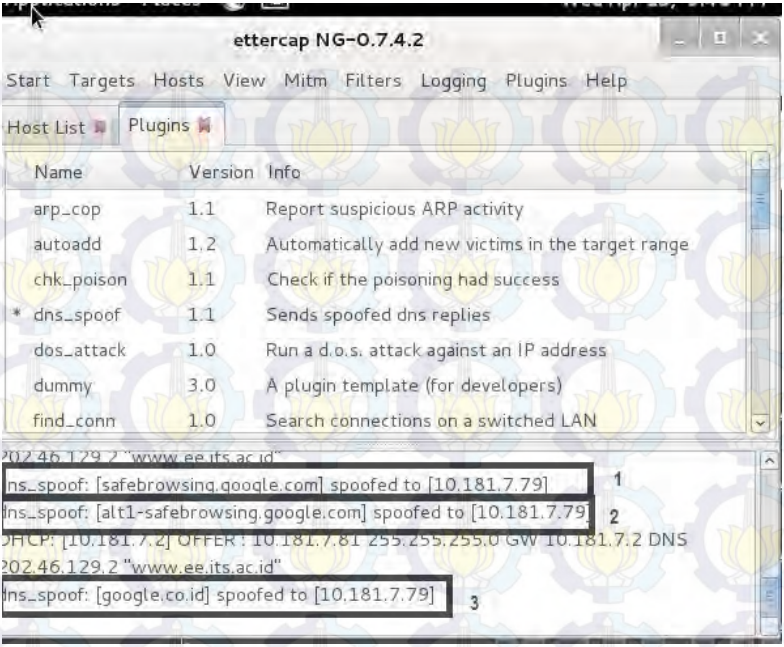
a. DNS Sppofing

Proses peralihan dari *website Re;ying Party* ke *website provider* rentan terhadap serangan *DNS Spoofing*. *DNS Spoofing* adalah teknik yang digunakan untuk mengambil alih DNS server sehingga *DNS* atau layanan yang menyimpan nama domain dan *IP address* sebuah situs akan dialihkan ke server yang lain. Jadi ketika pengguna A memilih *login* dengan Google, pada aturan yang sebenarnya, harusnya pengguna A dialihkan ke *website* Google untuk verifikasi data, namun karena DNS spoofing ini, pengguna A tidak dialihkan ke *website* Google, namun ke *website* yang lain.

Teknik *DNS Spoofing* ini dapat dilakukan dengan menggunakan beberapa *tools*, namun pada pengujian ini menggunakan salah satu tools di sistem operasi Kali Linux, yaitu *Ettercap*. Dan untuk properti yang digunakan dalam pengujian ini yaitu

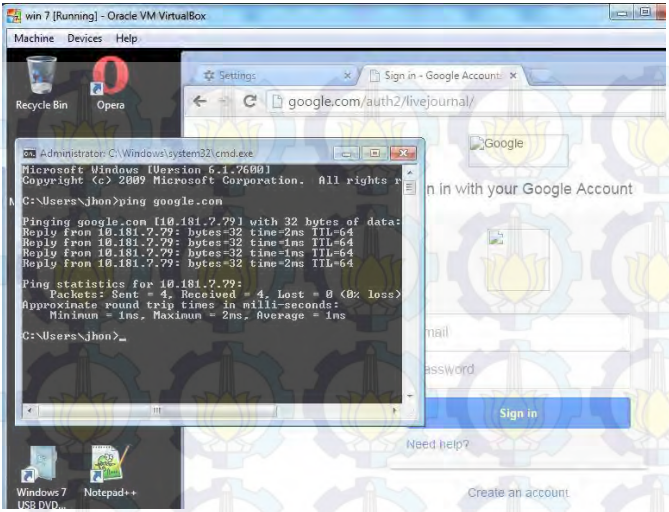
1. Komputer P dengan sistem operasi Kali Linux
   Komputer P bertindak sebagai seorang peretas yang akan mengalihkan korban menuju *website* palsu.
2. Komputer S dengan sistem operasi debian 7.0
   Komputer S bertindak sebagai webserver dari *website* palsu dan akan menyimpan *email* dan *password* korban dengan *ip address* 10.181.7.79
3. Komputer K dengan system operasi windows 7
   Komputer K adalah korban dari *DNS Spoofing* dengan *ip address* 10.181.7.79

Berikut adalah hasil pengujian dengan teknik *DNS spoofing*:



Gambar 4.8 Tampilan Hasil *DNS Spoofing* dari komputer P

Pada Gambar 4.8 terlihat permintaan data menuju google.com semuanya dialihkan menuju Komputer S, termasuk trafik dari computer K juga dialihkan. Pada gambar 4.9 akan terlihat dampak dari *DNS Spoofing* pada Komputer K.



Gambar 4.9 Dampak *DNS Spoofing* pada Komputer K

Ketika Komputer K menjadi korban *DNS Spoofing*, maka semua permintaan data dari Komputer K menuju google.com akan dialihkan menuju Komputer S. Dengan teknik *DNS Spoofing* ini, Komputer K akan percaya bahwa halaman *login* yang tampil adalah halaman asli Google, karena terlihat pada *address bar* bahwa *URL* yang tertulis adalah google.com. padahal sesungguhnya, jika Komputer K dialihkan ke halaman *login* Google, *URL* yang sebenarnya adalah: "https://accounts.Google.com/Service*Login*?service=lso&passive=1209600&continue=https://accounts.Google.com/o/oauth2/auth?zt%3DChQ0WExOcGQwbmc4ZnVrZmFqbjVpOBIfY3VrdGwtZHd2eThmY3A3dGRpbGpLS2J4TjFZU21nSQ%25E2%2588%2599APsBz4gAAAAAVUHZTZ8hNzQJg6HrWO1erSl_A-EB1ZB5%26from_*login*%3D1%26hl%3Den%26e%3D3100077%26as%3D60237f647d58fcf6&ltmpl=popup&shdf=CrECCxIRdGhpcmRQYXJ0eUxvZ29VcmwaugEvL2ltYWdlcy1sc28tb3BlbnNvY2lhbC5nb29nbGV1c2VyY29udGVudC5jb20vZ2FkZ2V0cy9wcm94eT91cmw9aHR0cDovL3doLmxqLnJ1L2dvb2dsZS9sal9sb2dvMTIweDYwdy5wbmcmY29udGFpbmVyPWxzbyZnYWR

nZXQ9YSZyZXdyaXRlTWltZT1pbWFnZS8qJnJlc2l6ZV9oPTEyMCZyZXNpemVfdz0xMjAmbm9fZXhwYW5kPTEMCxIVdGhpcmRQYXJ0eURpc3BsYXlOYW1lGgtMaXZlSm91cm5hbAwLEgZkb21haW4aC0xpdmVKb3VybmFsDAsSFXRoaXJkUGFydHlEaXNwbGF5VHlwZRoHREVGQVVMVAwSA2xzbyIUE-trWmUfoY1IgPQayTHG25vyY28oATIUELVUHhIVTyyqT_77029nd4gr0Gc&hl=en&sarp=1&scc=1”

Dengan *URL* dengan tulisan acak dan panjang tersebut, besar kemungkinan semua pengguna tidak akan tahu jika mereka dialihkan.Sehingga Komputer K terjebak dan memasukan *email* dan *passwordnya* pada kolom di *website* palsu tersebut. Hal ini tentu sangat berbahaya dan merugikan Komputer K.

b. *Deface website Relying Party*

Selain dengan menggunakan teknik *dns spoofing*, cara untuk mengalihkan pengguna menuju halaman *login* palsu adalah dengan memanfaatkan kelemahan pada *website Relying Party*. Kelemahan yang dimaksud adalah ketidakmampuan *website Relying Party* dalam menghadapi serangan dari peretas, seperti *website Relying Party* diambil alih oleh peretas.

Untuk itu, pada tahap pengujian ini, dilakukan peretasan pada *website Relying Party* dengan tujuan untuk merubah tampilan *website Relying Party*, yaitu merubah atribut tautan referensi pada tombol *login* dengan protokol *OpenID* (Google dan Facebook).

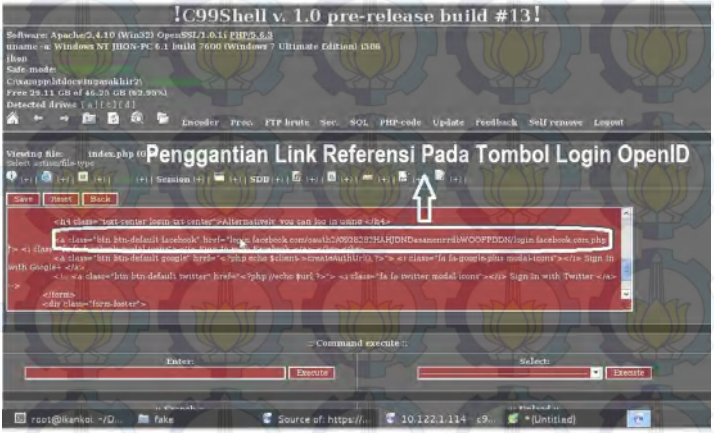
Pada pengujian ini dilakukan simulasi peretasan terhadap *website Relying Party*. Adapun perlengkapan yang digunakan dalam simulasi ini adalah:

1. *Website Relying Party*
   Website Relying Party ini menggunakan *provider* Facebook dan Google dengan alamat http://10.122.1.114/tugasakhir2
2. Halaman *Login* Palsu
   Tampilan dari halaman palsu ini sama dengan halaman *login* Facebook dan Google. Halaman ini masih terdapat dalam server yang sama dengan *website Relying Party*.

3. Komputer P

   Komputer P adalah computer penguji yang akan meretas *website Relying Party*

Skenario dalam pengujian ini adalah, pertama Komputer P melakukan penetrasi pada *website Relying Party*.Komputer P menemukan celah *Local File Inclusion(LFI)* pada *website Relying Party* untuk dimasukan sebuah *backdoor* kedalam server. Dengan menggunakan *backdoor* tersebut, Komputer kemudian membuat 2 direktori dan menambahkan 2 halaman palsu. Untuk halaman palsu "login.Facebook.php" dimasukan kedalam direktori "oauth*provider*=facebook.com" dan halaman palsu "*login*.Google.php" dimasukan kedalam direktori "oauth*provider*=google.com". Proses peretasan dapat dilihat pada gambar 4.10.



Gambar 4.10 Peretasan *website Relying Party*

Setelah memasukan halaman palsu kedalam server, kemudian Komputer P mengubah script halaman index.php pada *website Relying Party*, dimana halaman index.php ini adalah halaman *login website Relying Party*. Untuk proses perubahan atribut URL pada tombol *login OpenID* di *website Relying Party* dapat dilihat pada gambar 4.11 dan 4.12

Gambar 4.11 Tautan referensi yang asli

Gambar 4.12 Tautan referensi diubah menuju halaman palsu

Untuk hasil setelah peretasan dapat dilihat pada gambar 4.13.

Gambar 4.13 Hasil teknik phishing

Proses pengalihan pada protokol *OpenID* ini memunculkan sebuah celah keaamanan yang mempermudah seorang peretas untuk mencuri akun pengguna. Selain itu, lemahnya sistem keamanan yang digunakan oleh *Relying Party* juga menjadi ancaman terhadap data-data pengguna di*provider*.

4.2.4 Pengujian Keamanan *App ID/Client ID* Dan *App Secret/Client Secret Relying Party*

App ID/Client ID dan *App Secret/Client Secret* adalah kode yang berfungsi sebagai identitas dari aplikasi yang dibuat *Relying Party* di *provider*. *Provider* mengenali *Relying Party* dari *App ID/Client ID* dan *App Secret/Client Secret*. Pengujian tahap 3 bertujuan untuk mengetahui

respon dari *provider* jika *App ID/Client ID* dan *App Secret/Client Secret* digunakan pada *website Relying Party* yang lain.

Adapun yang diperlukan dalam pengujian ini adalah

1.Satu Aplikasi dari Facebook dan Google

a. Facebook

App id : 777843195644667

App Secret : cac60cc4a1ad308f781b8718717e9eb9

b. Google

Client ID :392853343800-ad93vos2rbd7erno4skg 2j0u89b8olpv.apps.Googleusercontent.com

Client Secret : RrK7QLi6doFgb7VPOJU2twU2

2.Dua *website Relying Party*

Kedua *website Relying Party* dipasang pada webhosting.

3.Satu Komputer P

Komputer P bertindak sebagai penguji.

Website Relying Party A pada *webhosting* dengan alamat http://tugasakhirsaya2015.esy.es dan *website Relying Party* B dengan alamat http://tugasakhir.esy.es. *Website Relying Party* A adalah *website* yang terdaftar dalam aplikasi Facebook dan Google serta yang mempunyai hak untuk menggunakan *App ID/Client ID* dan *App Secret/Client Secret*.

Kemudian, *App ID/Client ID* dan *App Secret/Client Secret* juga digunakan pada *website Relying Party* B yang sebenarnya tidak terdaftar dalam aplikasi Facebook dan Google. Ketika di coba *login* dengan menggunakan protokol *OpenID* pada *website Relying Party* B, ternyata tidak bisa *login* dan *provider* akan menampilkan pesan kesalahan seperti pada gambar 4.14 dan 4.15.

Gambar 4.14 Pesan Kesalahan dari Facebook

Gambar 4.15 Pesan Kesalahan dari Google

Berdasarkan pengujian keamanan *App ID/Client ID* dan *App Secret/Client Secret* dapat diketahui bahwa *provider* sudah mempunyai sistem keamanan untuk mengatasi tindakan penyalahgunaan/pencurian *App ID/Client ID* dan *App Secret/Client Secret* dari *Relying Party*. Sehingga jika ada peretas yang menggunakan *App ID/Client ID* dan *App Secret/Client Secret* pihak Y, maka protocol *OpenID* pada peretas tidak akan berjalan dan akan menampilkan pesan kesalahan.

4.2.5 Menembus Batasan Hak Akses *Relying Party* Yang Diberikan *Provider*

Setiap *Relying Party* mendapatkan hak untuk mengakses data dan beberapa kepentingan lainnya berdasarkan permintaan *Relying Party* ketika membuat aplikasi di *provider*. Seperti ketika membuat aplikasi Facebook, secara *default* Facebook memberi izin kepada pembuat aplikasi untuk mendapatkan identitas pengguna, alamat *email* dan daftar teman pengguna. Untuk mendapatkan hak akses tambahan seperti mengupdate status pengguna, pembuat aplikasi harus mengirim permintaan kepada Facebook serta menjelaskan tujuan dan cara kerja aplikasi yang dibuat dengan detail. Jika Facebook menyetujui permintaan tersebut, aplikasi akan mendapatkan hak akses untuk mengupdate status pengguna.

Pada pengujian ini, akan menguji apakah batasan hak akses tersebut bisa ditembus atau tidak. Untuk itu digunakan 1 buat aplikasi yang hanya mendapatkan hak akses default dari Facebook, kemudian akan dicoba mengupdate status pengguna dengan menggunakan alamat URL berikut:

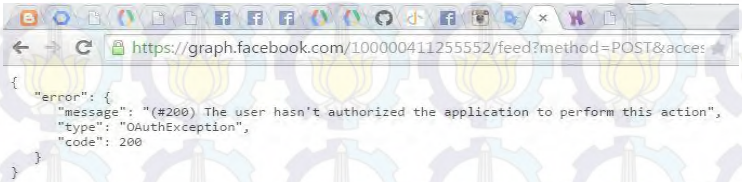
```
https://graph.Facebook.com/ID-
FACEBOOK/feed?method=POST&access_token=ACCESS-
TOKEN-KORBAN&message=KALIMAT STATUS
```

1. ID-FACEBOOK adalah kode id pengguna di Facebook
2. ACCESS TOKEN adalah kode token yang diberikan *provider* kepada aplikasi untuk menggnakan hak akses yang diberikan.
3. KALIMAT STATUS adalah isi dari status yang akan di tulis di halaman pengguna.

URL tersebut akan diubah menjadi seperti berikut:

```
https://graph.Facebook.com/100000411255552/feed?
method=POST
&access_token=CAALDcbteCvsBAGVFNgf4dY6INqLdGsoOk
FVyx6GVel9dxRuQyZBrU5mkzbdoSiNr1la0HIcGaZBialU4y
AyMTIohmGZCRFOp7suqxNF0UciUHXAuMswbmJ67kPBJBuMgt
jCgDu4AoVxuajfKHDIyOeKpEl5o4agZBAGge2ImbXBt5NYaU
VIkJuVPaGabiq4dxVzwVr5SuzGHCWKgjh3N&message=UJI
COBA UPDATE STATUS
```

Selanjutnya URL tersebut akan dijalankan di browser dan menghasilkan tampilan kesalahan seperti gambar 4.16 berikut:



Gambar 4.16 Pesan Kesalahan otorisasi

Ternyata Facebook tidak memberikan izin kepada aplikasi untuk mengupdate status pengguna, karena belum mendapat hak dari Facebook.

4.3 Potensi Celah Keamanan dan Potensi Bahaya yang dapat terjadi

Berdasarkan 5 tahapan pengujian keamanan yang telah dilakukan, diketahui bahwa terdapat beberapa celah keamanan yang pada protokol *OpenID* diantaranya adalah:

1. *Website Relying Party* yang menggunakan *HTTP* tanpa dilengkapi dengan layer yang aman (*Secure Socket Layer/SSL*) membuat semua paket data yang dikirimkan dan diterima pengguna dapat diketahui oleh pihak lain dan akun pengguna di *Relying Party* sangat rentan untuk dicuri. Tidak hanya akun pengguna di *Relying Party* yang menjadi rentan terhadap tindakan *hacking*, tetapi data-data pengguna di *Provider* juga ikut terancam. Karena ketika akun pengguna di *Relying Party* berhasil dicuri, peretas juga berpotensi dapat menemukan kode token yang tersimpan pada *Website Relying Party*. Dan jika kode token ini berhasil ditemukan, maka peretas akan mendapatkan hak akses yang sama dengan hak yang didapat oleh *Relying Party*, akan menjadi lebih bahaya jika *Relying Party* mendapatkan hak akses yang cukup besar dari *Provider*,

seperti bisa mengakses kotak pesan di *provider* Facebook dan Google, ataupun mengakses data pengguna di *Google Drive*.

Berbeda halnya dengan metode *login* biasa, ketika akun pengguna disuatu *Website* dicuri peretas, maka yang akan terkena dampak hanya akun pengguna di *Website* tersebut, tidak berpengaruh terhadap data-data pengguna di *Website* lain.

Contoh kasus penyalahgunaan kode token ini adalah tindakan *spamming* di Facebook, dimana banyak akun pengguna mengirimkan suatu secara otomatis tanpa diketahui oleh pemilik akun.

2. Proses otentikasi yang berada diluar sistem/*website Relying Party* membuat pengguna harus melewati proses pengalihan dari *Website Relying Party* menuju *Website Provider*. Proses pengalihan ini sangat rentan terhadap tindakan *Phishing*. Karena dengan adanya proses pengalihan ini, membuka peluang kepada peretas untuk mengubah tujuan pengalihan menuju *Website* palsu. Dan dampak paling bahaya dari celah ini adalah pencurian *email* dan *password* pengguna di *Provider*, dan jika ini terjadi maka semua data-data pengguna menjadi terancam.

   Pada metode login biasa, proses otentikasi berada didalam sistem *Website* penyedia layanan, tidak menggunakan jasa pihak ketiga, dan pengguna tidak melewati proses pengalihan. Sehingga peluang pengguna menjadi korban *phishing* tidak sebesar pada protokol *OpenID*, kecuali jika dari awal pengguna sudah terjebak dalam *dns Spoofing*. Dan walaupun pengguna sudah terjebak dalam *dns Spoofing*, dan akun pengguna berhasil dicuri oleh peretas, yang terkena dampak hanya akun pengguna pada *Website* penyedia layanan, tidak ada pengaruh dengan akun pengguna pada *provider* (*provider email* yang digunakan pengguna).

   Celah keamanan ini juga berpotensi menyebabkan pengguna mengalami kerugian finansial, karena ada beberapa *Website* toko *online*(*e-commerce*) yang menggunakan protokol *OpenID* seperti *Website* lazada.co.id, zalora.co.id, tokopedia.com, alibaba.com, blibli.com, elevenia.co.id dan bhineka.com.

Ketika akun pengguna pada *website-website* tersebut dicuri, bisa terjadi penyalahgunaan seperti menggunakan saldo pengguna yang tersimpan pada *website* tersebut, ataupun pemalsuan akun penjual dan lain sebagainya. Walaupun beberapa *website* toko *online* mengirimkan pesan konfirmasi ke *email* pengguna ketika terjadi transaksi, tetapi karena *email* dan *password* pengguna di *provider* juga dicuri dengan teknik *phishing*, *email* konfirmasi juga bisa disetujui sendiri oleh peretas. Ini menunjukan bahwa dampak dari celah ini sangat besar.

4.4 Saran dan Metode untuk Mencegah Pengguna Terjebak dalam Celah Keamanan

Dari celah keamanan yang berhasil ditemukan pada protokol *OpenID*, diketahui bahwa yang paling merasakan kerugian akibat celah keamanan ini adalah pengguna. Untuk membantu pengguna terhindar dari jebakan pada celah keamanan tersebut, ada beberapa cara dan metode yang perlu ditambahkan pada sistem/*website* yang menggunakan protocol *OpenID*, yaitu:

1. Untuk mencegah paket data yang dikirim dan diterima oleh pengguna dilihat/diketahui oleh pihak lain, maka pihak *Relying Party* sebaiknya atau mungkin diharuskan menggunakan layer yang aman, yaitu menggunakan protokol *HTTPS*. Dengan menggunakan *HTTPS*, paket data yang dikirim pengguna dijaringan dalam kondisi terenkripsi, sehingga tidak mudah diketahui oleh pihak lain.
2. Untuk mencegah pengguna terjebak dalam tindakan *phishing*, sejauh ini belum ada solusi yang benar-benar bisa mengatasi masalah tersebut, ditambah lagi protokol *OpenID* harus melewati proses pengalihan dari *website Relying Party* menuju *website Provider*. Walaupun begitu ada cara yang bisa membantu pengguna supaya terhindar dari jebakan *phishing*,

a. Sebaiknya *provider* tidak menampilkan halaman *login* ketika pengguna dialihkan menuju *website provider* jika pengguna belum *login* di *website provider*. Pengguna yang belum *login*

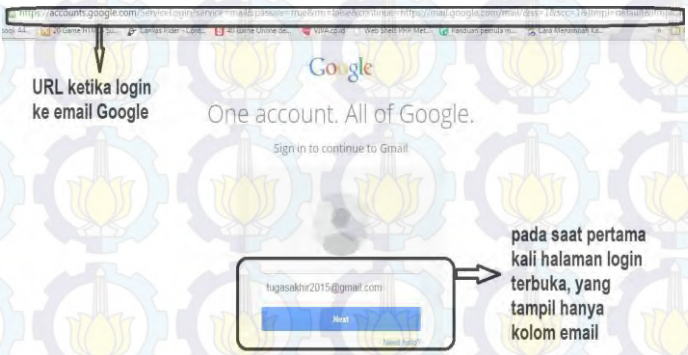
bisa *login* dengan membuka *website provider* pada tabulasi browser yang berbeda. Dengan begitu pengguna akan bisa mengetahui jika terjadi *DNS Spoofing* dijaringan, karena tampilan halaman *login* pada proses pengalihan berbeda dengan halaman *login* ketika pertama kali membuka *website provider*. Untuk mengetahui perbedaan antara kedua hal tersebut, dapat dilihat pada gambar 4.17 , gambar 4.18, gambar 4.19 dan gambar 4.20.

Gambar 4.17 Halaman utama Facebook

Gambar 4.18 Halaman *Login* ketika proses pengalihan

Ketika pengguna membuka *website* https://www.facebook.com/ maka halaman yang akan tampil seperti pada gambar 4.17, dan jika halaman yang tampil tidak seperti gambar 4.17, pengguna harus waspada karena bisa saja itu adalah sebuah jebakan *phishing*. Jika seorang peretas melakukan *phishing* dengan teknik *DNS Spoofing* pada protokol *OpenID*, peretas membuat sebuah halaman *login* yang mirip dengan dengan halaman *login* Facebook seperti pada gambar 4.18, peretas tidak bisa membuat 2 halaman palsu untuk 1 *provider* yang sama, karena prinsip kerja *DNS Spoofing* adalah berdasarkan request *ip address* menuju *webserver* halaman palsu, tidak menggunakan request *DNS*.

Dengan demikian, ketika peretas melakukan *DNS Spoofing*, dan pengguna membuka halaman https://www.facebook.com/ dan yang tampil seperti pada gambar 4.18, pengguna harus hati-hati dan sebaiknya tidak memasukan *email* dan *password*nya pada halaman tersebut.



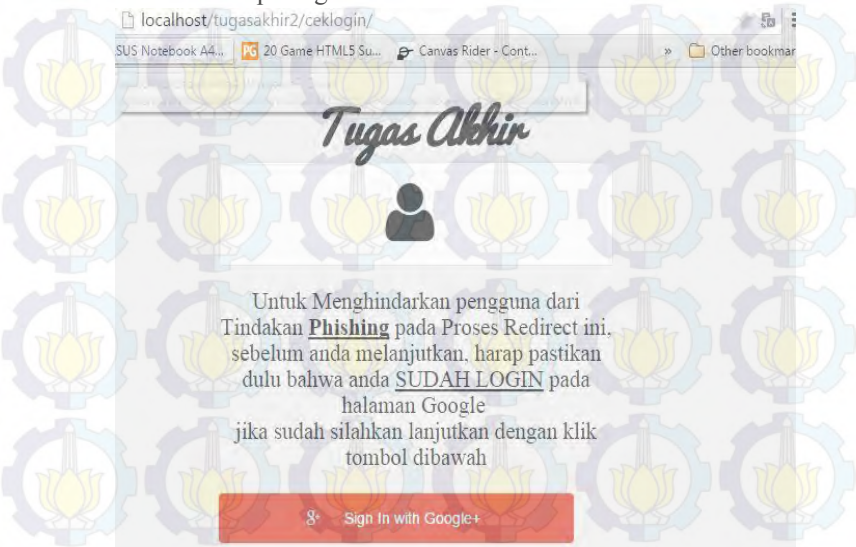
Gambar 4.19 Halaman *Login* Gmail tanpa pengalihan

Gambar 4.20 Halaman *Login* Gmail ketika proses pengalihan

Untuk halaman *login* Gmail asli tanpa pengalihan dengan halaman *login* Gmail pada proses pengalihan hampir sama, hanya terdapat sedikit perbedaan, yaitu perbedaan *URL* dan perbedaan kolom *login*. Pada halaman *login* Gmail asli tanpa pengalihan, hanya terdapat kolom *email* saja, sedangkan halaman *login* Gmail pada proses pengalihan terdapat kolom *email* dan kolom *password*. Sama halnya seperti *login* pada Facebook, pengguna harus lebih teliti memperhatikan tampilan dari halaman *login* Google, terlebih karena halaman *login* tanpa pengalihan dan halaman *login* pada proses pengalihan memiliki tampilan yang hampir sama.

b. *Relying Party* memberikan sebuah peringatan ketika pengguna memilih *login* dengan *OpenID*. Peringatan tersebut bertujuan untuk memastikan bahwa pengguna telah *login* pada *website provider*, jika pengguna belum *login* pada *website provider*, maka pengguna harus *login* terlebih dahulu di *website provider* sebelum *login* dengan *OpenID* pada

website Relying Party. Adapun contoh peringatan tersebut bisa dilihat pada gambar 4.21



Gambar 4.21 Peringatan sebelum *login* dengan *OpenID*

c. Adapun cara untuk terhindar dari tindakan *DNS Spoofing* adalah dengan mengubah *ip address* pengguna. Karena teknik *DNS Spoofing* biasanya dilakukan dengan memasukan *ip address* korban kedalam *tools spoofer* (seperti Ettercap, Cain & Abel, dan lain-lain). *DNS Spoofing* tidak berjalan pada *ip address* computer yang tidak terdapat dalam target *spoofer*. Jadi jika pengguna merasa sudah terjebak dalam *DNS Spoofing*, maka cara terbaik adalah dengan mengganti *ip address* yang sedang digunakan.

3. Untuk menghindari tindakan pencurian kode token, pihak *Relying Party* harus bisa menyimpan kode token tersebut pada tempat yang benar-benar aman, dimana tempat penyimpanan tersebut tidak dapat diakses dari sisi pengguna melalui browser, seperti dengan menyimpan dalam kode

PHP, dimana *PHP* merupakan bahasa pemrograman *server* side.

Penggunaan protokol *HTTPS* juga sangat dibutuhkan supaya kode token tidak dicuri melalui paket data yang dikirimkan di jaringan.

4. Setiap *Provider* harus membuat sebuah regulasi dan sebuah standar keamanan untuk *website Relying Party* yang ingin menggunakan jasa *provider* tersebut. *Website* yang boleh menggunakan protokol *OpenID* hanya *website* yang mempunyai kredibilitas yang tinggi dan reputasi yang benar-benar baik serta dengan standar keamanan yang tinggi. Dan pihak *provider* harus bisa menyeleksi *website* yang ingin menggunakan jasa *provider* tersebut, karena selama ini setiap *website* memiliki kebebasan untuk menggunakan protokol *OpenID* dan mendaftar pada suatu *provider* tanpa regulasi apapun. Dengan dibuatnya sebuah regulasi dan standar keamanan, diharapkan dapat meminimalisir dan mengurangi potensi celah keamanan yang mungkin terjadi dan mengurangi ancaman keamanan data pengguna akibat lemahnya sistem keamanan yang digunakan oleh *Relying Party*.

(Halaman ini sengaja dikosongkan)

DAFTAR PUSTAKA

[1] Amarudin, Widyawan, dan W. Najib,. 2014 "Analisis Keamanan Jaringan *Single Sign On (SSO)* Dengan Lightweight Directory Access Protocol (LDAP) Menggunakan Metode MITMA". Yogyakarta: Jurusan Teknik Elektro dan Teknologi Informasi Universitas Gadjah Mada.

[2] Pavol Sovis, Florian Kohlar dan Jorg Schwenk. "*Security Analysis of OpenID*" Jerman: Ruhr-University Bochum

[3] San-Tsai, Sun, E. Pospisil dan I. Muslukhov 2012 "*What Makes Users Refuse Web Single Sign-On? An Empirical Investigation of OpenID*". Symposium on Usable Privacy and Security (SOUPS). University of British Columbia Vancouver, BC, Canada

[4] San-Tsai Sun dan Konstantin Beznosov. 2012. "*On the Security of Web Single Sign-On*". Conference on Computer and Communications Security. Kanada: Department of Electrical and Computer Engineering University of British Columbia

[5] Tsyrklevich, Eugene. 2007. "*Single Sign On for the Internet: A Security Story*" BlackHat. Amerika Serikat: Las Vegas

[6] Urue˜na, Manuel dan Christian Busquiel. "Analysis of a Privacy Vulnerability in the OpenID Authentication Protocol". Spanyol:Universidad Carlos III de Madrid

[7] Zhou, Yuchen dan David Evans. 2014. "SSO Scan: Automated Testing of Web Applications for Single Sign-On Vulnerabilities" In 23rd USENIX Security Symposium, San Diego, 20–22 August 2014. Amerika Serikat:University of Virginia.

[8] Adhika Dwi Firmansyah, Idris Winarno, S.ST, M.Kom. 2010. "Implementasi OpenID Pada EEPIS Network". Surabaya: Jurusan Teknik Informatika Politeknik Elektronika Negeri Surabaya

[9] Fauziah, Septi Andryana. 2009. "Hijacking Session Pada Sistem Keamanan Komputer(Studi Kasus Pencegahan Virus Pada E-Mail)" Jurnal Artificial, ICT Research Center UNAS, Jakarta

[10] Wijaya, Ignatius Kusuma1, Puspaningtyas Sanjoyo Adi2.2012."Sistem Otentikasi Single Sign-On Menggunakan Algoritma Diffie-Hellman Dan Menggunakan Database Parallel Dengan Menggunakan Rmi (Remote Method Invocation)".Seminar Nasional Teknologi Informasi & Komunikasi Terapan 2012 (Semantik 2012)

[11] Abduh, Muhammad. Derry, R. Febrtiandar. "Multi Factor Authentication" <URL: http://ki.stei.itb.ac.id/2013/10/30/multi-factor-authentication/>, juni 2015

[12] Anonymous. "Yahoo meets OpenID Log in to websites with your Yahoo account" <URL: http://openid.yahoo.com/> juni 2015

[13] Anonymous. "What is OpenID?" <URL: http://openid.net/get-an-openid/what-is-openid/> Juni 2015

[14] Sabeel Ansari, Rajeev S.G. and Chandrashekar H.S. "Packet Sniffing: A Brief Introduction." 0278-6648/02/$17.00 2002 IEEE

[15] Anonymous. "Mengaktifkan atau menonaktifkan cookie" <URL:https://support.google.com/accounts/answer/61416?hl=id> juni 2015

[16] ITB Network Information Center. "Email Phishing"<URL:https://nic.itb.ac.id/mail/email-phising > juni 2015

[17] Anonymous. "Access Tokens" <URL: https://developers.facebook.com/docs/facebook-login/access-tokens> juni 2015

BAB 5

PENUTUP

5.1 Kesimpulan

Berdasarkan hasil analisa dan pengujian yang telah dilakukan terhadap system yang protokol *OpenID*, dapat ditarik beberapa kesimpulan:

1. *Website Relying Party* yang menggunakan protokol *HTTP* membuat semua paket data yang dikirim dan diterima oleh pengguna dapat dilihat oleh pihak lain. Hal tersebut membuat pengguna menjadi rentan menjadi korban tindakan *Session Hijacking* dan tindak pembajakan akun pengguna yang ada di *Relying Party*.

   Ketika akun pengguna bisa diambil alih oleh pihak lain, akan menimbulkan ancaman baru yang bisa merugikan pengguna, yaitu pencurian kode token. Ketika kode token berhasil dicuri, maka data-data pengguna yang ada di *provider* juga ikut terancam dan bisa disalahgunakan. Terutama untuk *Relying Party* yang mendapatkan hak otorisasi yang besar dari *provider*.
2. Kesalahan *Relying Party* dalam mengelola dan menyimpan kode token yang diberikan oleh *provider* juga dapat mengancam keamanan data pengguna di *provider*. Karena kesalahan tersebut, kode token lebih mudah dicuri dan dapat disalahgunakan oleh pihak lain.
3. Proses pengalihan ketika otentikasi pengguna menimbulkan celah keamanan yang bisa membuat pengguna terjebak dalam tindakan *phishing*. Pengguna rentan dialihkan menuju *website* palsu yang dibuat mirip dengan halaman *website provider*.
4. Kelemahan sistem keamanan yang digunakan *Relying Party* juga bisa membahayakan data-data pengguna yang ada di *provider*.
5. Facebook dan Google telah menerapkan aturan baru dengan mengharuskan *Relying Party* mendaftarkan alamat *URL* pengalihan setelah otentikasi pengguna, sehingga bisa

menghindari pengguna dari penyalahgunaan *AppID/Client ID* dan *App Secret/Client Secret* atau yang lebih dikenal dengan teknik *Covert Redirect*.

6. Regulasi untuk membatasi hak otorisasi yang diberlakukan oleh Facebook akan meningkatkan tingkat kepercayaan pengguna terhadap *Relying Party* serta juga bisa menghindari pengguna dari penyalahgunaan otorisasi oleh *Relying Party*

5.2 Saran

Saran untuk pengembangan tugas akhir ini adalah:

a. Proses *sniffing* harus lebih mendalam untuk mendapatkan semua parameter-parameter yang digunakan (selain *cookies*) dalam proses otentikasi dan dicoba *login* dengan menggunakan parameter-paramter tersebut.
b. Selain melakukan *pasive sniffing*, coba lakukan *active sniffing* untuk mengirim parameter otentikasi palsu kepada *Relying Party*.
c. Membuat fitur baru dalam *OpenID* dimana *provider* hanya bertindak sebagai otentikator dan tidak memberikan hak kepada *Relying Party* untuk mengakses data-data pengguna yang ada di *provider*, dengan tujuan untuk mengurangi dampak buruk dari pencurian kode token.

FINAL PROJECT – TE 141599

VULNERABILITY ANALISYS ON SINGLE SIGN ON WEBSITE THAT USE OPENID PROTOCOL

Akhirul Hajri
NRP 2210100093

Advisors
Christyowidiasmoro, ST.,MT
Reza Fuad Rachmadi, ST., MT

Departement of Electrical Engineering
Faculty of Industrial Technology
Sepuluh Nopember Institute of Technology
Surabaya 2015

ABSTRAK

Single sign on adalah metode yang memungkinkan pengguna hanya perlu melakukan satu kali proses otentikasi untuk menggunakan beberapa layanan. Salah satu protokol dalam *single sign on* adalah protokol *OpenID*. Proses otentikasi pada *OpenID* yang menggunakan satu akun untuk beberapa *website*, serta tanpa menggunakan sebuah kode sandi dan proses otentikasi berada diluar sistem/*website* penyedia layanan, timbul pertanyaan bagaimana tingkat keamanan pada protokol *OpenID* ini Untuk itu dilakukan pengujian keamanan pada protokol *OpenID*.Tugas Akhir ini bertujuan untuk menguji dan menemukan potensi celah keamanan yang terdapat pada protokol *OpenID* serta menemukan solusi untuk mengatasi dan menghindarkan pengguna dari bahaya celah keamanan tersebut. Berdasarkan pengujian yang telah dilakukan proses pengalihan memungkinkan pengguna menjadi korban *Phishing*. Cara terbaik untuk mencegah tindakan *phishing* adalah dengan cara tidak menampilkan halaman *login* pada proses pengalihan jika pengguna belum diotentikasi di Provider. Selain itu ketika akun pengguna di *Relying Party* dicuri, maka berpotensi kode token yang tersimpan pada *Relying Party* juga dapat dicuri dan ini akan membahayakan data-data pengguna di Provider. Untuk mengatasi celah keamanan ini, Provider harus membuat sebuah regulasi untuk membatasi *website* yang bisa menggunakan protokol *OpenID*.

Kata kunci: *Single sign on*, *OpenID*, *phishing,* kode token,dan keamanan data.

(Halaman ini sengaja dikosongkan)

ABSTRACT

Single sign on is a method which allow users to gain access to several services with just one authentication. One of the protocols in a single sign-on is the OpenID protocol. Authentication process in OpenID use one account for multiple websites, and without using a password and authentication processes are outside the system / website Relying Party. Raises the question of how the level of security in this OpenID protocol. Therefore dilakukan penetration testing to determine the quality of security in the OpenID protocol. The purpose of this penetration testing is to find the vulnerability in OpenID Protocol and find the solution to fix that and prevent users from the dangers of the vulnerability. Based on the penetration testing that was done the process of allowing users become victims of Phishing. The best way to prevent phishing action is to not display the login page in the transfer process if the user has not been authenticated in Provider. Additionally, when a user account on Relying Party is stolen, then potentially token code stored on a Relying Party can also be stolen and this will harm the user data in Provider. To resolve this vulnerability, Providers must make a regulation to limit the websites that can use the OpenID protocol.

***Keywords**: Single Sign On, OpenID, phishing, session hijacking dan data security.*

(Halaman ini sengaja dikosongkan)

KATA PENGANTAR

Puji syukur kehadirat ALLAH SWT atas segala rahmat dan karunia-Nya, penulis dapat menyelesaikan penelitian ini dengan judul : **Analisis Potensi Celah Keamanan pada *Website Single Sign On* yang menggunakan Protokol *OpenID***

Penelitian ini disusun dalam rangka pemenuhan bidang riset di Jurusan Teknik Elektro ITS, Bidang Studi Teknik Komputer dan Telematika, serta digunakan sebagai persyaratan menyelesaikan pendidikan S-1. Penelitian ini dapat terselesaikan tidak lepas dari bantuan berbagai pihak. Oleh karena itu, penulis mengucapkan terima kasih kepada:

1. Ayah dan ibu yang sangat banyak membantu dalam semua hal, terutama dalam hal spiritual dan material. Karena berkat doa dari kedua orang tualah semua proses penelitian tugas akhir ini dapat berjalan dengan lancar.
2. Bapak Dr. Tri Arief Sardjono, ST., MT. selaku Ketua Jurusan Teknik Elektro, Fakultas Teknologi Industri, Institut Teknologi Sepuluh Nopember.
3. Bapak Christyowidiasmoro, ST., MT dan Bapak Reza Fuad Rachmadi, ST., MT, selaku dosen pembimbing yang selalu memberikan saran serta bantuan dalam penelitian ini.
4. Bapak Ibu dosen pengajar Jurusan Teknik Elektro,
5. Seluruh teman-teman asisten bidang Studi Teknik Komputer dan Telematika Teknik Elektro ITS.

Kesmpurnaan hanya milik ALLAH SWT, untuk itu penulis memohon segenap kritik dan saran yang membangun serta meminta maaf atas segala kekurangan yang ada dalam penulisan buku ini. Semoga penelitian ini dapat memberikan manfaat bagi kita semua. Amin.

Surabaya. Juli 2015

Penulis

(Halaman ini sengaja dikosongkan)

DAFTAR ISI

Daftar Tabel

(Halaman ini sengaja dikosongkan)

Daftar Gambar

Daftar Kode

(Halaman ini sengaja dikosongkan)

BIOGRAFI PENULIS

Akhirul Hajri lahir di Lubuk Jambi pada tanggal 30 Juni 1992. Menyelesaikan pendidikan SD di SDN 001 Pasar Lubuk Jambi pada tahun 2004, kemudian melanjutkan pendidikan SMP di SMPN1 Kuantan Mudik dan lulus pada tahun 2007 dan menyelesaikan pendidikan SMA di SMAN Plus Provinsi Riau pada tahun 2010. Setelah lulus SMA, penulis melanjutkan pendidikan S1 di jurusan Teknik Elektro Institut Teknologi Sepuluh Nopember (ITS). Penulis sangat tertarik dengan jaringan komputer dan pernah mengikuti pelatihan serta sertifikasi jaringan komputer yang diadakan oleh salah satu perusahaan telekomunikasi PT Huawei, yaitu Huawei Ceritified Datacom Associate (HCDA) pada tahun 2014.

BAB 1

PENDAHULUAN

1.1 Latar Belakang

Single *Sign On* merupakan sebuah metode dimana pengguna hanya sekali melakukan *login* untuk dapat mengakses *network resources* yang ada. Metode *single sign on* menggunakan pihak ketiga sebagai pengelola *username* dan *password* dari pengguna aplikasi. Pada dasarnya otentikasi *single sign on* hanya melibatkan satu *credential user* untuk *login* ke banyak aplikasi berbasis *website* setelah sebelumnya melakukan otentikasi pada salah satu aplikasi yang terintegrasi dengan sistem *Single Sign On*.

Salah satu protokol yang digunakan dalam *single sign on* adalah protokol *OpenID*. Pada *OpenID* ini terdapat 3 pihak yang terlibat yaitu pengguna, pihak client atau lebih dikenal sebagai *Relying Party* dan pihak *provider* sebagai otentikator/penyedia jasa otentikasi dan penyimpan data pengguna. Proses otentikasi protokol *OpenID* menggunakan jasa pihak ketiga dan berada diluar sistem penyedia layanan. Jadi ketika proses otentikasi, pengguna akan dialihkan menuju sistem *provider* untuk diotentikasi. Ketika pengguna telah diotentikasi, *provider* akan memberitahukan kepada *Relying Party* bahwa pengguna telah diotentikasi dan bisa menggunakan layanannya.

Pada protokol *OpenID* terjalin kerja sama antara *Relying Party* dan *Provider* dalam hal identitas dan data-data pengguna. dari kerja sama yang terjalin antara *provider* dan *Relying Party*, *Relying Party* akan mendapatkan hak untuk mengakses data-data pengguna yang ada di *provider*. Dan untuk keamanan dan kenyamanan pengguna, seharusnya tidak semua data-data pengguna bisa diakses oleh *Relying Party*, harus ada regulasi untuk membatasi *Relying Party* untuk mengakses data pengguna.

Karena protokol *OpenID* ini adalah protokol untuk proses otentikasi, faktor keamanana adalah hal yang paling utama dalam proses otentikasi ini. Data-data untuk proses otentikasi harus dalam kondisi aman dan tidak boleh diketahui orang lain. Untuk itu dibutuhkan kredibilitas yang tinggi untuk bisa menjalin kerja sama antara *Relying*

Party dan *Provider*. Serta jaminan sistem yang benar-benar aman, dikarenakan hal ini berkaitan dengan keamanan data pengguna. Selain itu, proses otentikasi pada *OpenID* yang menggunakan 1 akun untuk beberapa website, serta tanpa menggunakan sebuah kode sandi dan proses otentikasi berada diluar sistem/*website* penyedia layanan. menimbulkan pertanyaan bagaimana tingkat keamanan pada protokol *OpenID* ini. Untuk itu dilakukan pengujian tingkat keamanan dan analisa mendalam pada protokol *OpenID* untuk mencari tahu potensi celah keamanan yang bisa terjadi. Dengan mengetahui potensi yang terdapat dalam protokol *OpenID* kemudian dapat dicari solusi untuk mengatasi celah keamanan tersebut. Dan diharapkan pengguna aplikasi dapat terhindar dari tindakan kriminal (*cyber crime*) dan berbagai tindakan *hacking* lainnya.

1.2 Permasalahan

Pengunaan protokol *HTTP (Hyper Text Transfer protocol)* dalam proses otentikasi berpotensi menimbulkan celah keamanan yang besar. Potensi celah ini akan menimbulkan celah-celah keamanan yang lain. Akibat dari penggunaan protokol *HTTP*, dan tidak menggunakan saluran yang aman seperti *SSL (Secure Socket Layer)*, semua data otentikasi sangat terbuka dan tidak terenkripsi. Dan ini sangat berbahaya, karena orang lain dapat melihat parameter-parameter yang digunakan dalam proses otentikasi. Dan parameter-parameter tersebut dapat dimanfaatkan untuk membajak akun dari pengguna dengan berbagai metode *hacking*, seperti *session hijacking, parameter injection*, dan lain sebagainya.

Tidak hanya itu, proses *redirect*/pengalihan ke pihak *provider* juga mempunyai kerentanan terhadap tindakan *phishing*, yaitu dengan memalsukan halaman pihak *provider*. *Phishing* tentu sangat berbahaya karena jika pengguna terjebak didalamnya, maka peretas akan mendapatkan *username* dan *password* pengguna aplikasi.

Jika akun pengguna di *Relying Party* diretas orang lain akan berpotensi menimbulkan bahaya baru, yaitu ada potensi pencurian kode token yang diberikan *provider* kepada *Relying Party*. Dimana kode token ini berfungsi sebagai alat untuk mendapatkan hak untuk mengakses data pengguna di *provider*.

Setiap *provider* mempunyai regulasi sendiri tentang hak akses yang diberikan kepada *Relying Party*. Kelemahan *provider* bisa mengakibatkan ternjadinya pembobolan hak akses yang seharusnya tidak dimiliki oleh *Relying Party* menjadi bisa di akses.

1.3 Tujuan

Penelitian tugas akhir ini bertujuan menganilisis potensi celah keamanan pada aplikasi website yang menggunakan metode SSO, khususnya yang menggunakan protokol *OpenID*. Dengan ditemukannya celah keamanan yang terdapat pada penggunaan protokol OpenID ini, dapat dicari solusi untuk mengatasinya. Dengan begitu diharapkan dapat meminimalisir tindakan kriminal (*cyber crime*) dan tindakan pencurian akun lainnya, serta diharapkan juga dapat membantu administrator *website* mengetahui celah keamanan pada sistem yang dikelolanya dan kemudian meningkatkan kualitas keamanan pada sistemnya.

1.4 Batasan Masalah

Batasan masalah dalam pengerjaam tugas akhir ini adalah proses analisis yang hanya terbatas pada:

1. *Website single sign on* yang menggunakan protkol *OpenID*
2. Hasil pengujian merupakan hasil yang didapat selama penelitian tugas akhir ini, jadi jika setelah laporan Tugas Akhir ini hasil pengujian tidak berfungsi lagi, kemungkinan sistem telah diperbaiki dan kualitas keamanan sistem telah ditingkatkan
3. Sampel yang digunakan untuk objek pengujian hanya untuk *provider* besar, seperti Google, Facebook dan Wordpress.

1.5 Sistematika Penulisan

Sistematika penulisan laporan Tugas Akhir ini dibagi dalam 4 bab, masing-masing bab diuraikan sebagai berikut:

1.Bab 1

Merupakan pendahuluan yang berisi latar belakang, tujuan permasalahan, batasan masalah, metodelogi dan sistematika penulisan.

2.Bab 2

Teori penunjang dan kepustakan yang digunakan dalam proses tugas akhir ini.

3.Bab 3

Desain sistem dan sistematika pengujian keamananan.

4.Bab 4

Hasil analisa dan pengujian keamanan sistem serta solusi untuk mengatasi permasalahan keamanan pada sistem.

BAB 2

DASAR TEORI

2.1 Otentikasi

Otentikasi adalah sebuah metode untuk memverifikasi seorang pengguna yang mempunyai hak untuk mengakses suatu sistem terntentu. Proses otentikasi dibutuhkan untuk memperkuat keamanan sistem, untuk itu ada beberapa faktor dalam proses otentikasi seorang pengguna, yaitu:

a. Sesuatu yang diketahui oleh pengguna, misalnya seperti kata sandi dan kode *PIN(Personal Idetification Number)*.
b. Sesuatu yang dimiliki pengguna, misalnya seperti kode token atau sebuah sertifikat perangkat lunak.
c. Sesuatu yang ada pada diri pengguna, misalnya seperti sidik jari atau retina dari pengguna.

Jika semua faktor tersebut diterapkan dengan benar, maka seorang peretas akan lebih sulit untuk melakukang penyerangan/peretasan terhadap suatu sistem, karena terdapat satu faktor yang tidak dimiliki oleh seorang peretas, seperti sesuatu yang ada pada diri pengguna, yaitu sidik jari. Dengan demikian informasi rahasia mennjadi lebih aman.

Ada beberapa kombinasi factor-faktor dalam proses otentikasi yang biasa digunakan untuk meningkatkan kualitas keamanan sistem, seperti :

a. Sebuah kartu pintar yang digunakan bersama dengan pembaca kartu pintar (*smartcard reader*) dan kode *PIN*.
b. Sidik jari yang digunakan bersama kata sandi atau kode *PIN*
c. Sertifikat perangkat lunak yang tersimpan dikomputer yang hanya dapat diakses dengan sebuah kata sandi atau kode *PIN*. [11]

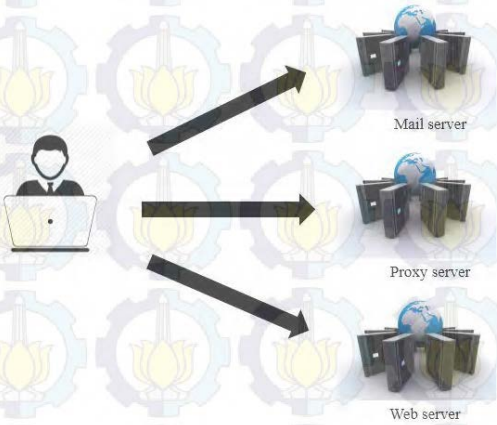
Semakin banyak kombinasi faktor yang digunakan akan membuat peluang peretas untuk melakukan pencurian data semakin kecil.

2.2 *Single sign on*

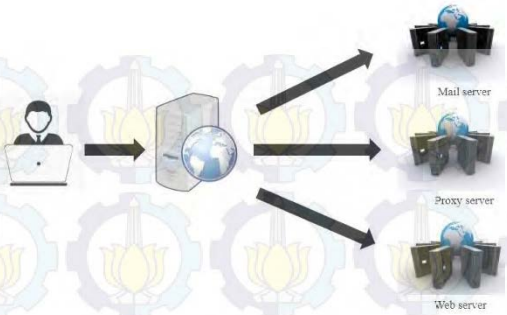
Single sign on adalah sebuah metode yang memungkinkan pengguna dapat menggunakan beberapa layanan sekaligus hanya dengan mengguna satu *username*/*userid* dan *password* saja. Pada *single sign on* ini, proses otentikasi hanya sekali, kemudian proses otentikasi akan dilakukan dengan otomatis ketika pengguna membuat layanan lain melalui sebuah *session* [1]

Jadi, jika pengguna menggunakan metode *single sign on* pada *website*, maka pengguna hanya perlu melakukan satu kali *login* dan kemudian pengguna dapat menggunakan beberapa layanan yang lain tanpa perlu memasukan *password* lagi.

Untuk perbandingan antara metode *single sign on* dengan metode *sign on* dapat dilihat pada gambar 2.1 dan gambar 2.2



Gambar 2.1 Metode *Sign On*



Gambar 2.2 Metode *Single sign on*

2.3 *OpenID*

OpenID adalah sebuah protokol terbuka yang memungkin pengguna untuk dapat menggunakan akun yang telah ada untuk *login* pada beberapa *website* tanpa perlu membuat *password* baru.[13] Jadi, dengan *OpenID*, pengguna dapat *login* pada *website* lain dengan akun yahoo miliknya. [12]

Pada protokol *OpenID* ini, terdapat 3 pihak yang terlibat, yaitu:

a. *Identity Provider*
   Identity *provider* adalah *website* yang bertugas sebagai penyedia identitas pengguna, misalnya Google, Yahoo, Facebook, dan Twitter.
b. *Relying Party*
   Relying Party adalah *website* yang menggunakan jasa identity *provider*.
c. Pengguna

Pengguna dapat memilih informasi apa saja yang akan diberikan atau informasi apa saja yang dapat diakses oleh *website* yang dikunjugi oleh pengguna. Sedangkan untuk *password* pengguna hanya diketahui oleh *provider*, dan *provider* yang mengelola identitas dari pengguna, selain itu tidak *website* yang mengetahui *password* pengguna. [13]

Jadi, dengan *OpenID* ini, pengguna dapat *login* pada suatu *website* dengan menggunakan akun di*website* lain. Misalnya, pengguna A dapat *login website* livejournal.com dengan menggunakan akun Google yang

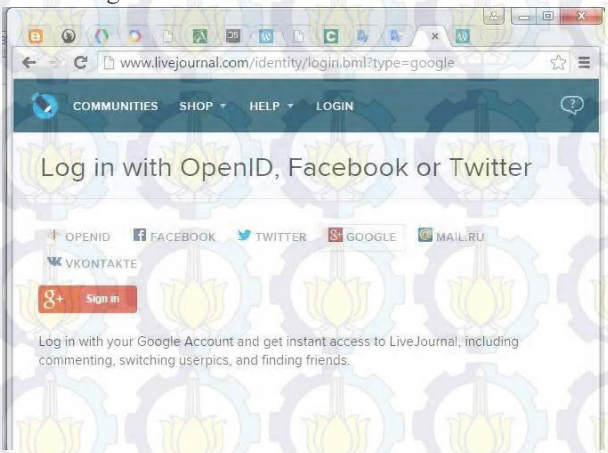
dimilikinya. Tapi dalam hal ini, *website* livejournal.com hanya mendapatkan beberapa hak untuk mengakses data pengguna dan hak tersebut harus sudah disetujui oleh pengguna A. livejournal.com tidak mengetahui *password* akun Google pengguna A. Dengan metode *login* seperti ini, seorang pengguna tidak perlu lagi mengingat banyak *userid* dan *password* untuk *login* pada beberapa *website* yang berbeda.

Saat ini *OpenID* memiliki lebih dari satu miliar pengguna *OpenID* dan lebih dari 50.000 *website* menggunakan *OpenID* untuk *login*. Beberapa perusahaan besar menggunakan *OpenID*, termasuk Google, Facebook, Yahoo !, Microsoft, AOL, MySpace, Sears, Universal Music Group, France Telecom, Novell, Sun, Telecom Italia, dan masih banyak lagi. [13]

2.3.1 Prosedur Penggunaan Protokol *OpenID*

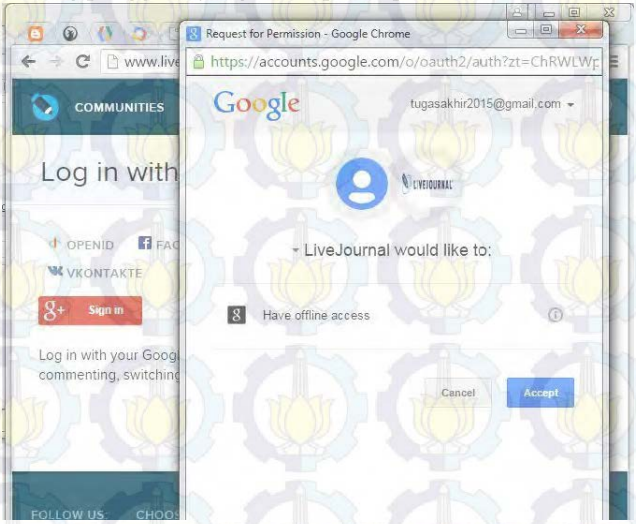
Berikut adalah proses ketika pengguna *login* pada *Website* livejournal.com dengan menggunakan *OpenID* (menggunakan *provider* Google dan wordpress)

1. Menggunakan Google
   a. Pengguna memilih *login* ke *Website* livejournal.com dengan akun Google+.



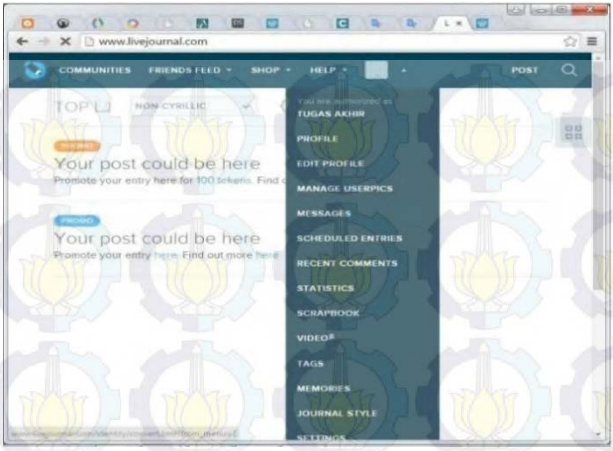
Gambar 2.3 Tombol *Login* Dengan Goole

b. Kemudian pengguna akan dialihkan ke *website* Google untuk otentikasi, serta meminta persetujuan pengguna bahwa *website* livejournal.com akan mengakses informasi tentang pengguna yang tersimpan di *provider*.



Gambar 2.4 Permintaan izin mengakses data pengguna

c. Setelah pengguna memberikan persetujuan kepada *Relying Party*, selanjutnya pengguna bisa menggunakan layanan pada *Website* livejournal.com



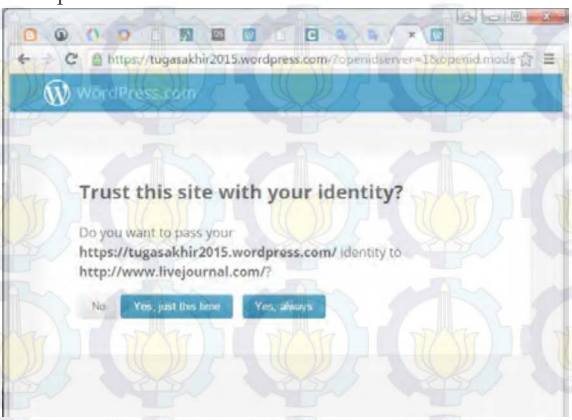
Gambar 2.5 Pengguna Berhasil *login* dengan Google

2. Menggunakan url wordpress.
   a. Pengguna memilih *login* ke *Website* livejournal.com dengan *URL* http://tugasakhir2015.wordpress.com

Gambar 2.6 Kolom *login* dengan Wordpress

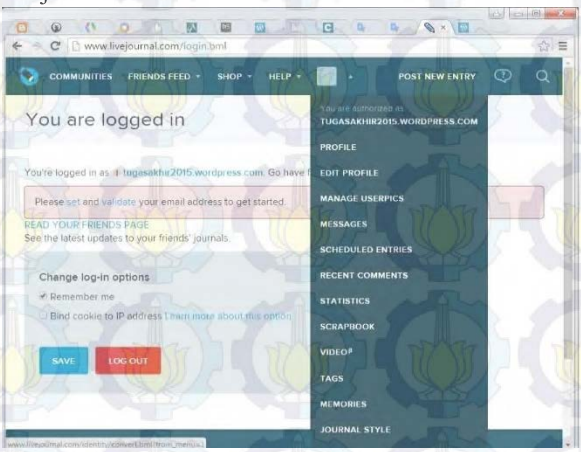
   b. Kemudian, pengguna akan dialihkan ke *website* wordpress.com untuk otentikasi dan meminta persetujuan pengguna bahwa *website* livejournal.com akan mengakses

informasi tentang pengguna yang tersimpan di wordpress.com



Gambar 2.7 Verifikasi data pengguna

c. Setelah itu, pengguna akan dialihkan kembali ke *website* livejournal.com



Gambar 2.8 Pengguna Berhasil *Login* dengan Wordpress

2.3.2 Proses Otentikasi dan Cara Kerja Protokol *OpenID*

Adapun prinsip kerja *OpenID* secara teknis dan proses otentikasi di lakukan adalah sebagai berikut:

a. Pengguna memasukan *URL OpenID* pada kolom yang tersedia (misalnya: http://alice.my*openid*.com) di *website Relying Party*.
b. Untuk mengetahui *Provider* yang digunakan oleh pengguna, *Relying Party* menggunakan *URL* yang dikirimkan oleh Pengguna. Pada umumnya *URL identifier OpenID* terletak pada bagian <head> dari kode html *Website Relying Party*, lebih spesifiknya terletak pada tag <*link*>. Didalam tag <*link*> ini terdapat atribut rel="`openid2.provider`" serta link dari *OpenID provider* yang digunakan.
c. Setelah *Relying Party* mengetahui *OpenID provider* yang digunakan, maka *Relying Party* akan membuat sebuah prosedur keamanan dengan *OpenID Provider* menggunakan algoritma Diffie-Hellman yaitu dengan membuat *security key*. *Security key* ini akan digunakan untuk memverifikasi dalam pertukaran data antara keduanya.
d. Kemudian *Relying Party* akan mengalihkan pengguna ke *Website OpenID* menggunakan sebuah *URL identifier* yang terdiri dari semua parameter yang dibutuhkan untuk proses otentikasi (`openid.mode=chceckid setup`). Parameter-parameter tersebut adalah identias dari pengguna *OpenID* (`openid.identity`), asosiasi (`openid.assoc`) dan *URL* yang akan mengalihkan pengguna kembali ke *Website Relying Party* (`openid.return to`). Selain itu, biasanya *Relying Party* juga akan meminta parameter tambahan kepada *OpenID provider* seperti nama lengkap, jenis kelamin, dan *email*.
   Berikut adalah contoh parameter yang digunakan:

```
GET      /openid-auth?OpenID.mode=
checked_setup&OpenID.identity
=http://alice.myid.org&openid
.return_to=http://www.example.org
/openid-login&openid.assoc_handle
=xxxxxxxxxxxxxxx
HTTP/1.1
Host: myid.org
```

Kode 2.1 Parameter dalam otentikasi

e. Selanjutnya pengguna akan melakukan proses otentikasi di *Website OpenID provider* (memasukan penggunaname dan *password*) kemudian memberikan persetujuan dari permintaan *Relying Party* untuk mengakses informasi pengguna tersebut. Demi kenyamaan pengguna, biasanya *Relying Party* akan membuat sebuah *HTTP Cookies* untuk memudahkan pengguna kedepannya jika akan kembali menggunakan *OpenID*.
f. Setelah pengguna melakukan proses otentikasi dan pemberian izin akses (otorisasi), kemudian pengguna akan di alihkan kembali ke *Website Relying Party* melalui sebuah *URL* yang terdiri dari parameter *authentication response* (`openid.mode=id res`), indentitas pengguna *OpenID* (`openid.identity`), alamat *Relying Party* (`openid.return to`), parameter *nonce* (`OpenID.response nonce`) dan *cryptographic signature* (`openid.sig`) [6]

Jadi, hampir semua proses otentikasi pada *OpenID* dapat dilihat langsung, karena semua parameter tersimpan di dalam *redirect-URL*. untuk menghindari hal yang tidak diinginkan, *Provider* akan memvalidasi *redirect-URL* yang di terimanya dari *Relying Party* berdasarkan *redirect-URL* yang sudah terdaftar sebelumnya.

Saat ini *OpenID* masih dalam tahap adopsi dan akan semakin populer seiring dengan banyaknya perusahaan besar yang mulai menerima dan menjadi penyedia layanan (*provider*) *OpenID*.

Berdasarkan data pada Desember 2009 diperkirakan sudah lebih dari 1 Miliar pengguna *OpenID* menggunakan *URL* (*URL enabled*) dengan sekitar kurang lebih 9 juta situs yang mendukung *login* melalui *OpenID*.

Berikut beberapa penyedia layanan *OpenID*

Tabel 2.1 Daftar *Provider OpenID*

Penyedia Layanan	Format URL
Google	Google.com/account/me
Yahoo	*OpenID*.yahoo.com
My*OpenID*	username.my*OpenID*.com
LiveJournal	username.livejournal.com
AOL	*OpenID*.aol.com/username
Wordpress	username.wordpress.com
Blogspot	username.blogspot.com
Verisign	username.pip.verisignlabs.com
Google Profile	google.com/profiles/username

2.4 Hypertext Transfer Protocol (HTTP)

Pengertian *HTTP* adalah Singkatan dari *Hyper Text Transfer Protocol*, protokol yang mendasari oleh *World Wide Web*. Dalam pengertian *HTTP* menetapkan bagaimana pesan diformat dan ditransmisikan, dan apa tindakan dari *webserver* dan *browser* sebagai respon pada berbagai perintah.

Secara khusus *HTTP* dapat diartikan sebagai pesan yang berbentuk format dan dapat dikirim melalui sebuah *server* ke *Client*. *HTTP* juga berfungsi sebagai alat yang mengatur bentuk dan aksi apapun yang dilakukan oleh *Web Server* juga *Web Browser* untuk direspon atas perintah yang ada pada protokol *HTTP*.

2.5 *Packet Sniffing*

Packet sniffing adalah metode penyadapan data-data yang berada di jaringan, dimana, dengan menggunakan metode ini, seorang peretas dapat melihat data pengguna lain yang dikirim dijaringan. Adapun perangkat yang digunakan untuk melakukan *sniffing* disebut dengan

packet sniffer. *Packet sniffer* ini bekerja pada layer *Transmission Control Protocol/Internet Protocol (TCP/IP)* [14].

Paket sniffing dapat dilakukan tidak hanya pada jaringan lokal saja (*LAN*), tetapi juga bisa dilakukan pada jaringan yang lebih luas(*WAN*). *Packet sniffing* ini umumnya digunakan untuk menganalisa paket data yang dikirimkan dijaringan dengan tujuan untuk mengetahui permasalahan yang terjadi dijaringan. Namun dalam perkembangannya, *packet sniffing* lebih sering digunakan untuk tujuan kejahatan, seperti untuk tindakan *hacking*, mencuri akun pengguna, dan mencuri data-data penting pengguna. Misalnya ketika terjadi komunikasi antara pengguna A dan pengguna B, maka dengan metode *packet sniffing*, semua data komunikasi antar A dan B dapat diketahui. Sama halnya jika seorang pengguna *login* pada suatu *website*, dengan metode *packet sniffing* ini, userid dan *password* pengguna dapat diketahui.

Packet sniffing tidak hanya bersifat pasif, dimana peretas hanya bertindak sebagai pendengar tanpa melakukan apa-apa terhadap koneksi antara 2 pengguna. Tetapi *packet sniffing* juga dapat bersifat aktif, dimana peretas dapat mengubah data-data pengguna yang dikirim dijaringan, sehingga data yang diterima oleh pengguna lain menjadi tidak sesuai dengan data asli yang dikirim sebelumnya.

2.6 *Session Hijacking*

Session hijacking adalah sebuah teknik *hacking* dengan tujuan untuk mengambil alih akun pengguna lain dengan mencuri sessi yang tersimpan didalam *cookie*. Ketika pengguna mengunjungi sebuah *website*, maka *website* akan mengirimkan *cookie* dan cookie tersebut akan tersimpan pada *browser* yang digunakan oleh pengguna. *Cookie* yang dikirimkan inilah yang kemudian dicuri peretas melalui *proses packet sniffing*, kemudian *cookie* yang sudah didapatkan disimpan ke *browser* yang digunakan oleh peretas.

Ada banyak cara yang dilakukan untuk mendapatkan cookie, seperti melakukan *packet sniffing* ataupun dengan mencurinya langsung pada *browser* yang digunakan pengguna. Dengan metode *session hijacking* ini, seorang peretas dapat mencuri akun pengguna tanpa perlu mengetahui *username* dan *password* pengguna.

2.7 Cookie

Cookie adalah berkas yang dibuat oleh *website* yang dikunjungi pengguna yang digunakan untuk menyimpan informasi pengguna, seperti informasi profil dari pengguna [15]. *Cookie* ini tersimpan *browser* yang digunakan oleh pengguna. Jadi ketika pengguna membuka *website* yang pernah dikunjungi sebelumnya, maka *browser* akan mengirimkan *cookie* yang sesuai kepada *website* tersebut. Dengan cara ini, *website* dapat menampilkan informasi yang sesuai dengan pengaturan pengguna.

Cookie dapat juga disebut sebagai *ID card* pengguna saat terhubung pada *website* tertentu. [9]. Cookie dapat menyimpan berbagai jenis informasi, termasuk di antaranya informasi pribadi seperti nama, alamat rumah, alamat *email*, atau nomor telepon pengguna. Akan tetapi informasi ini hanya akan disimpan jika pengguna pernah memberikan informasi ini kepada *website* tersebut. *Website* tidak dapat mengakses informasi yang tidak pernah diberikan pengguna kepada *website* tersebut, dan *website* juga tidak dapat mengakses berkas lainnya pada komputer pengguna..

2.8 Phishing

Phishing adalah suatu metode *hacking* yang digunakan untuk mendapatkan informasi pribadi pengguna seperti userid, *password* dan data-data rahasia lainnya dengan cara menyamar sebagai pihak lain. *Phishing* berasalah dari kata bahasa inggris *fishing* yang mempunyai arti memancing, memancing dalam hal ini adalah meancing pengguna untuk memberikan informasi rahasia seperti *password*, kode *PIN* dan lain sebagainya [16]

Ada banyak cara untuk melakukan *phishing*, seperti mengirimkan informasi palsu ke *email* pengguna, mengggunakan sebuah tautan menuju halaman palsu dan lain sebagainya, yang tujuan utamanya adalah untuk mendapatkan *userid* dan *password* pengguna.

2.9 Kode Token Akses

Kode token adalah sebuah kode acak yang mengidentifikasi seorang pengguna atau halaman dan dapat digunakan oleh aplikasi untuk melakukan panggilan *API(Application Programming Interface)*. Kode token ini meyimpan informasi tentang masa berlaku kode token tersebut serta aplikasi yang diakses oleh kode token tersebut.[17]

Dalam penggunaannya, ada beberapa jenis kode token, yaitu

a. Kode token pengguna

Kode token jenis ini digunakan ketika aplikasi(dalam hal ini *website Relying Party*) ingin mengakses data-data pengguna, seperti membaca, memodifikasi atau menulis data pengguna tertentu atas nama mereka. Kode token ini didapatkan melalui dialog *login*. Untuk alur *Relying Party* mendapatkan kode token dapat dilihat pada gambar 2.9

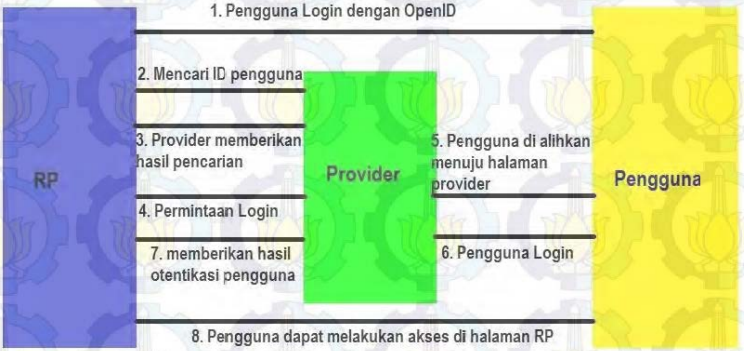
Gambar 2.9 Alur *Relying Party* mendapatkan kode token penguna [17]

b. Kode token aplikasi
   Kode token aplikasi ini diperlukan untuk memodifikasi dan membaca pengaturan aplikasi. Kode token jenis ini dihasilkan secara rahasia oleh aplikasi dan *provider*.
c. Kode token halaman
   Kode token ini hampir sama dengan kode token pengguna. Untuk mendapatkan kode token ini, terlebih dahulu harus mendapatkan kode token pengguna dan hak untuk mengakses *manage_pages*. Setelah itu barulah kemudian mendapatkan kode token halaman ini.
d. Kode token klien
   Kode token klien adalah sebuah kode pengenal yang dapat ditanamkan kedalam kode biner aplikasi *mobile* dan aplikasi desktop yang digunakan untuk mengidentifikasi aplikasi pengguna. Kode token klien digunakan untuk mengakses aplikasi setingkat *API*, tapi dengan hak akses yang terbatas.

BAB 3

DESAIN DAN SISTEMATIKA PENGUJIAN

3.1 Desain Sistem



Gambar 3.1 Arsitektur Protokol *OpenID*

Adapun prinsip kerja *OpenID* secara teknis dan proses otentikasi di lakukan adalah sebagai berikut:

a. Pengguna memasukan *URL OpenID* pada kolom yang tersedia (misalnya: http://alice.my*openid*.com) di *website Relying Party*.
b. Untuk mengetahui *provider* yang digunakan oleh pengguna, *Relying Party* menggunakan *URL* yang dikirimkan oleh pengguna. Pada umumnya *URL identifier OpenID* terletak pada bagian <*head*> dari kode *html website Relying Party*, lebih spesifiknya terletak pada tag <*link*>. Didalam *tag* <*link*> ini terdapat atribut rel="`openid2.provider`" serta *link* dari *openid provider* yang digunakan.
c. Setelah *Relying Party* mengetahui *OpenID provider* yang digunakan, maka *Relying Party* akan membuat sebuah prosedur keamanan dengan *OpenID Provider* menggunakan algoritma Diffie-Hellman yaitu dengan membuat *security key*.

Security key ini akan digunakan untuk memverifikasi dalam pertukaran data antara keduanya.

d. Kemudian *Relying Party* akan mengalihkan pengguna ke *website OpenID* menggunakan sebuah *URL identifier* yang terdiri dari semua parameter yang dibutuhkan untuk proses otentikasi (`OpenID.mode=chceckid setup`). Parameter-parameter tersebut adalah identias dari pengguna *OpenID* (`openid.identity`), asosiasi (`openid.assoc`) dan *URL* yang akan mengalihkan pengguna kembali ke *website Relying Party* (`openid.return to`). Selain itu, biasanya *Relying Party* juga akan meminta parameter tambahan kepada *OpenID provider* seperti nama lengkap, jenis kelamin, dan *email*.
   Berikut adalah contoh parameter yang digunakan:

```
GET      /openid-auth?OpenID.mode=
checked_setup&OpenID.identity
=http://alice.myid.org&openid
.return_to=http://www.example.org
/openid-login&openid.assoc_handle
=xxxxxxxxxxxxxxx
HTTP/1.1
Host: myid.org
```

Kode 3.1 Parameter dalam otentikasi

e. Selanjutnya pengguna akan melakukan proses otentikasi di *website OpenID provider* (memasukan *username* dan *password*) kemudian memberikan persetujuan dari permintaan *Relying Party* untuk mengakses informasi pengguna tersebut. Demi kenyamaan pengguna, biasanya *Relying Party* akan membuat sebuah *HTTP cookie* untuk memudahkan pengguna kedepannya jika akan kembali menggunakan *OpenID*.

f. Setelah pengguna melakukan proses otentikasi dan pemberian izin akses (otorisasi), kemudian pengguna akan di alihkan kembali ke *website Relying Party* melalui sebuah *URL* yang

terdiri dari parameter *authentication response* (`openid.mode=id res`), indentitas pengguna *OpenID* (`openid.identity`), alamat *Relying Party* (`OpenID.return to`), parameter nonce (`openid.response nonce`) dan *cryptographic signature* (`openid.sig`)[6].

Objek dalam penelitian ini adalah *website Relying Party* yang menggunakan *provider* Google dan Facebook. *Website Relying Party* ini dipasang pada server lokal (*localhost*) dan juga pada *server online*. Selain *website Relying Party* ini, pada penelitian ini juga menguji *website Relying Party* yang sudah ada seperti *website* livejournal.com. idhostinger.com, udemy.com dan mediafire.com.

Dan untuk tahapan pembuatam *website Relying Party* adalah sebagai berikut:

a. Pembuatan Google *API* dan Facebook *API*

Langkah pertama dalam membuat *website Relying Party* adalah dengan membuat aplikasi Google dan Facebook untuk mendapatkan kode *Oauth Client ID/App ID* dan *Client Secret/App Secret* pada *website* console.developers.google.com dan developers.facebook.com. untuk contoh hasil pembuatan aplikasi pada Google dan Facebook dapat dilihat pada gambar 3.2 dan gambar 3.3

Gambar 3.2 Data Aplikasi Google

Gambar 3.3 Data Aplikasi Facebook

b. Membuat basisdata

Setelah membuat aplikasi pada *provider* dan mendapatkan *client id/app id* dan *client secret/app secret*, kemudian dilanjutkan dengan membuat basisdata sebagai media penyimpanan data-data pengguna yang didapat dari *provider*. Tabel untuk setiap *provider* berbeda, ini berdasarkan dari data yang didapat dari *provider* tersebut. Untuk lebih detail bisa dilihat pada tabel 3.1 dan tabel 3.2.

Tabel 3.1 Data pengguna untuk *provider* Google.

No	Nama	Keterangan
1	user_id	Nomor registrasi pengguna
2	name	Nama pengguna
3	email	Alamat email pengguna
4	password	*Password* pengguna
5	social_id	Id pengguna di *provider*
6	picture	Foto pengguna
7	created	Tanggal pengguna mendaftar

Untuk *provider* Facebook, digunakan 2 buah tabel, yaitu untuk menyimpan data pengguna yang didapat dari Facebook dan tabel untuk menyimpan kode token.

Tabel 3.2 Data pengguna untuk *provider* Facebook

No	Nama	Keterangan
1	userid	Id pengguna di *provider*
2	nama	Nama pengguna
3	email	Alamat *email* pengguna
4	gender	Jenis kelamin pengguna

Tabel 3.3 Tabel Kode Token

No	Nama	Keterangan
1	userid	Id pengguna di *provider*
2	akses_token	Kode token

c. Membuat berkas konfigurasi.
Data untuk koneksi ke basisdata serta data *client id, client secret,* alamat *website* dan alamat pengalihan ditambahkan dalam berkas konfigurasi.

d. Menambahkan pustaka *oauth* yang digunakan oleh *provider*
Dalam *website* ini, digunakan 2 buah *Software Development Kit (SDK)* yaitu *PHP SDK* dari Google dan Facebook *Javascript SDK*.

e. Menyimpan data pengguna ke basisdata dan membuat sesi.
Ketika pengguna login dengan *OpenID*, maka *provider* akan memberikan data-data pengguna dan data tersebut akan disimpan dalam basisdata server *website Relying Party*. Dari data tersebut akan dibuat sesi untuk pengguna. Untuk lebih jelas bisa dilihat pada kode 3.2

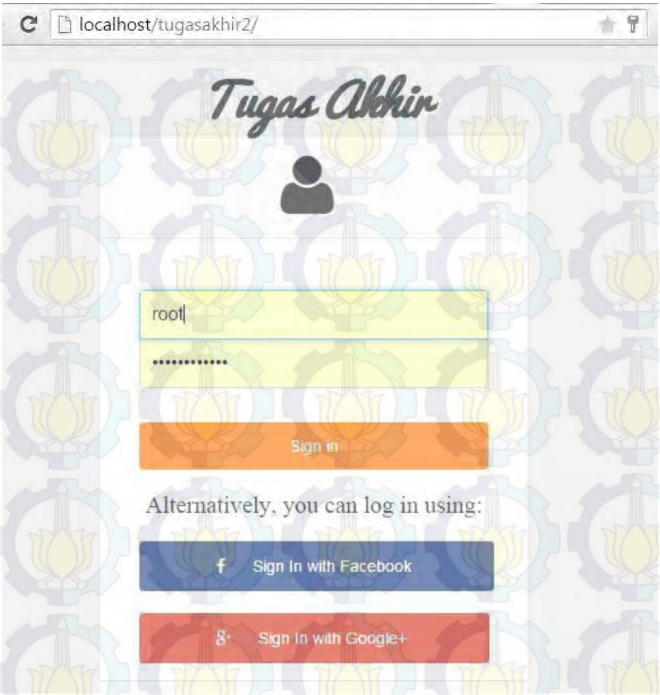
```
$cek_userid=mysql_num_rows(mysql_query
 ("SELECT userid FROM fb_data WHERE
userid='$userid'"));
if ($cek_userid > 0){
    $query=mysql_query("select * from
fb_data where userid='$userid' ") or
die(mysql_error());
    $xxx=mysql_num_rows($query);
      if($xxx==TRUE){
      $_SESSION['logged_in']=$userid;
            $_SESSION['name']=$nama;
      header("location:home.php");
      }
  }
```

Kode 3.2 Kode untuk memeriksa pengguna ketika *Login* dengan *OpenID*

Proses otentikasi terbagi menjadi 2 bagian, pertama ketika pengguna sudah pernah mendaftar sebelumnya pada *website Relying Party* dan kedua ketika pengguna baru pertama kali mendaftar pada *website Relying Party*. Jika pengguna sudah pernah terdaftar, data langsung diambil dari *basisdata* untuk dibuat sesi baru. Tapi jika pengguna belum terdaftar, maka data dari *provider* terlebih dahulu akan disimpan dalam *basisdata* kemudian diambil untuk dibuat sesi baru.

f. Hasil

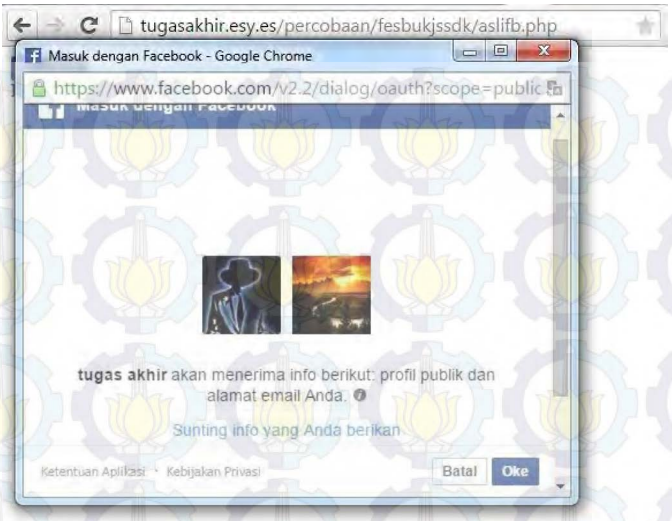
Setelah menambahkan semua pustaka dan *SDK*, dan semua proses konfigurasi serta penambahan fitur lainnya, maka *website Relying Party* sudah bisa digunakan. Dan untuk hasil dari *website Relying Party* dapat dilihat pada gambar 3.4



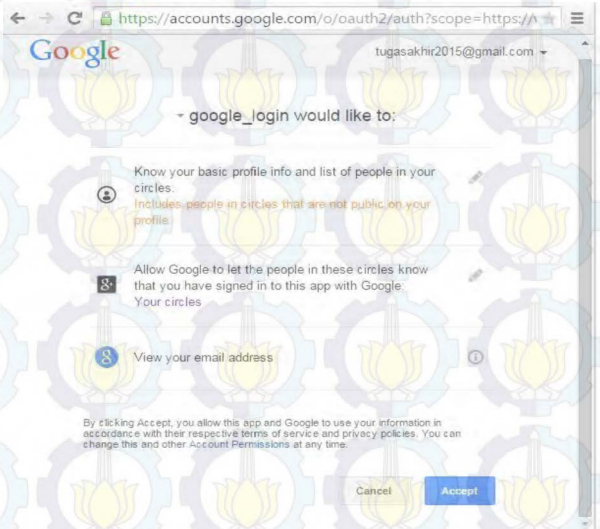
Gambar 3.4 Halaman Awal *website Relying Party*

3.2 Simulasi Sistem

Setelah semua konfigurasi dan pengaturan pada *website Relying Party* selesai, kemudian *website* tersebut akan dicoba untuk *login* dengan menggunakan *provider* Facebook dan Google. Tahapan proses *login* dengan *OpenID* sama dengan langkah-langkah yang dijelaskan pada sub bab 3.1.2. proses permintaan izin aplikasi untuk mengakses data pengguna di *provider* dapat dilihat pada gambar 3.5 dan gambar 3.6



Gambar 3.5 Permintaan izin aplikasi untuk mengakses data pengguna di Facebook



Gambar 3.6 Permintaan izin aplikasi untuk mengakses data pengguna di Google

Setelah pengguna memberi persetujuan kepada aplikasi, maka *provider* akan memberikan data-data pengguna yang diminta oleh aplikasi. Semua data tersebut akan disimpan didalam basisdata *website Relying Party*. Data yang tersimpan dalam *basisdata* dapat dilihat pada gambar 3.7

user_id	name	email	password	social_id	picture
1	ikan koi	goodmonday007@gmail.com		110778121099310401040	https://lh5.googleusercontent

userid	nama	email	gender
998668940156805	Akhirul	jhontheripper414@gmail.com	male

userid	akses_token
998668940156805	CAALDcbteCvsBAMIWldQRzh4PX1JyE1QmCvM0kb9ZACy1IfHpi...

Gambar 3.7 Data pengguna yang tersimpan di basisdata *Relying Party*

3.3 Sistematika Pengujian Sistem

Proses penelitian tugas akhir ini diawali dengan melakukan studi literatur mengenai protokol *OpenID* untuk mengetahui bagaimana prinsip kerja dari protokol *OpenID* ini. Kemudian dilakukan analisis prinsip kerja protokol *OpenID*, dengan tujuan untuk mencari tahu potensi celah keamanan yang ada. Berdasarkan hasil analisis, didapat hipotesa awal tentang potensi celah keamanan yang terdapat pada protokol *OpenID*. Dan dari hipotesa tersebut, dibuat perencanaan pengujian yang akan dilakukan. Pengujian dilakukan untuk membuktikan kebenaran hipotesa sebelumnya. Kemudian pembuatan *website Relying Party* sebagai objek pengujian. *Website Relying Party* tersebut menggunakan *provider* Facebook dan Google. Selain *website Relying Party* yang dibuat sendiri, objek pengujian pada penelitian ini juga menggunakan *website*-*website* umum yang juga menggunakan protokol *OpenID*, seperti *website* toko *online*, dengan tujuan untuk mengetahui validitas data yang didapat pada *website Relying Party* yang dibuat sendiri. Dari pengujian yang telah dilakukan, didapat beberapa celah keamanan yang ada pada protokol *OpenID*. Berdasarkan celah keamanan yang ditemukan kemudian dicari solusi untuk mengatasi permasalahan akibat celah keamanan tersebut.

Gambar 3.8 Metodelogi penelitian

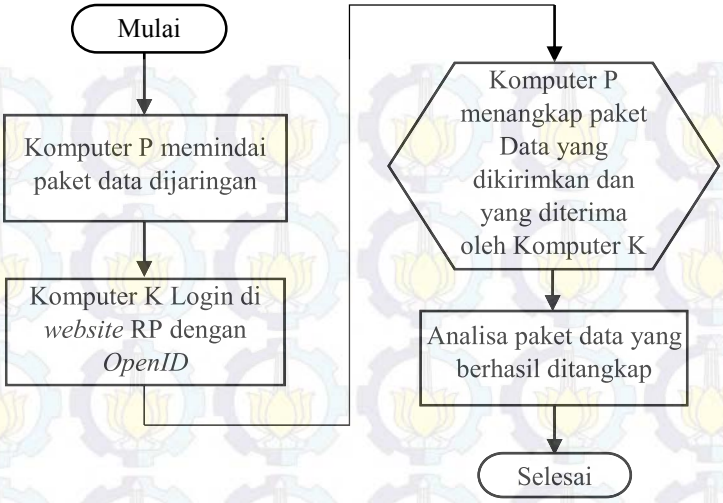
3.3.1 Pengujian Protokol Transfer Paket Data

Dalam komunikasi dan pertukaran data yang terjadi dalam *OpenID* menggunakan protokol *HTTP (Hypertext Transfer Protocol)*. penggunaan protokol *HTTP* tanpa dibungkus dengan jalur yang aman (*SSL*) memungkinkn semua data yang dikirimkan dijaringan bisa dilihat oleh pihak lain. Untuk membuktikan hal tersebut, akan dilakukan pemindaian paket data dijaringan dengan tujuan untuk menangkap paket data yang dikirim dalam otentikasi pada *OpenID*.

Perlengkapan yang digunakan dalam pengujian ini adalah:

a. Komputer penguji P
b. Komputer korban K
c. Jaringan *LAN*
d. Aplikasi *wireshark*

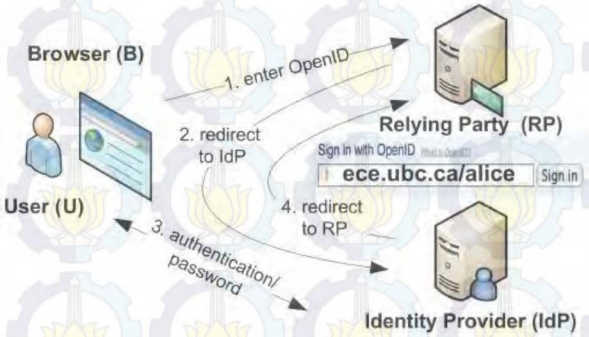
Untuk skema pengujian dapat dilihat pada gambar 3.9



Gambar 3.9 Skema pengujian protokol HTTP

3.3.2 DNS Spoofing

Proses Otentikasi pada *OpenID* melalui sebuah proses pengalihan dari *website Relying Party* menuju *website provider*. Untuk lebih jelas bisa dilihat pada gambar 3.10

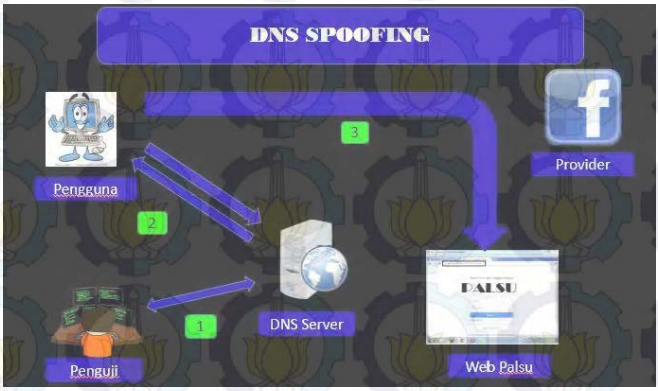


Gambar 3.10 Proses otentikasi *OpenID*[3]

Dari gambar 3.11 terlihat, pada nomor proses nomor 2, pengguna dialihkan menuju *website provider*. Proses pengalihan ini berpotensi

menimbulkan bahaya ketika pengguna dialihkan tidak menuju *website provider*, tetapi *website* lain. Sebuah pengujian akan dilakukan untuk membuktikan potensi celah keamanan pada proses pengalihan ini, yaitu dengan melakukan *DNS Spoofing* pada jaringan yang digunakan pengguna *OpenID*.

DNS Spoofing adalah teknik khusus yang digunakan untuk mengambil alih *DNS server* sehingga *DNS* atau layanan yang menyimpan nama domain dan *IP address* sebuah situs akan dialihkan ke komputer *webserver* lain. Secara umum, proses *DNS Spoofing* dapat dilihat pada gambar 3.11



Gambar 3.11 Cara kerja *DNS Spoofing*

3.3.3 Meretas *Website Relying Party*

Proses otentikasi pada *OpenID* terjadi dipihak ketiga, yaitu *provider*, bukan pada *Relying Party*. Proses ini bisa menjadi ancaman buat pengguna.Ancaman yang bisa terjadi adalah ketika peretas merubah link referensi pada atribut tombol *login* dengan *OpenID*, misalnya jika peretas merubah tautan asli tombol "Sign in with Google" dengan tautan halaman *login* yang dibuat mirip dengan halaman *login* Google. Selain itu, tautan referensi *OpenID* yang sangat panjang juga membuat potensi celah ini semakin besar. Untuk lebih jelas tentang tautan referensi pada tombol "Sign in with Google" bisa dilihat pada kode 3.3

```
https://accounts.google.com/o/oauth2/auth?res
ponse_type=code&redirect_uri=http%3A%2F%2Ftug
asakhirsaya2015.esy.es%2F&client_id=392853343
800-
ad93vos2rbd7erno4skg2j0u89b8olpv.apps.googleu
sercontent.com&scope=https%3A%2F%2Fwww.google
apis.com%2Fauth%2Fplus.login+https%3A%2F%2Fww
w.googleapis.com%2Fauth%2Fuserinfo.email+http
s%3A%2F%2Fwww.googleapis.com%2Fauth%2Fplus.me
&access type=offline&approval prompt=auto
```

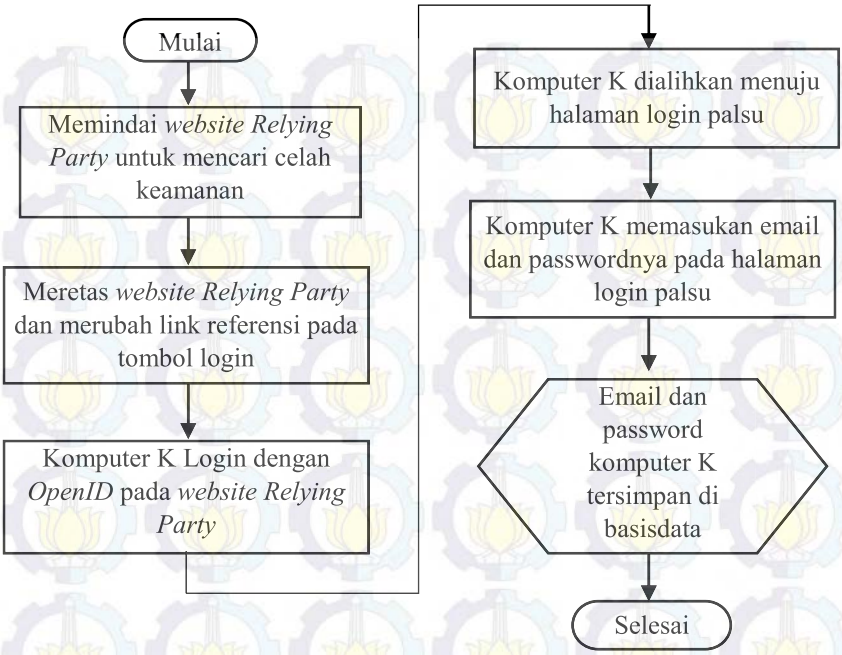
Kode 3.3 Tautan referensi pada tombol "Sign with Google"

Dengan tautan referensi yang sangat panjang tersebut, pengguna lebih cenderung tidak mempedulikannya. Teknik peretasan ini dikenal dengan nama *phishing*.

Pada tahap ini, pengujian dilakukan bertujuan untuk mengambil alih *website Relying Party* dan mengubah tautan referensi pada tombol *login OpenID* serta mencuri *email* dan *password* pengguna yang tersimpan di*provider*. Properti yang digunakan dalam pengujian ini adalah:

a. Komputer penguji P
b. Komputer korban K
c. *Webserver* untuk *Relying Party*

Untuk skema pengujian dapat dilihat pada gambar 3.12



Gambar 3.12 Skema kerja peretasan *website Relying Party*

3.3.4 Pengujian keamanan *AppID/Client ID* dan *App Secret/Client Secret*

App ID/Client ID dan *App Secret/Client Secret* adalah kode yang berfungsi sebagai identitas dari aplikasi yang dibuat *Relying Party* di *provider*. *Provider* mengenali *Relying Party* dari *App ID/Client ID* dan *App Secret/Client Secret*. Pengujian tahap 3 bertujuan untuk mengetahui respon dari *provider* jika *App ID/Client ID* dan *App Secret/Client Secret* digunakan pada *website Relying Party* yang lain.

Adapun yang diperlukan dalam pengujian ini adalah

1. Satu aplikasi dari Facebook dan Google

   a.Facebook

   App ID : 777843195644667

   App Secret : cac60cc4a1ad308f781b8718717e9eb9

b. Google

Client ID :392853343800-ad93vos2rbd7erno4skg 2j0u89b8olpv.apps.Googleusercontent.com

Client Secret : RrK7QLi6doFgb7VPOJU2twU2

2. Dua *website Relying Party*

   Website Relying Party A pada *webhosting* dan *website Relying Party* B di *localhost*.

3. 1 Komputer P

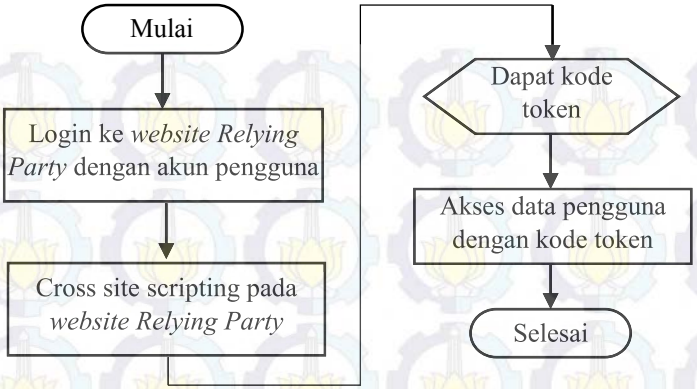
   Komputer P bertindak sebagai penguji.

3.3.5 Pencurian Kode Token

Kode token adalah sebuah kode otorisasi yang diberikan *provider* kepada *Relying Party* supaya *Relying Party* bisa mengakses data-data pengguna di *provider*. Ketika seorang pengguna sudah *login* dengan *OpenID*, maka *provider* akan mengirimkan kode token kepada *Relying Party*, dan kemudian *Relying Party* bisa mengakses data-data pengguna. Kode token ini dikirim bersamaan dengan pengiriman parameter-parameter otentikasi.

Sebuah celah keamanan muncul akibat dari penyalahgunaan kode token ini. Jika akun pengguna di *website Relying Party* berhasil di ambil alih, misalnya melalui *session hijacking*, dan kemudian peretas akan membuka *script website Relying Party* ketika pengguna sudah *login* dan kemudian kode token akan ditemukan. Dan peretas bisa menggunakannya untuk mengakses data-data pengguna di *provider* seperti halnya hak akses yang didapatkan oleh *Relying Party*.

Untuk skema pengujian untuk mendapatkan kode token ini dapat dilihat pada gambar 3.13

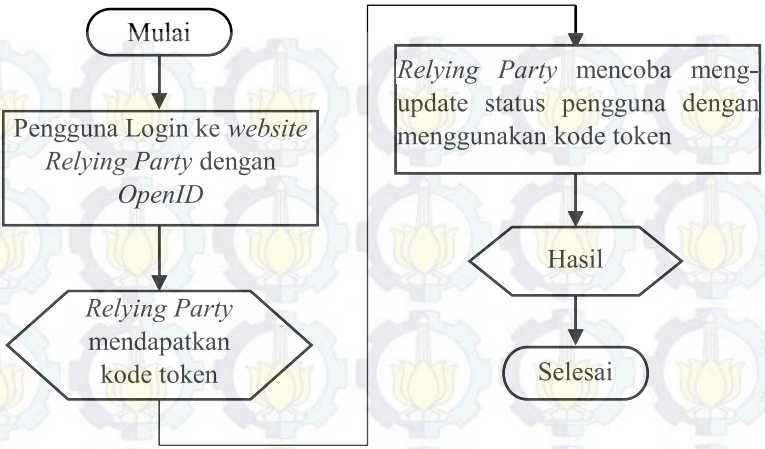


Gambar 3.13 Skema pencurian kode token

3.3.6 Menembus batasan otorisasi *provider*

Setiap *Relying Party* mendapatkan hak untuk mengakses data-data pengguna yang ada di *provider*. Namun, tidak semua data dapat diakses oleh *Relying Party*, hak akses/otorisasi yang didapat oleh *Relying Party* sesuai dengan permintaan *Relying Party* dan telah disetujui oleh pihak *Provider*. Jadi dapat diartikan Relying mempunyai batasan dalam mengakses data pengguna di *provider*.

Pada pengujian ini bertujuan untuk mencoba menembus batasan yang diberikan *provider*. Pengujian ini menggunakan 1 aplikasi *Relying Party* yang tidak mempunyai hak untuk meng-*update status* pengguna di Facebook dan kemudian akan dicoba untuk melakukan *update status* menggunakan kode token.Adapun skema pengujian ini dapat dilihat pada gambar 3.14



Gambar 3.14 Skema menembus batasan otorisasi

(Halaman ini sengaja dikosongkan)

BAB 4

ANALISA DAN PENGUJIAN SISTEM

Dalam bab ini dilakukan pembahasan mengenai analisa dari prinsip kerja protokol *OpenID*. Analisa ini meliputi analisa protokol transfer data (*HTTP*), analisa proses pengalihan pengguna dari *website Relying Party* menuju *website provider*, dan analisa kelemahan *website Relying Party* yang bisa membahayakan data-data pengguna di *provider*.

Selain itu, pada bab ini juga dilakukan pembahasan tentang pengujian dari prinsip kerja protocol *OpenID*. Pengujian dilakukan berdasarkan analisa yang telah dilakukan, dengan menggunakan objek *website Relying Party* pada localhost, menggunakan jasa *webhosting* dan beberapa *website* yang juga menggunakan protokol *OpenID*.

4.1 Analisa Prinsip Kerja *OpenID*

Protokol *OpenID* merupakan protokol yang menggunakan jasa pihak ketiga untuk melakukan otentikasi pengguna. Pada proses otentikasi, pengguna akan dialihkan/redirect dari *website Relying Party* menuju *website OpenID provider*. Proses pengalihan/*redirect* ini terjadi antara 2 jaringan yang berbeda, dan pada tahap ini pengguna sangat rentan terhadap serangan seorang peretas, yaitu melalui *metode man in the middle attack (MITMA)*. Berdasarkan prinsip kerja *OpenID*, diketahui bahwa *OpenID* bekerja dengan menggunakan protokol *HTTP GET request*. Penggunaan metode *GET* ini sangat rentan dalam hal keamanan, karena semua paket yang dikirimkan terlihat jelas pada *URL* yang dikirimkan oleh pengguna. Paket yang dikirimkan tersebut merupakan parameter-parameter yang digunakan dalam proses otentikasi seperti parameter identitas dari pengguna *OpenID* (*OpenID.identity*), asosiasi (*OpenID.assoc*) dan *URL* yang akan mengalihkan pengguna kembali ke *website Relying Party* (*OpenID.return to*). Semua parameter ini dapat dilihat oleh pihak lain seperti peretas melalui proses penyadapan (*sniffing*). Dan jika parameter ini didapat oleh seorang peretas, maka peretas bisa menggunakan layanan pada *website Relying Party* seperti menggunakan akun pengguna lain tanpa proses otentikasi.

Selain itu, penggunaan protokol *HTTP* tanpa melalui proses enkripsi seperti menggunakan *secure socket layer (SSL)* memungkinkan seorang peretas bisa mendapatkan *session cookie*s pengguna pada *website Relying Party* dengan cara menyadap di jaringan pengguna atau disebut dengan proses *sniffing*. Dengan memanfaatkan *cookies* ini, peretas juga bisa mengambilalih akun pengguna pada *website Relying Party* tanpa melalui proses otentikasi.

Selanjutnya, potensi celah keamanan yang mungkin terjadi yaitu ketika proses pengalihan pengguna dari *website Relying Party* menuju *website OpenID provider*. Proses yang melibatkan 2 jaringan yang berbeda ini memungkin peretas bisa merubah rute pengalihan tersebut dan pada proses ini pengguna rentan terhadap serangan *phishing*. *Phishing* adalah usaha untuk mendapatkan suatu informasi penting dan rahasia secara tidak sah, seperti pengguna *id, password, pin*, informasi rekening bank, informasi kartu kredit, atau informasi rahasia yang lain. Salah satu teknik yang digunakan adalah dengan mengalihkan pengguna menuju *website* palsu. Proses pengalihan tersebut dapat dilakukan dengan 2 cara, yaitu:

a. *DNS Spoofing*

   Proses yang melibatkan 2 jaringan yang berbeda ini memungkinkan peretas bisa merubah rute pengalihan tersebut. Dengan menggunakan teknik *dns spoofing*, pengguna bisa dialihkan menuju sebuah *website* palsu yang sudah disiapkan oleh peretas, dan jika pengguna terjebak dalam perangkap ini, tentu saja ini sangat berbahaya, karena pengguna *id* dan *password* pengguna bisa di ketahui oleh peretas.

b. Meretas *Website Relying Party*

   Selain melalui dns spoofing, keamanan data pengguna di *provider* juga bisa terancam jika *website Relying Party* menjadi korban hacking, seperti defacing. Ketika *website Relying Party* diambil alih oleh peretas, kemudian peretas merubah link referensi pada tombol "*Login* with Google" menjadi link referensi *website* palsu. Dan ketika pengguna memilih *login* dengan Google, maka pengguna tidak akan

dialihkan menuju *website* Google, tapi menuju *website* palsu dan kemudian pengguna akan diminta memasukan *email* dan *password* nya di *website* palsu tersebut.

Tindakan *phishing* ini tentu sangat berbahaya, karena kelemahan pada *Relying Party*, data-data penting pengguna di *provider* menjadi terancam. Salah satu *provider*, yaitu wordpress.com berusaha mengantisipasi tindakan *phishing* dengan cara tidak menampilkan halaman *login* ketika pengguna menggunakan *OpenID*. Jadi, jika ingin menggunakan *OpenID* wordpress.com, pengguna terlebih dahulu harus *login* di *website* wordpress.com baru bisa *login* dengan URL *OpenID*nya di *website Relying Party*.

Pada protokol *OpenID*, *Relying Party* mempunyai hak untuk mengakses data-data pengguna, tapi tidak semua data dapat pengguna yang ada di *provider* dapat diakses oleh *Relying Party*. Setiap *provider* mempunyai regulasi sendiri dalam mengatur otorisasi kepada *Relying Party*. Seperti pada *provider* Facebook, ketika *Relying Party* mendaftarkan aplikasi *OpenID* kepada Facebook, secara default *Relying Party* akan mendapatkan hak untuk mengakses profil pengguna (user_profiles), *email* dan teman pengguna (*user_friends*). Namun, *Relying Party* juga diberi kesempatan untuk mendapatkan hak otorisasi tambahan dengan memilih hak otorisasi yang diberikan oleh Facebook, dengan syarat *Relying Party* harus membuat sebuah permohonan dan menjelaskan alasan kenapa *Relying Party* memerlukan hak akses tersebut, serta juga harus menjelaskan secara detail bagaimana aplikasi *Relying Party* itu berjalan. Permohonan untuk mendapatkan hak otorisasi akan di pertimbangkan oleh Facebook. jika Facebook menyetujui permohonan tersebut maka *Relying Party* akan mendapatkan hak otorisasi tambahan tersebut. Untuk daftar hak akses yang diberikan oleh Facebook dapat dilihat pada gambar 4.1

Gambar 4.1 Daftar hak akses yang disediakan Facebook

Ketika pengguna *login* dengan menggunakan *OpenID*, maka pengguna akan diminta persetujuan bahwa pengguna akan memberi izin kepada *Relying Party* untuk mengakses data-data pengguna. Pada saat memberi persetujuan ini dibutuhkan ketelitian pengguna, karena data yang akan diakses *Relying Party* tidak hanya identitas pribadi, tapi masih ada fitur lain yang bisa diakses oleh *Relying Party*. Facebook menerapkan regulasi yang cukup ketat tentang otorisasi ini, tapi berbeda dengan Google yang memberi keleluasaan *Relying Party* untuk menambah hak otorisasi mereka tanpa perlu meminta permohonan kepada Google.

Adapun cara *Relying Party* untuk bisa mendapatkan otorisasi terhadap data-data pengguna yaitu dengan menggunakan kode token. Kode token akan didapatkan *Relying Party* setiap pengguna *login* di aplikasi *Relying Party* dengan menggunakan *OpenID*, dan kode token ini juga mempunyai masa berlaku yang terbatas, sehingga tidak setiap waktu *Relying Party* dapat menggunakannya. Namun walaupun begitu, terdapat potensi bahaya ketika kode token ini berhasil dicuri oleh orang lain. Jika kode token ini diketahui orang lain, maka orang tersebut juga akan mendapatkan hak otorisasi yang sama dengan hak otorisasi yang dimiliki oleh *Relying Party*. Misalnya, jika *Relying Party* A

menggunakan *provider* Facebook dan mendapatkan akses yang cukup besar, seperti akses *user_posts* dan *read_mailbox*, maka orang lain yang mendapatkan kode token milik *Relying Party* A juga bisa melakukan hal yang sama dengan yang dilakukan *Relying Party* A.

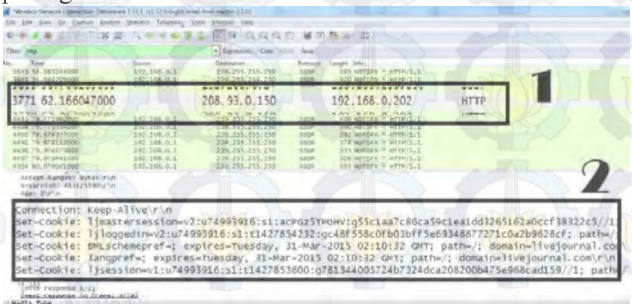
4.2 Pengujian Keamanan Sistem pada *OpenID*

Pengujian keamanan pada protocol *OpenID* berpatokan dari hasil analisa sebelumnya. Pengujian keamanan protokol *OpenID* dilakukan dengan beberapa tahap, yaitu:

1. Pengujian Protokol Transfer Paket Data (*HTTP*)
2. Pencurian kode token
3. Pemalsuan Halaman *Login Provider*
4. Pengujian keamanan *App ID/Client ID* dan *App Secret/Client Secret Relying Party*.
5. Menembus batasan hak akses *Relying Party* yang diberikan *Provider*

4.2.1 Pengujian Protokol Pengiriman Paket Data (*HTTP*)

Untuk mengetahui bagaimana proses pertukaran data dijaringan pada protokol *OpenID* dapat dilakukan dengan cara *sniffing*. Teknik *sniffing* ini dapat dilakukan dengan bantuan aplikasi wireshark. Proses *sniffing* ini dilakukan ketika pengguna A dengan *ip address* 192.168.0.202 melakukan *login* dengan *OpenID* pada *website* livejournal.com dengan *ip address* 208.93.0.150. Hasil *sniffing* dapat dilihat pada gambar 4.2



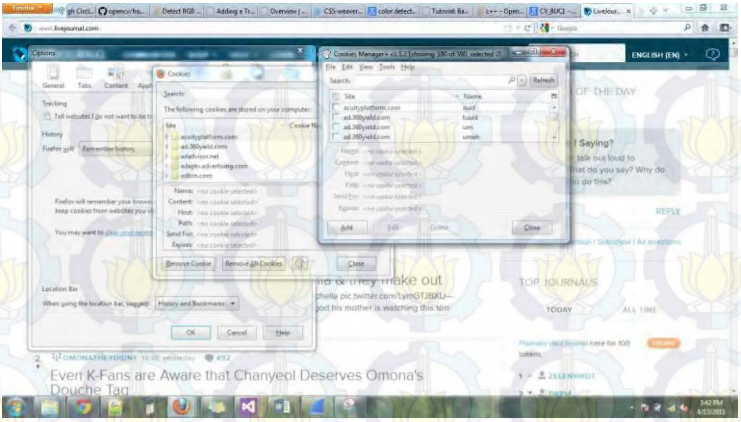
Gambar 4.2 Hasil *Sniffing* pada aplikasi Wireshark

Dari hasil *sniffing* dapat terlihat semua paket data yang terkirim dijaringan antara pengguna A dengan *website* livejournal.com. dan yang paling penting dari hasil *sniffing* ini adalah pada gambar 4.2 nomor 2 terlihat *cookies* yang dikirimkan oleh livejournal.com kepada pengguna A. Ini merupakan celah keamanan yang berbahaya jika *cookies* ini diketahui oleh orang lain atau peretas. *Cookies* dapat digunakan oleh peretas untuk *login* pada akun pengguna A pada *website* livejournal.com tanpa harus melewati proses otentikasi. Teknik pembajakan akun dengan menggunakan *cookies* ini disebut dengan *session hijacking*. Untuk membuktikan celah keamanan pada *cookies* ini, kemudian pengujian dilanjutkan dengan melakukan *session hijacking* pada akun pengguna A.

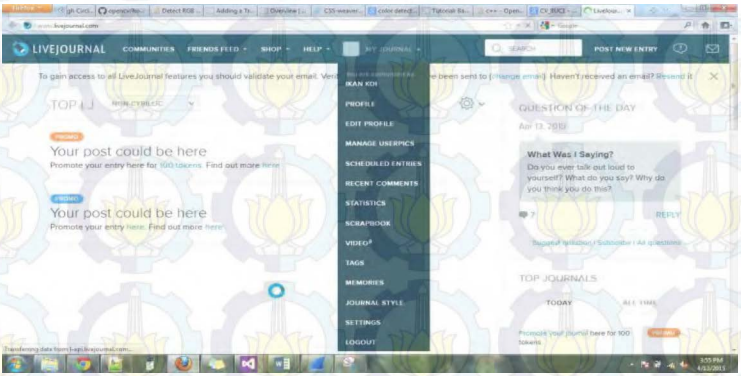
Dari *cookies* yang didapatkan dari *sniffing* tersebut, kemudian *cookies* tersebut disimpan kedalam browser dengan menggunakan *add-on Cookies Manager* pada *browser* mozilla firefox. Daftar *cookies* yang dimasukan kedalam *browser* adalah sebagai berikut:

a. `ljmastersession=v2:u74993916:s1:acPGz5TPOHV:g55c1aa7c86ca59c1ea1dd3265162a0ccf38322c5//1; path=/; domain=www.livejournal.com; HttpOnly`
b. `ljloggedin=v2:u74993916:s1:t1427854232:gc48f558c0fb03bff5e69348877271c0a2b9628cf; path=/; domain=livejournal.com; HttpOnly`
   `BMLschemepref=; expires=Tuesday, 31-Mar-2015 02:10:32 GMT; path=/; domain=livejournal.com`
c. `langpref=; expires=Tuesday, 31-Mar-2015 02:10:32 GMT; path=/; domain=livejournal.com`
d. `ljsession=v1:u74993916:s1:t1427853600:g781344005724b7324dca208200b475e968cad159//1; path=/; domain=livejournal.com; HttpOnly`

Proses dan tahapan pengujian dapat dilihat pada gambar 4.3 dan gambar 4.4



Gambar 4.3 Proses menambahkan *cookies* di browser Mozilla Firefox



Gambar 4.4 Berhasil *login* pada akun Ikan Koi dengan menggunakan *cookies*

Pada gambar 4.4 dapat dilihat bahwa penguji bisa masuk kedalam akun pengguna A tanpa harus melewati proses otentikasi. Hal ini menunjukan salah satu celah keamanan pada protocol *OpenID.*

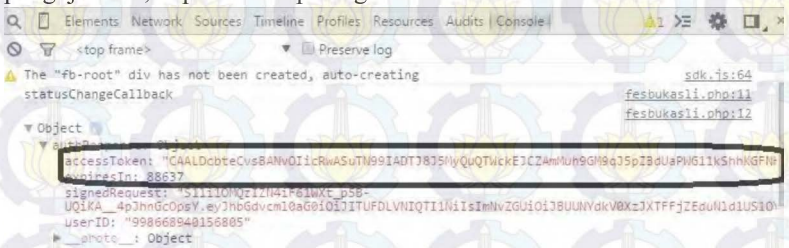
4.2.2 Pencurian Kode Token

Kode token adalah sebuah deretan kode unik yang diberikan *provider* kepada *Relying Party* untuk mengakses data-data pengguna yang ada di *provider*. Kode token ini akan diberikan kepada *Relying*

Party ketika pengguna *login* di *website Relying Party* menggunakan *OpenID*.

Pada pengujian ini, akan dicoba mendapatkan kode token yang diberikan *provider* kepada *Relying Party*. Untuk mendapatkan kode token, pengguna harus *login* terlebih dahulu pada *website Relying Party* dengan *OpenID*. Untuk *provider* yang digunakan adalah Facebook, dengan *SDK (Software Development Kit)* yang digunakan adalah *Javascript SDK* yang disediakan Facebook.

Untuk melihat kode token yang diberikan Facebook, dapat dilakukan dengan cara melihat *element website Relying Party* pada *browser*. Untuk lebih rinci mengenai kode token yang didapat dari pengujian ini, dapat dilihat pada gambar 4.5



Gambar 4.5 Kode Token yang berhasil ditemukan

Dari pengujian ini diketahui bahwa sangat mudah untuk mendapatkan kode token. Potensi bahayanya adalah jika akun pengguna *Relying Party* berhasil diretas melalui *session hijacking* dan peretas menemukan kode token ini, maka peretas juga akan bisa mengakses data-data pengguna di *provider*. Peretas akan memiliki hak otorisasi/akses yang sama dengan hak yang dimiliki oleh *Relying Party*. Celah ini semakin berbahaya jika *Relying Party* mendapatkan hak akses yang cukup besar terhadap data-data pengguna.

4.2.3 Pemalsuan Halaman *Login Provider* (*Phishing*)

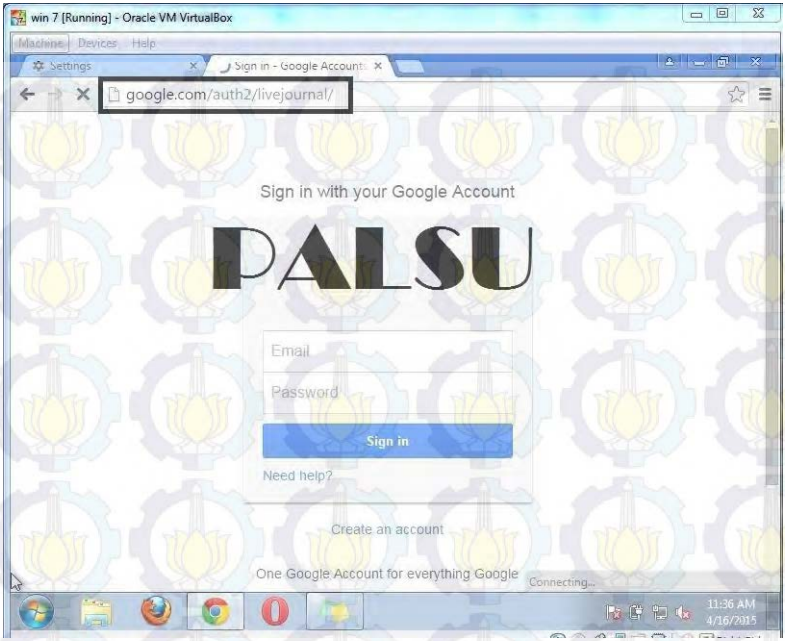
Pada saat pengguna memilih *login* dengan menggunakan protokol *OpenID*, maka setelah itu pengguna akan dialihkan ke *website provider*. Pengguna akan diminta *login* dan memasukan *email* dan *password*-nya jika pengguna belum *login* pada *website provider*. Pengujian pada tahap ini adalah mengubah arah pengalihan (*redirect*)

menuju halaman *login* palsu yang mirip dengan halaman *login provider* dan sebagai sampel pada pengujian ini menggunakan 2 buah *provider* yaitu Facebook dan Google.

Tahap awal yang dilakukan adalah dengan membuat sebuah halaman *login* palsu yang mirip dengan halaman *login website provider*. Cara untuk membuat halaman palsu tersebut adalah dengan mengambil *script* asli dari *website provider* dan kemudian menyimpannya pada *webhosting/webserver online*.

Untuk tampilan dari halaman palsu tersebut dapat dilihat pada gambar 4.6 dan 4.7

Gambar 4.6 Halaman *Login* Facebook Palsu



Gambar 4.7 Halaman *Login* Google Palsu

Halaman palsu tersebut berguna untuk menerima data yang dimasukan oleh pengguna, dan ketika pengguna menekan tombol *Sign in*/Masuk, maka *email* dan *password* pengguna akan tersimpan kedalam basisdata.

Tahap selanjutnya adalah membuat basisdata yang berfungsi untuk menyimpan *email* dan *password* pengguna/korban/target. Adapun rincian dari basisdata yang digunakan dapat dilihat pada Tabel 4.1

Tabel 4.1 Tabel penyimpanan *email* dan *password* pengguna dibasisdata

No	Nama	Keterangan
1	*Email*	*Email* pengguna
2	*Password*	*Password* pengguna di *provider*

Setelah membuat halaman palsu dan basisdatanya, tahap selanjutnya adalah merubah alur pengalihan menuju *website provider*. Ada 2 cara untuk mengalihkan pengguna menuju halaman *login* palsu, yaitu:

a. DNS Sppofing

Proses peralihan dari *website Re;ying Party* ke *website provider* rentan terhadap serangan *DNS Spoofing*. *DNS Spoofing* adalah teknik yang digunakan untuk mengambil alih DNS server sehingga *DNS* atau layanan yang menyimpan nama domain dan *IP address* sebuah situs akan dialihkan ke server yang lain. Jadi ketika pengguna A memilih *login* dengan Google, pada aturan yang sebenarnya, harusnya pengguna A dialihkan ke *website* Google untuk verifikasi data, namun karena DNS spoofing ini, pengguna A tidak dialihkan ke *website* Google, namun ke *website* yang lain.

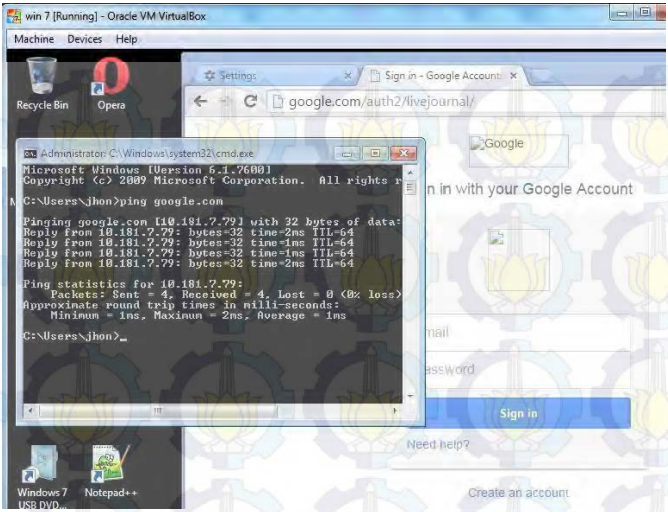
Teknik *DNS Spoofing* ini dapat dilakukan dengan menggunakan beberapa *tools*, namun pada pengujian ini menggunakan salah satu tools di sistem operasi Kali Linux, yaitu *Ettercap*. Dan untuk properti yang digunakan dalam pengujian ini yaitu

1. Komputer P dengan sistem operasi Kali Linux
   Komputer P bertindak sebagai seorang peretas yang akan mengalihkan korban menuju *website* palsu.
2. Komputer S dengan sistem operasi debian 7.0
   Komputer S bertindak sebagai webserver dari *website* palsu dan akan menyimpan *email* dan *password* korban dengan *ip address* 10.181.7.79
3. Komputer K dengan system operasi windows 7
   Komputer K adalah korban dari *DNS Spoofing* dengan *ip address* 10.181.7.79

Berikut adalah hasil pengujian dengan teknik *DNS spoofing*:

Gambar 4.8 Tampilan Hasil *DNS Spoofing* dari komputer P

Pada Gambar 4.8 terlihat permintaan data menuju google.com semuanya dialihkan menuju Komputer S, termasuk trafik dari computer K juga dialihkan. Pada gambar 4.9 akan terlihat dampak dari *DNS Spoofing* pada Komputer K.



Gambar 4.9 Dampak *DNS Spoofing* pada Komputer K

Ketika Komputer K menjadi korban *DNS Spoofing*, maka semua permintaan data dari Komputer K menuju google.com akan dialihkan menuju Komputer S. Dengan teknik *DNS Spoofing* ini, Komputer K akan percaya bahwa halaman *login* yang tampil adalah halaman asli Google, karena terlihat pada *address bar* bahwa *URL* yang tertulis adalah google.com. padahal sesungguhnya, jika Komputer K dialihkan ke halaman *login* Google, *URL* yang sebenarnya adalah: "https://accounts.Google.com/Service*Login*?service=lso&passive=1209600&continue=https://accounts.Google.com/o/oauth2/auth?zt%3DChQ0WExOcGQwbmc4ZnVrZmFqbjVpOBIfY3VrdGwtZHd2eThmY3A3dGRpbGpLS2J4TjFZU21nSQ%25E2%2588%2599APsBz4gAAAAAVUHZTZ8hNzQJg6HrWO1erSl_A-EB1ZB5%26from_*login*%3D1%26hl%3Den%26e%3D3100077%26as%3D60237f647d58fcf6&ltmpl=popup&shdf=CrECCxIRdGhpcmRQYXJ0eUxvZ29VcmwaugEvL2ltYWdlcy1sc28tb3BlbnNvY2lhbC5nb29nbGV1c2VyY29udGVudC5jb20vZ2FkZ2V0cy9wcm94eT91cmw9aHR0cDovL3doLmxqLnJ1L2dvb2dsZS9sal9sb2dvMTIweDYwdy5wbmcmY29udGFpbmVyPWxzbyZnYWR

nZXQ9YSZyZXdyaXRlTWltZT1pbWFnZS8qJnJlc2l6ZV9oPTEyMCZyZXNpemVfdz0xMjAmbm9fZXhwYW5kPTEMCxIVdGhpcmRQYXJ0eURpc3BsYXlOYW1lGgtMaXZlSm91cm5hbAwLEgZkb21haW4aC0xpdmVKb3VybmFsDAsSFXRoaXJkUGFydHlEaXNwbGF5VHlwZRoHREVGQVVMVAwSA2xzbyIUE-trWmUfoY1IgPQayTHG25vyY28oATIUELVUHhIVTyyqT_77029nd4gr0Gc&hl=en&sarp=1&scc=1”

Dengan *URL* dengan tulisan acak dan panjang tersebut, besar kemungkinan semua pengguna tidak akan tahu jika mereka dialihkan.Sehingga Komputer K terjebak dan memasukan *email* dan *passwordnya* pada kolom di *website* palsu tersebut. Hal ini tentu sangat berbahaya dan merugikan Komputer K.

b. *Deface website Relying Party*

Selain dengan menggunakan teknik *dns spoofing*, cara untuk mengalihkan pengguna menuju halaman *login* palsu adalah dengan memanfaatkan kelemahan pada *website Relying Party*. Kelemahan yang dimaksud adalah ketidakmampuan *website Relying Party* dalam menghadapi serangan dari peretas, seperti *website Relying Party* diambil alih oleh peretas.

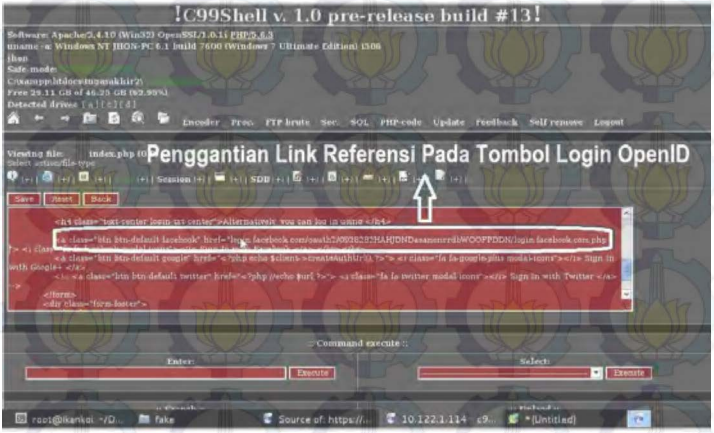
Untuk itu, pada tahap pengujian ini, dilakukan peretasan pada *website Relying Party* dengan tujuan untuk merubah tampilan *website Relying Party*, yaitu merubah atribut tautan referensi pada tombol *login* dengan protokol *OpenID* (Google dan Facebook).

Pada pengujian ini dilakukan simulasi peretasan terhadap *website Relying Party*. Adapun perlengkapan yang digunakan dalam simulasi ini adalah:

1. *Website Relying Party*
   Website Relying Party ini menggunakan *provider* Facebook dan Google dengan alamat http://10.122.1.114/tugasakhir2
2. Halaman *Login* Palsu
   Tampilan dari halaman palsu ini sama dengan halaman *login* Facebook dan Google. Halaman ini masih terdapat dalam server yang sama dengan *website Relying Party*.

3. Komputer P
   Komputer P adalah computer penguji yang akan meretas *website Relying Party*

Skenario dalam pengujian ini adalah, pertama Komputer P melakukan penetrasi pada *website Relying Party*.Komputer P menemukan celah *Local File Inclusion(LFI)* pada *website Relying Party* untuk dimasukan sebuah *backdoor* kedalam server. Dengan menggunakan *backdoor* tersebut, Komputer kemudian membuat 2 direktori dan menambahkan 2 halaman palsu. Untuk halaman palsu "login.Facebook.php" dimasukan kedalam direktori "oauth*provider*=facebook.com" dan halaman palsu "*login*.Google.php" dimasukan kedalam direktori "oauth*provider*=google.com". Proses peretasan dapat dilihat pada gambar 4.10.



Gambar 4.10 Peretasan *website Relying Party*

Setelah memasukan halaman palsu kedalam server, kemudian Komputer P mengubah script halaman index.php pada *website Relying Party*, dimana halaman index.php ini adalah halaman *login website Relying Party*. Untuk proses perubahan atribut URL pada tombol *login OpenID* di *website Relying Party* dapat dilihat pada gambar 4.11 dan 4.12

Gambar 4.11 Tautan referensi yang asli

Gambar 4.12 Tautan referensi diubah menuju halaman palsu

Untuk hasil setelah peretasan dapat dilihat pada gambar 4.13.

Gambar 4.13 Hasil teknik phishing

Proses pengalihan pada protokol *OpenID* ini memunculkan sebuah celah keaamanan yang mempermudah seorang peretas untuk mencuri akun pengguna. Selain itu, lemahnya sistem keamanan yang digunakan oleh *Relying Party* juga menjadi ancaman terhadap data-data pengguna di*provider*.

4.2.4 Pengujian Keamanan *App ID/Client ID* Dan *App Secret/Client Secret Relying Party*

App ID/Client ID dan *App Secret/Client Secret* adalah kode yang berfungsi sebagai identitas dari aplikasi yang dibuat *Relying Party* di *provider*. *Provider* mengenali *Relying Party* dari *App ID/Client ID* dan *App Secret/Client Secret*. Pengujian tahap 3 bertujuan untuk mengetahui

respon dari *provider* jika *App ID/Client ID* dan *App Secret/Client Secret* digunakan pada *website Relying Party* yang lain.

Adapun yang diperlukan dalam pengujian ini adalah

1.Satu Aplikasi dari Facebook dan Google

a. Facebook
   App id : 777843195644667
   App Secret : cac60cc4a1ad308f781b8718717e9eb9
b. Google
   Client ID :392853343800-ad93vos2rbd7erno4skg2j0u89b8olpv.apps.Googleusercontent.com
   Client Secret : RrK7QLi6doFgb7VPOJU2twU2

2.Dua *website Relying Party*

Kedua *website Relying Party* dipasang pada webhosting.

3.Satu Komputer P

Komputer P bertindak sebagai penguji.

Website Relying Party A pada *webhosting* dengan alamat http://tugasakhirsaya2015.esy.es dan *website Relying Party* B dengan alamat http://tugasakhir.esy.es. *Website Relying Party* A adalah *website* yang terdaftar dalam aplikasi Facebook dan Google serta yang mempunyai hak untuk menggunakan *App ID/Client ID* dan *App Secret/Client Secret*.

Kemudian, *App ID/Client ID* dan *App Secret/Client Secret* juga digunakan pada *website Relying Party* B yang sebenarnya tidak terdaftar dalam aplikasi Facebook dan Google. Ketika di coba *login* dengan menggunakan protokol *OpenID* pada *website Relying Party* B, ternyata tidak bisa *login* dan *provider* akan menampilkan pesan kesalahan seperti pada gambar 4.14 dan 4.15.

Gambar 4.14 Pesan Kesalahan dari Facebook

Gambar 4.15 Pesan Kesalahan dari Google

Berdasarkan pengujian keamanan *App ID/Client ID* dan *App Secret/Client Secret* dapat diketahui bahwa *provider* sudah mempunyai sistem keamanan untuk mengatasi tindakan penyalahgunaan/pencurian *App ID/Client ID* dan *App Secret/Client Secret* dari *Relying Party*. Sehingga jika ada peretas yang menggunakan *App ID/Client ID* dan *App Secret/Client Secret* pihak Y, maka protocol *OpenID* pada peretas tidak akan berjalan dan akan menampilkan pesan kesalahan.

4.2.5 Menembus Batasan Hak Akses *Relying Party* Yang Diberikan *Provider*

Setiap *Relying Party* mendapatkan hak untuk mengakses data dan beberapa kepentingan lainnya berdasarkan permintaan *Relying Party* ketika membuat aplikasi di *provider*. Seperti ketika membuat aplikasi Facebook, secara *default* Facebook memberi izin kepada pembuat aplikasi untuk mendapatkan identitas pengguna, alamat *email* dan daftar teman pengguna. Untuk mendapatkan hak akses tambahan seperti mengupdate status pengguna, pembuat aplikasi harus mengirim permintaan kepada Facebook serta menjelaskan tujuan dan cara kerja aplikasi yang dibuat dengan detail. Jika Facebook menyetujui permintaan tersebut, aplikasi akan mendapatkan hak akses untuk mengupdate status pengguna.

Pada pengujian ini, akan menguji apakah batasan hak akses tersebut bisa ditembus atau tidak. Untuk itu digunakan 1 buat aplikasi yang hanya mendapatkan hak akses default dari Facebook, kemudian akan dicoba mengupdate status pengguna dengan menggunakan alamat URL berikut:

```
https://graph.Facebook.com/ID-
FACEBOOK/feed?method=POST&access_token=ACCESS-
TOKEN-KORBAN&message=KALIMAT STATUS
```

1. ID-FACEBOOK adalah kode id pengguna di Facebook
2. ACCESS TOKEN adalah kode token yang diberikan *provider* kepada aplikasi untuk menggnakan hak akses yang diberikan.
3. KALIMAT STATUS adalah isi dari status yang akan di tulis di halaman pengguna.

URL tersebut akan diubah menjadi seperti berikut:

```
https://graph.Facebook.com/100000411255552/feed?
method=POST
&access_token=CAALDcbteCvsBAGVFNgf4dY6INqLdGsoOk
FVyx6GVel9dxRuQyZBrU5mkzbdoSiNr1la0HIcGaZBialU4y
AyMTIohmGZCRFOp7suqxNF0UciUHXAuMswbmJ67kPBJBuMgt
jCgDu4AoVxuajfKHDIyOeKpEl5o4agZBAGge2ImbXBt5NYaU
VIkJuVPaGabiq4dxVzwVr5SuzGHCWKgjh3N&message=UJI
COBA UPDATE STATUS
```

Selanjutnya URL tersebut akan dijalankan di browser dan menghasilkan tampilan kesalahan seperti gambar 4.16 berikut:

Gambar 4.16 Pesan Kesalahan otorisasi

Ternyata Facebook tidak memberikan izin kepada aplikasi untuk mengupdate status pengguna, karena belum mendapat hak dari Facebook.

4.3 Potensi Celah Keamanan dan Potensi Bahaya yang dapat terjadi

Berdasarkan 5 tahapan pengujian keamanan yang telah dilakukan, diketahui bahwa terdapat beberapa celah keamanan yang pada protokol *OpenID* diantaranya adalah:

1. *Website Relying Party* yang menggunakan *HTTP* tanpa dilengkapi dengan layer yang aman (*Secure Socket Layer/SSL*) membuat semua paket data yang dikirimkan dan diterima pengguna dapat diketahui oleh pihak lain dan akun pengguna di *Relying Party* sangat rentan untuk dicuri. Tidak hanya akun pengguna di *Relying Party* yang menjadi rentan terhadap tindakan *hacking*, tetapi data-data pengguna di *Provider* juga ikut terancam. Karena ketika akun pengguna di *Relying Party* berhasil dicuri, peretas juga berpotensi dapat menemukan kode token yang tersimpan pada *Website Relying Party*. Dan jika kode token ini berhasil ditemukan, maka peretas akan mendapatkan hak akses yang sama dengan hak yang didapat oleh *Relying Party*, akan menjadi lebih bahaya jika *Relying Party* mendapatkan hak akses yang cukup besar dari *Provider*,

seperti bisa mengakses kotak pesan di *provider* Facebook dan Google, ataupun mengakses data pengguna di *Google Drive*.

Berbeda halnya dengan metode *login* biasa, ketika akun pengguna disuatu *Website* dicuri peretas, maka yang akan terkena dampak hanya akun pengguna di *Website* tersebut, tidak berpengaruh terhadap data-data pengguna di *Website* lain.

Contoh kasus penyalahgunaan kode token ini adalah tindakan *spamming* di Facebook, dimana banyak akun pengguna mengirimkan suatu secara otomatis tanpa diketahui oleh pemilik akun.

2. Proses otentikasi yang berada diluar sistem/*website Relying Party* membuat pengguna harus melewati proses pengalihan dari *Website Relying Party* menuju *Website Provider*. Proses pengalihan ini sangat rentan terhadap tindakan *Phishing*. Karena dengan adanya proses pengalihan ini, membuka peluang kepada peretas untuk mengubah tujuan pengalihan menuju *Website* palsu. Dan dampak paling bahaya dari celah ini adalah pencurian *email* dan *password* pengguna di *Provider*, dan jika ini terjadi maka semua data-data pengguna menjadi terancam.

   Pada metode login biasa, proses otentikasi berada didalam sistem *Website* penyedia layanan, tidak menggunakan jasa pihak ketiga, dan pengguna tidak melewati proses pengalihan. Sehingga peluang pengguna menjadi korban *phishing* tidak sebesar pada protokol *OpenID*, kecuali jika dari awal pengguna sudah terjebak dalam *dns Spoofing*. Dan walaupun pengguna sudah terjebak dalam *dns Spoofing*, dan akun pengguna berhasil dicuri oleh peretas, yang terkena dampak hanya akun pengguna pada *Website* penyedia layanan, tidak ada pengaruh dengan akun pengguna pada *provider* (*provider email* yang digunakan pengguna).

   Celah keamanan ini juga berpotensi menyebabkan pengguna mengalami kerugian finansial, karena ada beberapa *Website* toko *online*(*e-commerce*) yang menggunakan protokol *OpenID* seperti *Website* lazada.co.id, zalora.co.id, tokopedia.com, alibaba.com, blibli.com, elevenia.co.id dan bhineka.com.

Ketika akun pengguna pada *website-website* tersebut dicuri, bisa terjadi penyalahgunaan seperti menggunakan saldo pengguna yang tersimpan pada *website* tersebut, ataupun pemalsuan akun penjual dan lain sebagainya. Walaupun beberapa *website* toko *online* mengirimkan pesan konfirmasi ke *email* pengguna ketika terjadi transaksi, tetapi karena *email* dan *password* pengguna di *provider* juga dicuri dengan teknik *phishing*, *email* konfirmasi juga bisa disetujui sendiri oleh peretas. Ini menunjukan bahwa dampak dari celah ini sangat besar.

4.4 Saran dan Metode untuk Mencegah Pengguna Terjebak dalam Celah Keamanan

Dari celah keamanan yang berhasil ditemukan pada protokol *OpenID*, diketahui bahwa yang paling merasakan kerugian akibat celah keamanan ini adalah pengguna. Untuk membantu pengguna terhindar dari jebakan pada celah keamanan tersebut, ada beberapa cara dan metode yang perlu ditambahkan pada sistem/*website* yang menggunakan protocol *OpenID*, yaitu:

1. Untuk mencegah paket data yang dikirim dan diterima oleh pengguna dilihat/diketahui oleh pihak lain, maka pihak *Relying Party* sebaiknya atau mungkin diharuskan menggunakan layer yang aman, yaitu menggunakan protokol *HTTPS*. Dengan menggunakan *HTTPS*, paket data yang dikirim pengguna dijaringan dalam kondisi terenkripsi, sehingga tidak mudah diketahui oleh pihak lain.
2. Untuk mencegah pengguna terjebak dalam tindakan *phishing,* sejauh ini belum ada solusi yang benar-benar bisa mengatasi masalah tersebut, ditambah lagi protokol *OpenID* harus melewati proses pengalihan dari *website Relying Party* menuju *website Provider*. Walaupun begitu ada cara yang bisa membantu pengguna supaya terhindar dari jebakan *phishing*,

a. Sebaiknya *provider* tidak menampilkan halaman *login* ketika pengguna dialihkan menuju *website provider* jika pengguna belum *login* di *website provider*. Pengguna yang belum *login*

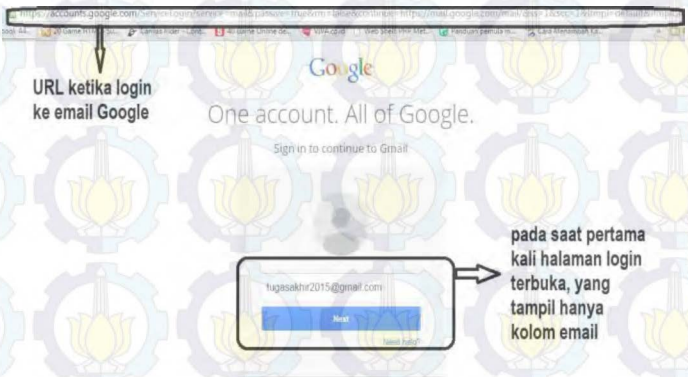
bisa *login* dengan membuka *website provider* pada tabulasi browser yang berbeda. Dengan begitu pengguna akan bisa mengetahui jika terjadi *DNS Spoofing* dijaringan, karena tampilan halaman *login* pada proses pengalihan berbeda dengan halaman *login* ketika pertama kali membuka *website provider*. Untuk mengetahui perbedaan antara kedua hal tersebut, dapat dilihat pada gambar 4.17 , gambar 4.18, gambar 4.19 dan gambar 4.20.

Gambar 4.17 Halaman utama Facebook

Gambar 4.18 Halaman *Login* ketika proses pengalihan

Ketika pengguna membuka *website* https://www.facebook.com/ maka halaman yang akan tampil seperti pada gambar 4.17, dan jika halaman yang tampil tidak seperti gambar 4.17, pengguna harus waspada karena bisa saja itu adalah sebuah jebakan *phishing*. Jika seorang peretas melakukan *phishing* dengan teknik *DNS Spoofing* pada protokol *OpenID*, peretas membuat sebuah halaman *login* yang mirip dengan dengan halaman *login* Facebook seperti pada gambar 4.18, peretas tidak bisa membuat 2 halaman palsu untuk 1 *provider* yang sama, karena prinsip kerja *DNS Spoofing* adalah berdasarkan request *ip address* menuju *webserver* halaman palsu, tidak menggunakan request *DNS*.

Dengan demikian, ketika peretas melakukan *DNS Spoofing*, dan pengguna membuka halaman https://www.facebook.com/ dan yang tampil seperti pada gambar 4.18, pengguna harus hati-hati dan sebaiknya tidak memasukan *email* dan *password*nya pada halaman tersebut.



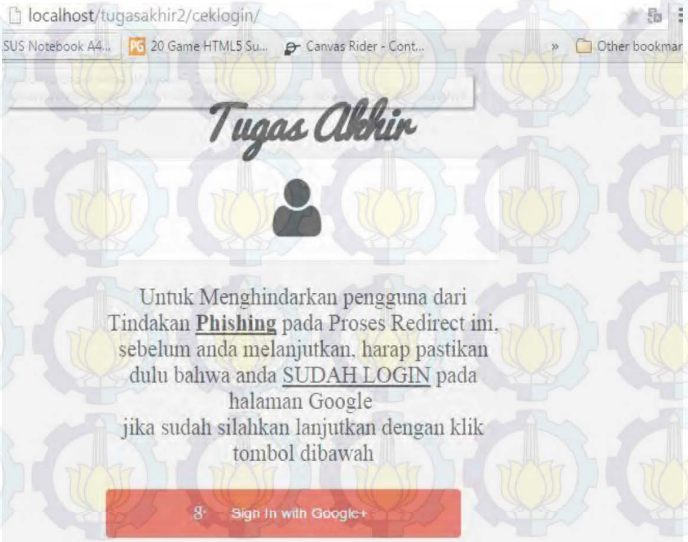
Gambar 4.19 Halaman *Login* Gmail tanpa pengalihan

Gambar 4.20 Halaman *Login* Gmail ketika proses pengalihan

Untuk halaman *login* Gmail asli tanpa pengalihan dengan halaman *login* Gmail pada proses pengalihan hampir sama, hanya terdapat sedikit perbedaan, yaitu perbedaan *URL* dan perbedaan kolom *login*. Pada halaman *login* Gmail asli tanpa pengalihan, hanya terdapat kolom *email* saja, sedangkan halaman *login* Gmail pada proses pengalihan terdapat kolom *email* dan kolom *password*. Sama halnya seperti *login* pada Facebook, pengguna harus lebih teliti memperhatikan tampilan dari halaman *login* Google, terlebih karena halaman *login* tanpa pengalihan dan halaman *login* pada proses pengalihan memiliki tampilan yang hampir sama.

b. *Relying Party* memberikan sebuah peringatan ketika pengguna memilih *login* dengan *OpenID*. Peringatan tersebut bertujuan untuk memastikan bahwa pengguna telah *login* pada *website provider*, jika pengguna belum *login* pada *website provider*, maka pengguna harus *login* terlebih dahulu di *website provider* sebelum *login* dengan *OpenID* pada

website Relying Party. Adapun contoh peringatan tersebut bisa dilihat pada gambar 4.21



Gambar 4.21 Peringatan sebelum *login* dengan *OpenID*

c. Adapun cara untuk terhindar dari tindakan *DNS Spoofing* adalah dengan mengubah *ip address* pengguna. Karena teknik *DNS Spoofing* biasanya dilakukan dengan memasukan *ip address* korban kedalam *tools spoofer* (seperti Ettercap, Cain & Abel, dan lain-lain). *DNS Spoofing* tidak berjalan pada *ip address* computer yang tidak terdapat dalam target *spoofer*. Jadi jika pengguna merasa sudah terjebak dalam *DNS Spoofing*, maka cara terbaik adalah dengan mengganti *ip address* yang sedang digunakan.

3. Untuk menghindari tindakan pencurian kode token, pihak *Relying Party* harus bisa menyimpan kode token tersebut pada tempat yang benar-benar aman, dimana tempat penyimpanan tersebut tidak dapat diakses dari sisi pengguna melalui browser, seperti dengan menyimpan dalam kode

PHP, dimana *PHP* merupakan bahasa pemrograman *server* side.

Penggunaan protokol *HTTPS* juga sangat dibutuhkan supaya kode token tidak dicuri melalui paket data yang dikirimkan di jaringan.

4. Setiap *Provider* harus membuat sebuah regulasi dan sebuah standar keamanan untuk *website Relying Party* yang ingin menggunakan jasa *provider* tersebut. *Website* yang boleh menggunakan protokol *OpenID* hanya *website* yang mempunyai kredibilitas yang tinggi dan reputasi yang benar-benar baik serta dengan standar keamanan yang tinggi. Dan pihak *provider* harus bisa menyeleksi *website* yang ingin menggunakan jasa *provider* tersebut, karena selama ini setiap *website* memiliki kebebasan untuk menggunakan protokol *OpenID* dan mendaftar pada suatu *provider* tanpa regulasi apapun. Dengan dibuatnya sebuah regulasi dan standar keamanan, diharapkan dapat meminimalisir dan mengurangi potensi celah keamanan yang mungkin terjadi dan mengurangi ancaman keamanan data pengguna akibat lemahnya sistem keamanan yang digunakan oleh *Relying Party*.

(Halaman ini sengaja dikosongkan)

DAFTAR PUSTAKA

[1] Amarudin, Widyawan, dan W. Najib,. 2014 "Analisis Keamanan Jaringan *Single Sign On (SSO)* Dengan Lightweight Directory Access Protocol (LDAP) Menggunakan Metode MITMA". Yogyakarta: Jurusan Teknik Elektro dan Teknologi Informasi Universitas Gadjah Mada.

[2] Pavol Sovis, Florian Kohlar dan Jorg Schwenk. "*Security Analysis of OpenID*" Jerman: Ruhr-University Bochum

[3] San-Tsai, Sun, E. Pospisil dan I. Muslukhov 2012 "*What Makes Users Refuse Web Single Sign-On? An Empirical Investigation of OpenID*". Symposium on Usable Privacy and Security (SOUPS). University of British Columbia Vancouver, BC, Canada

[4] San-Tsai Sun dan Konstantin Beznosov. 2012. "*On the Security of Web Single Sign-On*". Conference on Computer and Communications Security. Kanada: Department of Electrical and Computer Engineering University of British Columbia

[5] Tsyrklevich, Eugene. 2007. "*Single Sign On for the Internet: A Security Story*" BlackHat. Amerika Serikat: Las Vegas

[6] Urue˜na, Manuel dan Christian Busquiel. “Analysis of a Privacy Vulnerability in the OpenID Authentication Protocol”. Spanyol:Universidad Carlos III de Madrid

[7] Zhou, Yuchen dan David Evans. 2014. "SSO Scan: Automated Testing of Web Applications for Single Sign-On Vulnerabilities" In 23rd USENIX Security Symposium, San Diego, 20–22 August 2014. Amerika Serikat:University of Virginia.

[8] Adhika Dwi Firmansyah, Idris Winarno, S.ST, M.Kom. 2010. "Implementasi OpenID Pada EEPIS Network". Surabaya: Jurusan Teknik Informatika Politeknik Elektronika Negeri Surabaya

[9] Fauziah, Septi Andryana. 2009. "Hijacking Session Pada Sistem Keamanan Komputer(Studi Kasus Pencegahan Virus Pada E-Mail)" Jurnal Artificial, ICT Research Center UNAS, Jakarta

[10] Wijaya, Ignatius Kusuma1, Puspaningtyas Sanjoyo Adi2.2012."Sistem Otentikasi Single Sign-On Menggunakan Algoritma Diffie-Hellman Dan Menggunakan Database Parallel Dengan Menggunakan Rmi (Remote Method Invocation)".Seminar Nasional Teknologi Informasi & Komunikasi Terapan 2012 (Semantik 2012)

[11] Abduh, Muhammad. Derry, R. Febrtiandar. "Multi Factor Authentication" <URL: http://ki.stei.itb.ac.id/2013/10/30/multi-factor-authentication/>, juni 2015

[12] Anonymous. "Yahoo meets OpenID Log in to websites with your Yahoo account" <URL: http://openid.yahoo.com/> juni 2015

[13] Anonymous. "What is OpenID?" <URL: http://openid.net/get-an-openid/what-is-openid/> Juni 2015

[14] Sabeel Ansari, Rajeev S.G. and Chandrashekar H.S. "Packet Sniffing: A Brief Introduction." 0278-6648/02/$17.00 2002 IEEE

[15] Anonymous. "Mengaktifkan atau menonaktifkan cookie" <URL:https://support.google.com/accounts/answer/61416?hl=id> juni 2015

[16] ITB Network Information Center. "Email Phishing"<URL:https://nic.itb.ac.id/mail/email-phising > juni 2015

[17] Anonymous. "Access Tokens" <URL: https://developers.facebook.com/docs/facebook-login/access-tokens> juni 2015

BAB 5

PENUTUP

5.1 Kesimpulan

Berdasarkan hasil analisa dan pengujian yang telah dilakukan terhadap system yang protokol *OpenID*, dapat ditarik beberapa kesimpulan:

1. *Website Relying Party* yang menggunakan protokol *HTTP* membuat semua paket data yang dikirim dan diterima oleh pengguna dapat dilihat oleh pihak lain. Hal tersebut membuat pengguna menjadi rentan menjadi korban tindakan *Session Hijacking* dan tindak pembajakan akun pengguna yang ada di *Relying Party*.

   Ketika akun pengguna bisa diambil alih oleh pihak lain, akan menimbulkan ancaman baru yang bisa merugikan pengguna, yaitu pencurian kode token. Ketika kode token berhasil dicuri, maka data-data pengguna yang ada di *provider* juga ikut terancam dan bisa disalahgunakan. Terutama untuk *Relying Party* yang mendapatkan hak otorisasi yang besar dari *provider*.
2. Kesalahan *Relying Party* dalam mengelola dan menyimpan kode token yang diberikan oleh *provider* juga dapat mengancam keamanan data pengguna di *provider*. Karena kesalahan tersebut, kode token lebih mudah dicuri dan dapat disalahgunakan oleh pihak lain.
3. Proses pengalihan ketika otentikasi pengguna menimbulkan celah keamanan yang bisa membuat pengguna terjebak dalam tindakan *phishing*. Pengguna rentan dialihkan menuju *website* palsu yang dibuat mirip dengan halaman *website provider*.
4. Kelemahan sistem keamanan yang digunakan *Relying Party* juga bisa membahayakan data-data pengguna yang ada di *provider*.
5. Facebook dan Google telah menerapkan aturan baru dengan mengharuskan *Relying Party* mendaftarkan alamat *URL* pengalihan setelah otentikasi pengguna, sehingga bisa

menghindari pengguna dari penyalahgunaan *AppID/Client ID* dan *App Secret/Client Secret* atau yang lebih dikenal dengan teknik *Covert Redirect*.

6. Regulasi untuk membatasi hak otorisasi yang diberlakukan oleh Facebook akan meningkatkan tingkat kepercayaan pengguna terhadap *Relying Party* serta juga bisa menghindari pengguna dari penyalahgunaan otorisasi oleh *Relying Party*

5.2 Saran

Saran untuk pengembangan tugas akhir ini adalah:

a. Proses *sniffing* harus lebih mendalam untuk mendapatkan semua parameter-parameter yang digunakan (selain *cookies*) dalam proses otentikasi dan dicoba *login* dengan menggunakan parameter-paramter tersebut.
b. Selain melakukan *pasive sniffing*, coba lakukan *active sniffing* untuk mengirim parameter otentikasi palsu kepada *Relying Party*.
c. Membuat fitur baru dalam *OpenID* dimana *provider* hanya bertindak sebagai otentikator dan tidak memberikan hak kepada *Relying Party* untuk mengakses data-data pengguna yang ada di *provider*, dengan tujuan untuk mengurangi dampak buruk dari pencurian kode token.