IDENTIFIKASI DAN ANALISIS CELAH KEAMANAN PADA LAYANAN MEDIA PEMBELAJARAN BERBASIS WEB SERVICE

Danny Putra H.T[1],Eko Sakti P., S.Kom, M.Kom[2], Kasyful Amron, S.T, M.Sc[3]
[1,2,3]Teknik Informatika, Program Teknologi Informasi dan Ilmu Komputer, Universitas Brawijaya
Jl. Veteran No. 8 Malang, Informatika, Gedung A PTIIK-UB
Email: dannyputraht@gmail.com[1], ekosakti@ub.ac.id[2], kasyful@ub.ac.id[3]

ABSTRAK

Web service adalah suatu model pemrograman terdistribusi (*distributed computing*) yang pada masa ini telah diimplementasikan secara luas. Banyak kelebihan yang ditawarkan oleh *web service* terutama *interoperabilitas* yang tinggi serta mendukung komunikasi antar aplikasi dan integrasi aplikasi dengan menggunakan beberapa teknologi pendukung salah satunya adalah REST yang dapat berjalan pada protokol *HTTP*. Dengan teknologi tersebut *web service* menawarkan kemudahan untuk menjadi fasilitator pertukaran informasi antar layanan *web* walaupun memiliki *platform* yang berbeda. Di sisi lain kemudahan yang ditawarkan oleh *web service* haruslah didukung dengan sistem keamanan yang baik. Terlebih *web service* yang digunakan untuk layanan publik seperti *e-health*, *e-commerce*, *e-learning*, dan lain-lain yang berisikan informasi penting pengguna, haruslah mendapatkan perhatian lebih untuk keamanannya untuk menjaga layanan-layanan yang digunakan dalam keadaan aman.

Pada penelititan ini akan dilakukan *vulnerability assessment* atau penilaian kerentanan terhadap aplikasi *web* yang mengimplementasikan *web service* dengan cara melakukan pengujian yang berpedoman pada *OWASP testing guide* dengan bantuan beberapa *software* pendukung seperti *burp suite proxy*, *wireshark*, *cookies manager*, *nmap* dan ditambah pengujian menggunakan *software vulnerability scanner* yaitu *acunetix* untuk menambah lengkap pencarian dan pengujian celah keamanan. Dengan melakukan kombinasi pengujian menggunakan beberapa *software* pendukung diharapkan hal ini dapat meningkatkan area pengujian menjadi lebih luas.

Kata Kunci: *Web Service*, *Vulnerability*, *Vulnerability Assessment*, *Burp Suite Proxy*, *Wireshark*, *OWASP-Testing Guide*, *nmap*, *Acunetix Vulnerability Scanner*

ABSTRACT

Web service is one of distributed computing model that nowadays have been widely implemented. many advantages offered by web service espceially high interoperability and to support communication between applications and the integration of applications by using some assistive technologies, one of which is REST that able to run on HTTP protocol. With this technologi, web service offers easy to be an information exchange facilitator between web services despite having different platforms. On the other side of the convenience offered, web service must be supported by a good security system. Moreover, a web service that is used for public services such as e-health, e-commerce, e-learning, and others containing important user's information, should get more attention for their security to keep the services that are used in safe condition.

This research will be conducted vulnerability assessments againts a web application that implements web service by doing testing based on the OWASP testing guide with the help of some supporting software such as burp suite proxy, wireshark, cookies manager, nmap and additionally testing using vulnerability scanner software, Acunetix to add a complete search and testing of security loopholes. By doing this combination of tests using some supporting software is expected that it could increase the area of testing becomes more widespread.

Keywords**:** *Web Service*, *Vulnerability*, *Vulnerability Assessment*, *Burp Suite Proxy*, *Wireshark*, *OWASP-Testing Guide*, *nmap*, *Acunetix Vulnerability Scanner*.

1. LATAR BELAKANG

Web service adalah suatu model pemrograman terdistribusi (*distributed computing*) yang pada masa ini telah diimplementasikan secara luas (Djuandi, 2010). Banyak kelebihan yang ditawarkan oleh *web service* terutama *interoperabilitas* yang tinggi serta mendukung komunikasi antar aplikasi dan integrasi aplikasi dengan menggunakan beberapa teknologi pendukung seperti *REST* (*Representional State Transfer*) yang dikirim melalui *HTTP* (*Hypertext Transport Protocol*). Dengan teknologi tersebut *web service* menawarkan kemudahan untuk menjadi fasilitator pertukaran informasi antar layanan *web* walaupun memiliki *platform* yang berbeda.

Disisi lain kemudahan yang ditawarkan oleh *web service* haruslah didukung dengan sistem keamanan yang baik. Sayangnya banyak *developer* saat ini lebih berorientasi terhadap mekanisme performasi layanan yang dihasilkan oleh suatu aplikasi dan sedikit memperhatikan masalah keamanan aplikasi mereka. Tidak jarang terdapat anggapan bahwa pengimplementasian keamanan dapat menganggu performasi yang dihasilkan sehingga sering kali sisi keamanan tidak mendapatkan perhatian yang baik. Selain itu tidak semua pemilik *website* memiliki latar belakang pengetahuan IT sehingga kemanan *website* tidak menjadi fokus utama hingga terjadi insiden (Rahardjo, 2002). Terlebih *web service* yang digunakan untuk layanan publik seperti *e-health*, *e-commerce*, *e-learning*, dan lain-lain yang berisikan informasi penting pengguna, haruslah mendapatkan perhatian lebih untuk keamanannya untuk menjaga layanan-layanan yang digunakan dalam keadaan aman. Dengan sifat *web service* yang terbuka dalam berinteraksi antar sistem dalam layanan, *web service* menjadi rentan terhadap berbagai bentuk serangan. Bentuk serangan tersebut seperti *XML Inspection*, *XSS Inspection*, *HTTP header* manipulatif, serangan *denial of service*, *replay attack*, dan lain sebagainya. Oleh karena itu masalah keamanan masih menjadi masalah utama dalam pengimplementasian *web service* dalam berbagai bidang.

Dari beberapa permasalahan tersebut perlu adanya suatu kajian mendalam mengenai identifikasi dan analisis celah keamanan pada layanan *web* yang memanfaatkan teknologi *web service* untuk memastikan keamanan layanan-layanan yang disediakan oleh *web service*. Terdapat beberapa aspek keamanan pada *web service* yang perlu dipenuhi untuk menjamin keamanan dari suatu layanan yang disediakan oleh suatu *web service*, aspek-aspek keamanan tersebut yaitu *authentication*, *authorization*, *confidentiality*, *integrity*, *availability,* dan *non-repudiation* (Banaei & Khorsandi, 2012).

Pada penelitian ini penulis akan melakukan *vulnerability assessment* atau yang dapat disebut juga dengan penilaian celah keamanan atau kerentanan terhadap *website* yang mengimplementasikan *web service* dalam membantu menjalankan layanan-layanan yang disediakan sebagai cara untuk menguji dan menganalisis celah keamanan yang terdapat pada *web service*. Terdapat beberapa langkah yang dilakukan dalam melakukan *vulnerabiity assessment* yaitu pengambilan data, dan pengolahan data celah keamanan. Pengambilan data dilakukan dengan dua fase, fase pertama adalah *passive mode* yang merupakan fase dimana penguji mengumpulkan informasi sebanyak mungkin dari target yang diuji dari berbagai sumber yang dibutuhkan, fase yang kedua adalah *active mode* yaitu pengujian parameter keamanan yang telah ditentukan oleh OWASP testing guide kemudian dilanjutkan dengan scanning celah keamanan menggunakan bantuan *software acunetix*, untuk menambah kelengkapan hasil celah keamanan yang terdeteksi. Setelah proses pengujian selesai dilakukan selanjutnya dilakukan pengolahan data untuk membuat daftar celah keamanan yang terdeteksi dari hasil pengujian.

Hasil dari penelitian ini adalah berupa nilai kerentanan dari aplikasi dan juga solusi untuk setiap kerentanan yang terdeteksi. Dari hasil *vulnerability assessment* celah keamanan pada *web service* tersebut diharapkan dapat meningkatkan keamanan dan meminimalisir celah keamanan yang ada pada *website* yang mengimplementasikan *web service* dalam menjalankan layanan-layananya.

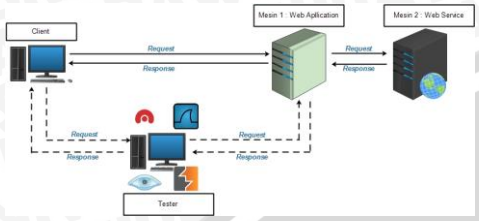
2. METODOLOGI PENELITIAN

2.1 Studi Literatur

Dasar teori diperoleh dari penelitian relevan yang dilakukan sebelumnya dan terkait dengan penelitian yang dilakukan oleh penulis, seperti dari jurnal, *e-book*, dan situs *web* resmi yang dapat dipertanggungjawabkan, antara lain *OWASP testing Guide*, *Burp Suite Proxy*, *Wireshark*, *Software Acunetix*, *Vulnerability Assessment*, nmap, dan *Penetration Testing*.

2.2 Lingkungan Penelitian

Lingkungan penelitian yang digunakan pada penelitian ini adalah dengan menggunakan beberapa mesin pendukung dan beberapa *tools* atau *software* untuk proses pengujian celah keamanan. Berikut merupakan gambaran lingkungan penelitian yang digunakan pada penelitian ini :



Gambar 2. 1 Lingkungan Penelitian

Gambar 2.1 merupakan gambaran logikal dari lingkungan penelitian yang digunakan dalam penelitian ini. *Client* pada gambar diatas merupakan pengguna sah yang akan mengakses aplikasi web yang akan diuji, sedangkan *tester* merupakan penguji yang melakukan pengujian terhadap aplikasi web dibantu dengan beberapa *software* pendukung yaitu *burp suite proxy* sebagai proxy, *nmap* berperan untuk melakukan *scanning system* atau *platform*, *wireshark* berperan sebagai paket *sniffing*, dan *software* pendukung yang terakhir adalah *acunetix vulnerability scanner* yang berperan dalam pengujian secara otomatis. Sedangkan mesin 1 berperan sebagai aplikasi web yang akan diuji, aplikasi web tersebut berisikan *user interface* dari aplikasi media pembelajaran yang diuji. Dan mesin 2 berperan sebagai *web service* dari aplikasi web yang akan diuji, *web service* berperan sebagai pengelolah layanan-layanan yang diberikan oleh web aplikasi yang diuji.

2.3 Metodologi Penelitian

Metodologi yang digunakan dalam penelitian Identifikasi dan Analisis Celah Keamanan Pada Layanan Media Pembelajaran Berbasis *Web Service* ini adalah melalui beberapa langkah yaitu pengambilan data yang dilakukan dengan dua fase yaitu *passive mode*(*information gathering*) dan *active mode*(pengujian parameter keamanan berdasarkan OWASP dan pengujian dengan *software vulnerability scanner*), dan pengolahan data.

2.3.1 Pengambilan Data

Pengambilan data merupakan tahap awal dari metodologi yang digunakan dalam penelitian Identifikasi dan Analisis Celah Keamanan Pada Layanan Media Pemebelajaran Berbasis *Web Service* yaitu dengan melakukan dua fase utama yaitu *passive mode* dan *active mode* setelah kedua fase tersebut dilakukan akan ditambah dengan melakukan scanning celah keamanan menggunakan *software* pendukung yaitu *acunetix vulnerability scanner*. Kedua pengujian tersebut dilakukan agar celah keamanan yang di dapatkan lebih lengkap dan menyeluruh, karena mungkin akan terdapat celah keamanan yang tidak terdeteksi dengan pengujian beberapa parameter yang ditentukan oleh *OWASP* namun dapat terdeteksi dengan bantuan *software vulnerability scanner*.

2.3.1.1 *Passive Mode*

Fase pertama atau yang disebut juga dengan *passive mode* dalam identifikasi celah keamanan adalah dengan mengumpulkan informasi sebanyak mungkin dari *target* aplikasi yang akan diuji atau yang disebut juga dengan *information gathering*. *Information gathering* merupakan tahap yang sangat berguna dan diperlukan dalam melakukan identifikasi celah keamanan. *Information gathering* dapat dilakukan dengan berbagai cara yaitu dengan menggunakan *search engine*, *scanner*, mengirim *HTTP request* yang memungkinkan aplikasi yang diuji untuk memberikan informasi aplikasi atau informasi eror yang mungkin akan berguna dalam pengidentifikasian celah keamanan. Berikut merupakan beberapa cara yang akan dilakukan untuk melakukan *information gathering*:

1. *Spiders*, *Robots*, dan *Crawlers*

   Tujuan dari *spider*, *robots*, dan *crawlers* adalah untuk mengetahui halaman *web* yang diijinkan atau tidak diijinkan untuk diakses secara umum. Pengujian dilakukan dengan menambahkan robots.txt pada bagian belakang url dari aplikasi *web* yang sedang diuji.
2. *Search Engine Discovery/Reconnaissance*

   Search engine seperti google dapat digunakan untuk menemukan masalah yang berkaitan dengan aplikasi *web* yang diuji ataupun kesalahan/eror pada halaman yang telah terekspos secara publik.
3. *Testing Web Application Fingerprint*

   Information gathering dengan cara ini dilakukan untuk mengetahui versi dan jenis web server yang sedang berjalan atau digunakan yang memungkinkan penguji untuk mengetahui kerentanan yang mungkin terdapat pada web server yang digunakan.

2.3.1.2 *Active Mode*

Setelah fase *passive mode* telah selesai dilakukan dengan tujuan untuk mengumpulkan segala informasi yang penting untuk mengenal aplikasi web yang diuji, hal selanjutnya yang akan dilakukan adalah memasuki fase yang kedua yaitu fase *active mode*. Pada fase *active mode* ini penguji akan melakukan pengujian terhadap beberapa parameter yang ditentukan oleh *OWASP testing guide* dalam

menentukan atau menguji keamanan dari aplikasi *web*.

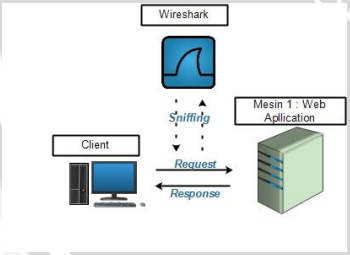
Terdapat beberapa parameter celah keamanan yang penting untuk dilakukan pengujian diantaranya:

1. ***Authentication***:
   a. Pengujian terhadap *transport data*

      Pengujian *transport data* dapat dilakukan dengan bantuan dua software pendukung, yang pertama adalah *burp suite proxy* yaitu dengan mengirimkan http request dari klien menuju aplikasi melalui perantara *burp suite proxy*, dan yang kedua adalah *wireshark* dengan melakukan *sniffing* paket data http saat pengguna melakukan *request login* menuju aplikasi *web*.

Gambar 2.1 *Pengujian Transport Data dengan Burp Suite*

Gambar 2.1 menunjukkan gambaran proses logik dari pengujian *transport data* dimana pengguna akan mengirimkan data berupa *request* halaman *login* dengan menginputkan *username* dan *password* pengguna yang sah menuju aplikasi web yang diuji dengan perantara *burp suite proxy* untuk membaca data yang dikirim.



Gambar 2.2 Pengujian dengan *Wireshark*

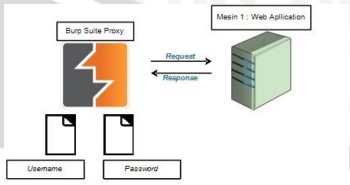
Gambar 2.2 menunjukkan pengujian menggunakan *wireshark* dimana pengguna melakukan login menuju aplikasi web dan wireshark melakukan sniffing paket data yang dikirimkan pengguna menuju aplikasi.

   b. Pengujian terhadap *user enumeration*

      Pengujian *user enumeration* dilakukan dengan cara menginputkan kombinasi *username* dan *password* yaitu *username* benar dan *password* benar, *username* salah dan *password* benar, *username* benar dan *password* salah, *username* salah dan *password* salah, dan *username* benar dan *password* benar namun pengguna belum aktif.

   c. Pengujian terhadap *default* dan *guessable user account* (*dictionary attack*)

      Pengujian *default* dan *guessable user account* dilakukan dengan melakukan dictionary attack pada aplikasi web yang diuji. *Dictionary attack* dilakukan dengan menggunakan bantuan *burp suite proxy* dan kumpulan *username* dan *password* yang telah umum digunakan dalam dua buah *file* berbeda.



Gambar 2.3 Pengujian *Dictionary Attack*

Gambar 2.3 meunjukkan pengujian default dan guessable account yang dilakukan dengan bantuan burp suite proxy menuju aplikasi web yang diuji.

   d. Pengujian terhadap *bypassing authentication schema*

      Pengujian terhadap *bypassing authentication schema* dilakukan dengan melakukan *direct page* menuju halaman setelah *login* tanpa pengguna melakukan *login* terlebih dahulu dan dengan melakukan prediksi terhadap id sesi yang akan muncul atau digunakan pengguna berikutnya.

   e. Pengujian terhadap proses *logout*

      Pengujian terhadap proses dilakukan dengan dua cara, cara yang pertama dengan memastikan terlebih dahulu fungsi *logout* terdapat pada setiap halaman aplikasi web yang diuji untuk menghindari pengguna yang telah melakukan *login* tidak ingat untuk mengakhiri sesi mereka, dan cara berikutnya dengan melakukan *logout* keluar dari aplikasi dan memastikan bahwa sesi id telah diakhiri.

   f. Pengujian terhadap captcha

      Pengujian terhadap captcha dilakukan dengan menginputkan captcha yang tidak sesuai dengan yang di sarankan oleh aplikasi *web* saat pengguna melakukan registrasi.

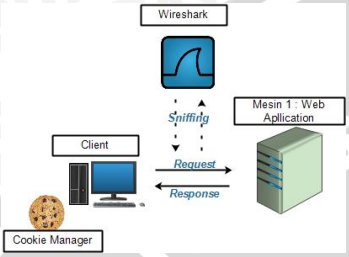
   g. Pengujian fungsi "*remmember me*" dan fungsi *autocomplete* pada *password field*

      Pengujian terhadap fungsi *autocomplete* pada *source code* aplikasi, melihat ke dalam *source code* aplikasi apakah aplikasi web yang

diuji telah menerapkan fungsi *autocomplete="off"*.

h. *Man in The Middle*(MiTM)

Pengujian *man in the middle* dilakukan dengan melakukan *sniffing* data pengguna sah yang melakukan *login* menuju aplikasi *web* yang diuji untuk nantinya *cookies* dari pengguna yang didapat dari proses *sniffing* akan diggunakan penguji untuk melakukan *login* dan mendapatkan otentikasi pengguna yang sah.



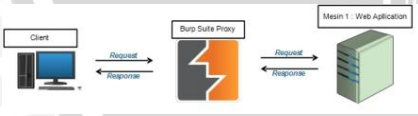
Gambar 2.4 Pengujian *Man in The Middle*

Gambar 2.4 merupakan gambaran proses *sniffing* penguji untuk mendapatkan *cookies* pengguna yang sah dan mengakses halaman pengguna sah tersebut yang seharusnya didapat dengan proses *login*.

2. ***Session Management***:

a. Pengujian terhadap atribut pada *cookies* yaitu atribut *secure*, atribut *HTTPOnly*, atribut domain, atribut *path*, dan atribut *expires*

Pengujian atribut pada *cookies* dilakukan dengan menggunakan *burp suite proxy* dengan mengidentifikasi atribut *cookies* yang dikirimkan.



Gambar 2.5 Pengujian Atribut *Cookies*

Gambar 2.5 merupakan gambar cara pengambilan data cookies menggunakan *burp suite proxy* dengan melakukan login menuju aplikasi melalui perantara *burp suite proxy*.

b. Pengujian terhadap *Session Fixation*

Pengujian dilakukan dengan bantuan *cookies manager* yang merupakan *plug-in* dari *firefox*. Dengan bantuan *cookies manager* akan menjadi mungkin untuk melihat sesi id dari pengguna yang melakukan *login* ke dalam aplikasi.

c. Pengujian terhadap CSRF

Pengujian serangan CSRF dilakukan dengan mengirimkan file CSRF yang dimaksudkan untuk mengganti password pengguna yang sah saat pengguna telah melakukan otentikasi dan mengeksekusi file CSRF yang dikirimkan.

d. Pengujian terhadap *framming*

Pengujian *framming* dilakukan dengan membuat *file* yang berisikan *iframe* pengujian untuk serangan *framming* pada aplikasi *web* yang diuji.

3. ***Authorization***:

a. Pengujian terhadap *path traversal*

Pengujian *path traversal* dilakukan dengan menginputkan dengan memasukkan string berbahaya "../../../../etc/passwd" untuk mengakses *password file hash* dari sistem Linux/Unix.

b. Pengujian terhadap *bypassing authorization schema*

Pengujian yang dilakukan adalah dengan memanggil *URL* yang diperuntukkan untuk dosen melalui akun pengguna biasa(mahasiswa).

c. Pengujian terhadap *eskalasi privilege*

Pengujian yang dilakukan untuk menemukan celah kemanan pada *privilage* adalah melakukan modifikasi level pengguna pada *header* saat pengguna melakukan *request* pada *server*. Pada pengujian ini digunakan *burp suite proxy* untuk membantu melakukan modifikasi pada saat proses login.

Gambar 2.6 Pengujian Eskalasi Privilage

Gambar 2.6 menunjukkan pengujian *eskalasi privilage* menggunakan bantuan *burp suite proxy* dengan memodifikasi parameter pada data cookies yang didapat dari hasil *burp suite proxy*.

4. ***Data Validation Testing***:

a. Pengujian terhadap serangan *XSS*

Pengujian serangan *XSS* dilakukan dengan menginputkan script berbahaya dengan tujuan untuk mengambil *cookies* pengguna, dan menginputkan kode HTML pada halaman profil pengguna untuk menguji

validasi *input* yang diterapkan pada aplikasi *web* yang diuji.

b. Pengujian terhadap *SQL Injection*

Pengujian dilakukan pada halaman *login* yang dimaksudkan untuk melakukan dengan *SQL injection*.

5. ***Web Service Testing***:

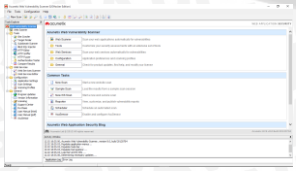
a. Pengujian *endpoint*

Pengujian dilakukan dengan mengakses beberapa *endpoint* yang terdapat pada *web service* untuk mendapatkan data yang terdapat pada *web service* tersebut.

Setelah pengujian parameter keamanan *OWASP* telah selesai dilakukan selanjutnya akan dilakukan *scanning* menyeluruh pada aplikasi web yang diuji dengan menggunakan bantuan *software vulnerability scanner*. *Software vulnerability scanner* yang dipilih dalam pengujian ini haruslah dapat melakukan audit pada aplikasi *web* dengan menguji kerentanan yang ada seperti *SQL Injection*, *Cross-Site Scripting*, dan kerentanan lain yang dapat dieksploitasi secara umum. Pada penelitian ini akan digunakan *software acunetix* yang dirasa telah memenuhi kriteria-kriteria pengujian yang telah disebutkan sebelumnya. *Acunetix* juga dapat melakukan *scan* pada setiap halaman dan situs yang terdapat pada aplikasi *web* yang dapat diakses melalui *web browser* menggunakan protokol *HTTP* ataupun *HTTPS*.

Berikut merupakan langkah dalam proses pengujian dengan metode otomatis pada aplikasi *web* yang diuji:

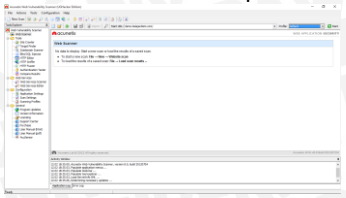
1. Hal pertama yang dilakukan terlebih dahulu adalah melakukan persiapan *software* yang digunakan dengan melakukan *install software acunetix*,
2. Setelah *software acunetix* siap digunakan hal selanjutnya yang dilakukan adalah melakukan *scanning* pada aplikasi *web* yang diuji yaitu mesin 1 untuk menemukan celah keamanan yang ada. Untuk melakukan proses *scanning*, berikut langkah yang dilakukan:
   a. Jalankan *software acunetix* yang telah siap digunakan.



Gambar 2.7 *Software Acunetix*

Gambar 2.7 merupakan *software* acunetix yang telah selesai dilakukan *install* pada laptop penguji dan siap digunakan.

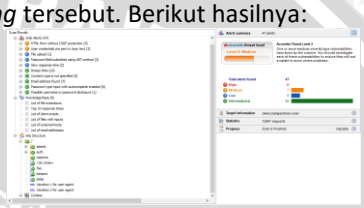
b. Pilih fitur "*web scanner*" untuk melakukan *scanning* pada aplikasi *website* kemudian tekan tombol *start* untuk memulai proses *scanning*.



Gambar 2.8 Fitur *Web Scanner*

Gambar 2.8 merupakan salah satu fitur yang diberikan oleh *software acunetix* yaitu fitur untuk melakukan *scanning* celah keamanan pada aplikasi *web*.

c. Setelah proses *scanning* telah selesai dilakukan, maka akan muncul hasil dari proses *scanning* tersebut. Berikut hasilnya:



Gambar 2.9 Hasil *Scanning*

Gambar 2.9 menunjukkan contoh hasil *scanning* yang telah dilakukan oleh *software acunetix* dalam mendeteksi celah keamanan pada aplikasi *web*.

Setelah hasil *scanning* didapatkan maka akan dilakukan pengujian terhadap hasil *scanning* untuk memastikan hasil pengujian *"false-positive"* dari hasil tersebut.

2.3.2 Pengolahan Data

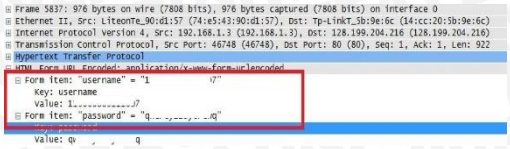
Setelah pengujian selesai dilakukan tahap berikutnya adalah melakukan pengolahan data terhadap data celah keamanan yang terdeteksi. Maksud dari pengolahan data adalah dengan memilih hasil pengujian yang menghasilkan celah keamanan yang terdeteksi, karena tidak semua pengujian yang dilakukan menghasilkan celah keamanan dan kemudian memberikan nilai kerentanan pada setiap celah keamanan yang terdeteksi. Setelah hasil pengolahan data tersebut didapat, maka akan dilakukan analisis untuk melakukan perbaikan dan penambalan terhadap celah keamanan pada layanan *web* tersebut.

3. HASIL

3.1 Hasil Pengambilan Data

1. Pengujian terhadap *Credentials Transport Data*:

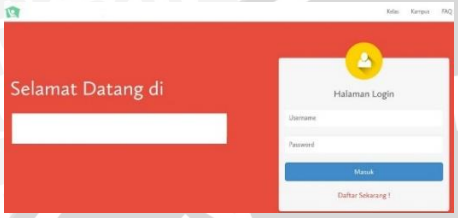
Pengujian pengiriman data otentikasi atau data *credential* pengguna yang dikirimkan telah terenkripsi dan telah terhindar dari pengguna yang tidak bertanggung jawab yang berusaha memotong data yang dikirim.



Gambar 3. 1 Data Login Pengguna

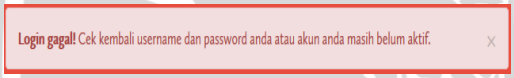
2. Pengujian *User Enumeration* :

Pengujian *response* aplikasi web terhadap penginputan *username* yang benar dan *password* yang salah, penginputan *username* yang benar dan *password* benar namun pengguna belum aktif, dan *username* salah *password* salah pada halaman login :



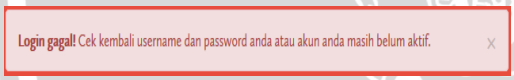
Gambar 3. 2 Halaman Login

a. Pengujian dengan username benar dan password yang salah :



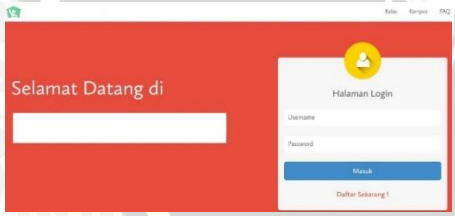
Gambar 3. 3 Response Server

b. Pengujian dengan *username* salah dan *password* salah :



Gambar 3. 4 Response Server

c. Pengujian dengan *username* benar dan *password* benar namun pengguna belum diaktifkan oleh admin. Aplikasi tidak memberikan pesan error:



Gambar 3. 5Response Server

3. Pengujian *CAPTCHA* :

Pengujian fungsi captcha apakah telah berjalan dengan semestinya atau dapat dilakukan bypass terhadap captcha tersebut.

Gambar 3. 6 Captcha pada halaman registrasi

Gambar 3.6 merupakan pengujian dari fungsi captcha. Dilakukan input karakter yang tidak sesuai dengan captcha yang diberikan. Dan hasilnya adalah pada gambar 3.7.

Gambar 3. 7 Proses Registrasi selesai

Gambar 3.7 menunjukkan bahwa registrasi tetap berhasil dilakukan walaupun captcha yang diinputkan tidak sesuai.

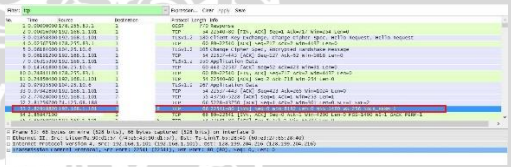
4. Pengujian terhadap *autocomplete password field* :

Pengujian terhadap fungsi *autocomplete* pada password field untuk memastikan fungsi *autocomplete* pada *password field* telah di non-aktifkan.

Gambar 3. 8 Source Code Fungsi Login

5. Pengujian *Man in The Middle* :

Pengujian MiTM dilakukan untuk menguji aplikasi apakah pengguna dengan sesi yang sama dapat melakukan login secara bersamaan.

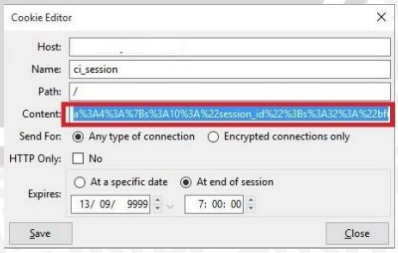


Gambar 3. 9 Sniffing paket data

Gambar 3.9 menjelaskan aplikasi wireshark telah menangkap paket data dari pengguna yang melakukan login pada aplikasi web yang diuji.

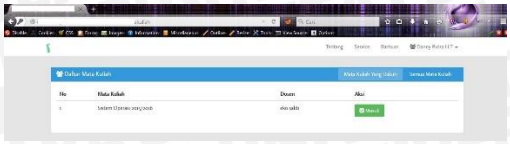
Gambar 3. 10Mendapatkan sesi pengguna sah

Gambar 3.10 menjelaskan bahwa pengguna yang tidak berhak telah mendapatkan sesi id dari pengguna yang sah dengan melakukan sniffing dengan bantuan wireshark.



Gambar 3. 11Input sesi pengguna sah

Gambar 3.11 menjelaskan pengguna yang tidak berhak menginputkan cookie pengguna yang sah yang telah didapat tadi untuk melakukan login sebagai pengguna yang sah.



Gambar 3. 12 Berhasil Login

Gambar 3.12 menjelaskan bahwa pengguna yang tidak sah dapat melakukan login dengan mencuri *cookies* pengguna yang sah melalui proses *sniffing*.

6. Pengujian *attribute* pada *cookies* :

   Pengujian atribut cookies yang penting untuk mengamankan informasi baik yang bersifat sensitif ataupun tidak dari pengguna yang terotentikasi.

   a. HTTPOnly Flag : Atribut untuk mencegah cookies untuk dibaca dan diakses melalui sisi klien, mencegah serangan *CSRF*.
   b. Domain : Atribut untuk membandingkan domain yang dipanggil melalui URL.
   c. Path : Atribut untuk membandingkan sub-domain yang dipanggil.
   d. Secure : Atribut untuk memberi tahu browser untuk hanya mengirimkan cookies hanya jika berasal dari saluran yang aman seperti *HTTPS*.
   e. Expires : Atribut untuk memberikan masa berlaku pada cookies.

Gambar 3. 13Header Response

Gambar 3.13 menjelaskan bahwa pada *header* aplikasi hanya menggunakan atribut path.

7. Pengujian serangan *CSRF* :

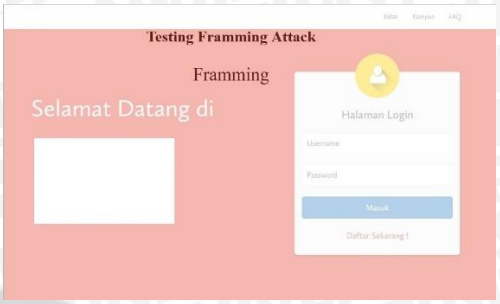
   Pengujian terhadap serangan CSRF dengan menggunakan script sederhana yang dikirimkan pengguna yang tidak bertanggung jawab kepada pengguna sah untuk dalam hal ini mengganti password pengguna yang sah dengan password yang dikehendaki oleh pengguna yang tidak bertanggung jawab.

Gambar 3. 141 File CSRF

Gambar 3.14 merupakan file CSRF yang akan dikirim ke pengguna sah yang terotentikasi.

8. Pengujian serangan *framming* :

   Pengujian framming untuk mengetahui apakah aplikasi web yang diuji rentan terhadap serangan clickjacking. Pengujian dilakukan dengan menggunakan script iframe.

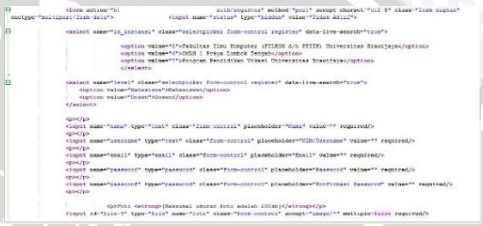


Gambar 3. 15 Serangan Framming

3.1.1 Hasil Pengujian *Acunetix*

1. *HTML for without CSRF Protection* :

   Software acunetix menemukan form yang tidak terlindung terhadap serangan CSRF yaitu pada halaman ../auth/register.



Gambar 3. 16HTML for without CSRF Protection

2. File Upload :

   Software acunetix menemukan halaman ../auth/register mengijinkan pengguna untuk mengupload file menuju server. mengijinkan upload file memungkinkan dapat menyebabkan resiko yang signifikan jika tidak ditangani dengan benar.

Gambar 3. 17Fungsi Upload File

3.2 Pengolahan Data

Dari hasil pengujian secara manual dan otomatis yang telah dilakukan pada layanan media pembelajaran berbasis web service tersebut, telah diketahui bahwa terdapat beberapa celah keamanan yang telah terdeteksi pada aplikasi web yang telah diuji kerentanannya. Berikut akan ditampilkan report celah keamanan dari hasil pengujian yang telah dilakukan dan telah dikelompokkan berdasarkan resiko yang dihasilkan.

Tabel 3. 1 Report Celah Keamanan

Pengujian	Deskripsi	Report	Level
Credentials Transport Data	Verifikasi *transport credential* data pengguna.	Transport data tidak ter-enskripsi	Critical

Man in The Middle	Sniffing paket data pengguna dan melakukan login dengan sesi pengguna	Berhasil mendapatkan cookie pengguna dan melakukan login sebagai pengguna sah	High
Auto complete field password	Verifikasi fungsi auto complete pada password field	Aplikasi web tidak menerapkan autocomplete="off"	Medium
Serangan *Framming*	Pengujian serangan framming/ clickjacking	Aplikasi web dapat diserang dengan serangan *framming*	Medium
User Enumeration	Verifikasi respons aplikasi terhadap inputan username dan password	Aplikasi memberikan respons yang berbeda pada user yang belum aktif	Medium
CAPTCHA	Verifikasi fungsi CAPTCHA	Fungsi CAPTCHA tidak bekerja	Medium
Serangan CSRF	Pengujian serangan CSRF	CSRF dapat dilakukan pada aplikasi web yang diuji	Medium
File Upload	Pengujian fungsi aplikasi yang mengijinkan file upload	Aplikasi web mengijinkan pengguna mengupload file, namun file	Low
		tidak dapat diakses	

Keterangan :

1. **Critical** : Exploitasi celah keamanan yang dapat mendapatkan akses *robot-level*, dan eksploitasi tidak memerlukan manipulasi kepada pengguna yang sah.
2. **High** : Celah keamanan yang dapat menyebabkan akses pada aplikasi terhadap *system*, menyebabkan hilangnya data atau *downtime* terhadap aplikasi.
3. **Medium** : Celah keamanan yang memerlukan informasi pengguna yang sah dengan cara memanipulasi pengguna dengan social engineering.
4. **Low** : Celah keamanan yang memberikan dampak kecil pada aplikasi yang diuji.

4. ANALISIS

Berdasarkan hasil *report* celah keamanan yang telah didapat sebelumnya, hal selanjutnya adalah melakukan analisis untuk menentukan cara penanggulangan yang tepat dan sesuai dalam melakukan antisipasi kemungkinan serangan yang mungkin terjadi akibat celah keamanan yang terdeteksi. Berikut merupakan analisis dari hasil *report* celah keamanan yang terdeteksi :

1. Credential Transport Data

Tabel 4. 1 Analisis Credential Transport Data

Hasil Report	**Critical**
Aplikasi web yang diuji(mesin 1) mengirim data kredensial pengguna saat melakukan login dan registrasi dengan menggunakan *clear text*.	
Analisis Report	
Sebaiknya aplikasi menggunakan enkripsi pada tingkat transportasi seperti SSL atau TLS untuk melindungi semua komunikasi yang sensitif antara klien dan server. Komunikasi yang harus dilindungi termasuk mekanisme login dan fungsi-fungsi yang terkait dengan data sensitif. Jika HTTP *cookies* digunakan untuk transmisi sesi token, maka atribut keamanan harus diterapkan untuk mencegah terjadinya komunikasi yang tidak aman (Scanner, 2013).	

2. *Man in The Middle* (MiTM)

Tabel 4. 2 Analisis MiTM

Hasil Report	**High**
Aplikasi web yang diuji mengijinkan username yang sama untuk login secara	

bersamaan. Wireshark dapat menangkap *traffic* data dalam hal ini cookies pengguna yang sah, dan dengan menggunakan *cookies manager* penyerang dapat mengirim kembali paket (*cookies*) menuju server target.
Analisis Report
Sebaiknya pada aplikasi web yang diuji perlu adanya pencegahan dari serangan seperti ini dengan tidak mengijinkan *cookies* id pengguna yang sama untuk melakukan login disaat bersamaan. (Agarwwal , et al., 2008)

3. Fungsi *autocompete* pada *password field*

Tabel 4. 3 Analisis Fungsi autocomplete

Hasil Report	**Medium**
Fungsi *autocomplete* pada halaman login tidak dinonaktifkan hal ini dapat berbahaya jika pengguna yang terotentikasi menggunakan komputer bersama dalam mengakses aplikasi. *Attacker* dapat dengan mudah login ke dalam aplikasi dengan memilih username yang tersedia karena *autocomplete field* dan juga browser secara otomatis menginputkan password dari username yang dipilih.	
Analisis Report	
Sebaiknya pada form login dan form-form lain yang berhubungan dengan pengisian berbagai inputan, fungsi *autocomplete* harus dinonaktifkan (Scanner, 2013).	

4. Serangan *Framming*

Tabel 4. 4 **Analisis Serangan Framming**

Hasil Report	**High**
Serangan framming dapat dilakukan pada aplikasi ini dengan cara membuat halaman yang tumpang tindih dengan halaman antar muka target. Dengan mengarahkan pengguna yang sah untuk melakukan tindakan tanpa mereka sadar(*clickjacking*).	
Analisis Report	
Untuk melakukan pencegahan terhadap serangan *framming* suatu aplikasi harus memiliki respon *header* dengan X-Frame-Options dengan nilai *DENY*. Sedangkan pada aplikasi web yang diuji respon header aplikasi tidak terdapat X-Frame-Options (Scanner, 2013).	

5. *User Enumeration*

Tabel 4. 5 Analisis User Enumeration

Hasil Report	**Medium**
Pada aplikasi web yang diuji(mesin 1) respons server berbeda saat username yang diinput benar dan password benar namun pengguna belum diaktifkan oleh admin.	
Analisis Report	
Sebaiknya *response* server terhadap masalah di atas segera diperbaiki dengan memberikan *response* yang sama seperti inputan username dan password yang salah. (Agarwwal , et al., 2008)	

6. CAPTCHA

Tabel 4. 6 Analisis Fungsi CAPTCHA

Hasil Report	**Medium**
Captcha pada aplikasi web yang diuji(mesin 1) tidak berjalan dengan semestinya..	
Analisis Report	
Sebaiknya fungsi captcha pada aplikasi yang diuji(mesin 1) diperbaiki untuk menghindari terjadinya Pam pada halaman register(letak fungsi captcha yang diimplementasikan (Agarwwal , et al., 2008).	

7. Serangan *CSRF*

Tabel 4. 7 Analisis Serangan CSRF

Hasil Report	**Medium**
Pada aplikasi web yang diuji(mesin 1) serangan CSRF dapat dilakukan dengan mengirimkan file berbahaya pada pengguna yang telah terotentikasi dalam mengakses aplikasi. Pengguna diberi file yang berfungsi untuk mengganti password dari pengguna tersebut.	
Analisis Report	
Sebaiknya pengguna atau *user* yang terotentikasi tidak sembarangan dalam membuka suatu file yang dikirimkan. (Agarwwal , et al., 2008)	

8. *File Upload*

Tabel 4. 8 Analisis File Upload

Hasil Report	**Low**
Pada aplikasi web yang diuji terdapat fungsi upload file. Fungsi tersebut tidak sepenuhnya berjalan dengan baik, walaupun content-type fungsi tersebut menggunakan	

file gambar (JPG dan yang lainnya) pengguna yang melakukan registrasi dapat mengirimkan file bukan gambar ke server. Hal ini dapat membahayakan jika file yang dikirim merupakan file berbahaya seperti *payload*.
Analisis Report
Berdasarkan hasil perbaikan yang disarankan oleh *software acunetix* sebaiknya aplikasi web yang diuji tidak terlalu membutuhkan fungsi tersebut alangkah baiknya jika fungsi tersebut dihapuskan. (Acunetix, 2012)

5. KESIMPULAN

Kesimpulan yang diambil dari penelitian ini yaitu :

1. Pengambilan data celah keamanan pada aplikasi *web* yang berbasis pada *web service* dapat dilakukan dengan menguji celah keamanan berdasarkan parameter yang telah ditentukan dan untuk melengkapi pencarian celah keamanan digunakan bantuan *software acunetix vulnerability scanner* untuk mendeteksi celah keamanan yang tidak dapat terdeteksi saat pengujian parameter keamanan dilakukan.
2. Pengolahan data dilakukan dengan memilih celah keamanan yang terdeteksi dari hasil pengujian yang telah dilakukan, karena tidak semua hasil pengujian merupakan celah keamanan.
3. Analisis celah keamanan dilakukan dengan melihat hasil pengolahan data celah keamanan dan memberikan solusi terhadap setiap celah keamanan yang terdeteksi dengan berpedoman pada sumber yang dapat dipertanggungjawabkan di setiap solusi yang diberikan.

4. SARAN

Saran yang diperoleh dari hasil penelitian yang dilakukan penulis terkait dengan pengembangan penelitian berikutnya yaitu :

1. Perlu dilakukan penelitian lebih lanjut terhadap metode yang lebih efektif dalam melakukan identifikasi, pengujian, dan analisis celah keamanan pada aplikasi web yang berbasis *web service*,
2. Perlu dilakaukan penambahan parameter pengujian pada metode manual untuk mendapatkan hasil celah keamanan yang lebih menyeluruh,
3. Perlu dilakukan penelitian lebih lanjut terhadap dampak *exploit* yang mungkin terjadi dari setiap celah keamanan yang terdeteksi.

5. DAFTAR PUSTAKA

Acunetix, 2012. *Acunetix Web Vulnerability Scanner.* V8 penyunt. s.l.:Acunetix Ltd.

Agarwwal , A., Belluci, D., Coronel, A. & Paola, S. D., 2008. *OWASP Testing Guide.* 3rd penyunt. s.l.:OWASP Foundation.

Ahmad, A., Ahmad, S. R., Awang, N. F. & Ali, Z. M., 2011. Web Vulnerability Assessment: Outsource Dilemmas. *IEEE.*

Ardiansyah, A., Arifandi, W. & Wicaksono, N., 2010. Keamanan Web Service.

Bose, A. K. & Vibhandik, R., 20115. Vulnerability Assessment of Web Application - A Testing Approach. *IEEE.*

Chandratre, P. & . U., 2014. Security Issues Related to Web Services In E-commerce. *IEEE.*

Djuandi, F., 2010. Web Service Security.

Rahardjo, B., 2002. Keamanan Sistem Informasi Berbasis Internet.

Scanner, B., 2013. *Burp Scanner Sample Report.* [Online]
Available at: https://portswigger.net/burp/samplereport/BurpScannerSampleReport.html#7
[Diakses 8 11 2015].