

APLIKASI PENGUJIAN CELAH KEAMANAN PADA APLIKASI BERBASIS *WEB*



Disusun Oleh:

N a m a : Andi Muhammad Panji Muryandi T
NIM : 13523260

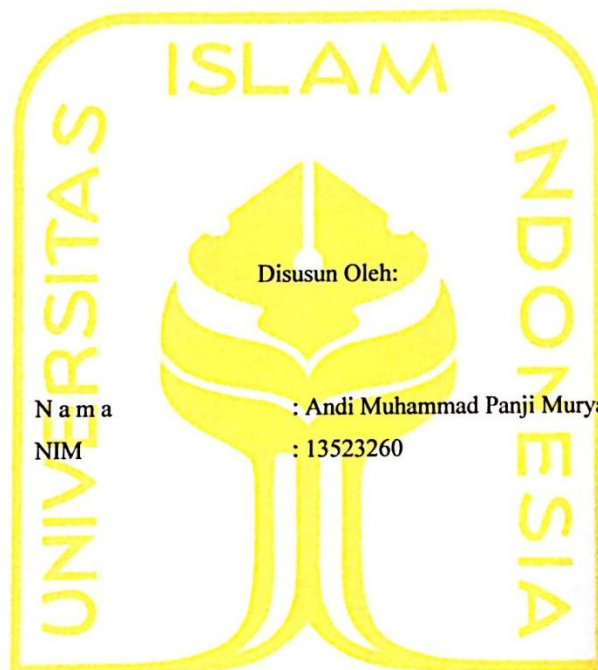
**JURUSAN TEKNIK INFORMATIKA
FAKULTAS TEKNOLOGI INDUSTRI
UNIVERSITAS ISLAM INDONESIA**

2018

HALAMAN PENGESAHAN DOSEN PEMBIMBING

**APLIKASI PENGUJIAN CELAH KEAMANAN PADA
APLIKASI BERBASIS *WEB***

TUGAS AKHIR



N a m a
NIM

: Andi Muhammad Panji Muryandi T

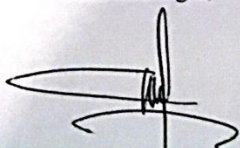
: 13523260

Yogyakarta, 1 Februari 2018

Pembimbing 1,


(Yudi Prayudi, S.Si., M.Kom.)

Pembimbing 2,


(Fietyata Yudha, S.Kom., M.Kom.)

HALAMAN PENGESAHAN DOSEN PENGUJI

**APLIKASI PENGUJIAN CELAH KEMAMAN PADA
APLIKASI BERBASIS *WEB*
TUGAS AKHIR**

Telah dipertahankan di depan sidang penguji sebagai salah satu syarat untuk
memperoleh gelar Sarjana Teknik Informatika
di Fakultas Teknologi Industri Universitas Islam Indonesia
Yogyakarta, 1 Februari 2018

Tim Penguji

Yudi Prayudi, S.Si., M.Kom.

Anggota 1

Fietyata Yudha, S.Kom., M.Kom

Anggota 2

Erika Ramadhani, S.T., M.Eng.

Mengetahui,

Ketua Jurusan Teknik Informatika

Fakultas Teknologi Industri

Universitas Islam Indonesia



Hendrik, S.T., M.Eng.

HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR

Yang bertanda tangan di bawah ini:

Nama : Andi Muhammad Panji Muryandi T

NIM : 13523260

Tugas akhir dengan judul:

APLIKASI PENGUJIAN CELAH KEMANAN PADA APLIKASI BERBASIS *WEB*

Menyatakan bahwa seluruh komponen dan isi dalam tugas akhir ini adalah hasil karya saya sendiri. Apabila dikemudian hari terbukti ada beberapa bagian dari karya ini adalah bukan hasil karya sendiri, tugas akhir yang diajukan sebagai hasil karya sendiri ini siap ditarik kembali dan siap menanggung resiko dan konsekuensi apapun.

Demikian surat pernyataan ini dibuat, semoga dapat dipergunakan sebagaimana mestinya.

Yogyakarta, 1 Februari 2018



(Andi Muhammad Panji Muryandi T)

HALAMAN PERSEMBAHAN

Alhamdulillah Robbil 'Alamin. Segala puji dan syukur atas kehadiran Allah Subhana Wa Ta'ala yang telah memberikan rahmat, ridho, dan karunia-Nya kepada saya. Shalawat serta salam kepada Nabi Muhammad Shallallahu 'Alaihi Wasallam, sebagai pembawa risalah Allah terakhir dan penyempurna seluruh risalah-Nya.

Tugas akhir ini kupersembahkan untuk semua orang yang aku cintai. Kepada Ibu Mestina tersayang yang tidak pernah lelah memberikan kasih sayang, bimbingan akhlak, dan doa dari kecil hingga sekarang. Kepada Ayah Tamzil Patahangi tercinta yang selalu memberikan kasih sayang, nasehat, ajaran, doa, dan tidak pernah lupa mengingatkan untuk selalu melaksanakan shalat 5 waktu dan mengaji.

Kepada sahabat-sahabatku, terima kasih atas segala kebersamaan, bantuan, dukungan, pengalaman, nasehat, dan doa yang telah diberikan.

HALAMAN MOTO

“Barang siapa menginginkan kebahagiaan di dunia dan di akhirat, maka haruslah memiliki banyak ilmu”

(HR. Ibnu Asakir)

“Niscaya Allah akan meninggikan orang-orang yang beriman diantara kamu dan orang-orang yang diberi ilmu pengetahuan beberapa derajat.”

(QS. Al-Mujadillah:11)

KATA PENGANTAR

Assalamu'alaikum Warahmatullahi Wabarakatuh

Dengan mengucapkan *Alhamdulillah*, puji dan syukur atas kehadiran Allah Subhana Wa Ta'ala yang telah memberikan berkat rahmat dan hidayah-Nya, sehingga tugas akhir yang berjudul “**Aplikasi Pengujian Celah Keamanan pada Aplikasi Berbasis Web**” dapat diselesaikan dengan baik. Shalawat serta salam semoga senantiasa dilimpahkan kepada Nabi Muhammad Shallallahu ‘Alaihi Wasallam, yang telah membawa kita dari zaman jahiliyah menuju ke zaman terang benderang.

Laporan tugas akhir ini disusun sebagai salah satu syarat untuk memperoleh gelar Sarjana Strata-1 (S1) di Jurusan Teknik Informatika, Fakultas Teknologi Industri, Universitas Islam Indonesia. Selain itu, tugas akhir ini juga sebagai sarana untuk menerapkan ilmu dan teori yang telah didapatkan selama menjalani masa studi di jurusan Teknik Informatika Universitas Islam Indonesia.

Akhirnya, dengan segala kerendahan hati izinkanlah penulis untuk menyampaikan rasa terima kasih dan penghargaan yang setinggi-tingginya atas motivasi, bantuan, bimbingan, dan doa. Penulis menyampaikan rasa dan penghargaan tersebut kepada :

1. Orang tua yang tidak pernah berhenti memberikan do'a, dukungan, dan motivasi sehingga penulis dapat menyelesaikan laporan tugas akhir ini.
2. Bapak Hendrik, S.T., M.Eng, selaku Ketua Jurusan Teknik Informatika Fakultas Teknologi Industri Universitas Islam Indonesia.
3. Bapak Yudi Prayudi S.Si., M.Kom selaku Dosen Pembimbing 1 tugas akhir yang telah memberikan masukan, arahan, serta dorongan terhadap aplikasi yang dibangun sehingga tugas akhir ini selesai.
4. Bapak Fietyata Yudha S.Kom., M.Kom selaku Dosen Pembimbing 2 tugas akhir yang telah memberikan ide, masukan, dan bimbingan mengenai aplikasi yang dibangun, serta arahan dalam pembuatan laporan tugas akhir.
5. Nelly Nur Hamidah yang selalu memberikan motivasi dan dukungan kepada penulis dalam menghadapi kesulitan-kesulitan selama proses pengerjaan tugas akhir.
6. Sahabat-sahabat saya selama menempuh masa studi khususnya Informatika angkatan 2013 (Eternity) terima kasih atas pengalaman, dukungan, bantuan, dan doa yang diberikan.

7. Kepada semua pihak yang telah membantu baik secara langsung maupun tidak langsung, semoga Allah SWT menjadikannya amal shaleh yang senantiasa mendapatkan balasan dan kebaikan berlipat ganda.

Penulis menyadari bahwa masih terdapat beberapa kekurangan dalam laporan tugas akhir ini, untuk itu saran dan kritik yang sifatnya membangun akan diterima dengan senang hati. Akhir kata, penulis berharap agar laporan tugas akhir ini dapat bermanfaat bagi semua pihak.

Wassalamu'alaikum Warahmatullahi Wabarakatuh

Yogyakarta, 12 Desember 2017

(Andi Muhammad Panji Muryandi T)

SARI

Kali Linux dapat dikatakan sebagai sistem operasi favorit para peretas karena sistem operasi tersebut memiliki banyak *tools* yang dapat dimanfaatkan untuk menguji atau meretas sebuah sistem. Namun sebagian besar penggunaan *tools* yang tersedia di *Kali Linux* masih bersifat manual, seperti *sqlmap* dan *SET (Social Engineering Toolkit)*, dimana pengguna masih harus mengetikkan sintaks untuk menjalankan *tools* tersebut. Hal tersebut tentunya kurang efektif dan efisien mengingat saat ini sudah banyak *library* atau *framework* yang dapat digunakan untuk mempermudah penggunaan *tools* tersebut. Berdasarkan uraian masalah diatas, peneliti mengangkat judul “Aplikasi Pengujian Celah Keamanan pada Aplikasi Berbasis *Web*”.

Aplikasi ini diharapkan dapat membantu administrator atau pemilik layanan untuk melakukan pengujian secara berkala. Tujuan membangun aplikasi pengujian celah keamanan ini adalah untuk membuat pengujian celah keamanan terhadap aplikasi berbasis *web* menjadi lebih efisien dan efektif. Aplikasi ini memiliki 3 jenis serangan untuk melakukan pengujian celah keamanan, diantaranya *SQL Injection*, *Cross-Site Scripting*, dan *Phising*.

Hasil yang dicapai berdasarkan pengujian *White Box*, pengujian *Black Box*, dan pengujian secara manual terhadap aplikasi ini adalah Aplikasi Pengujian Celah Keamanan pada Aplikasi Berbasis *Web* mampu membuat pengujian celah keamanan terhadap aplikasi berbasis *web* menjadi lebih efektif dan efisien.

Kata kunci: Aplikasi *Python*, *PyQt5*, Pengujian celah keamanan.

GLOSARIUM

Backdoor	Mekanisme yang dapat digunakan untuk mengakses sistem, aplikasi, atau jaringan.
Browser	Perangkat lunak yang digunakan untuk menjelajah dan menelusuri situs-situs di internet.
Chat	Pesan obrolan.
Compiler	Proses untuk menerjemahkan bahasa pemrograman menjadi berkas yang siap dieksekusi oleh sistem operasi.
Crawling	Sebuah teknik pada <i>website</i> yang digunakan untuk memindai halaman dari sebuah <i>website</i> .
Database	Basis Data.
E-mail	Pesan elektronik.
Hacking	Peretasan terhadap sebuah sistem.
Hosting	Tempat untuk menampung dan menyimpan data-data <i>website</i> sehingga <i>website</i> tersebut dapat diakses melalui internet.
IP Address	Identitas pada sebuah komputer dan alat komputer yang digunakan sebagai sarana komunikasi.
Library	Pustaka yang siap dipakai dalam pengembangan aplikasi.
Login	Proses untuk mengakses sebuah sistem atau komputer dengan memasukkan identitas pengguna.
Malware	Program yang dirancang untuk merusak dan menyusup ke dalam sistem komputer.
Open source	Pengembangan kode yang bersifat terbuka.
Password	Kata sandi.
Path	Istilah yang digunakan untuk menunjukkan alamat dari sebuah <i>file</i> .
Penetration testing	Sebuah metode yang dilakukan untuk menguji keamanan dari sebuah sistem.
Script	Naskah kode.
Server	Sebuah sistem komputer yang menyediakan layanan tertentu.
Username	Nama pengguna.
Vulnerability	Kerentanan atau Celah.
Website	Situs.

DAFTAR ISI

HALAMAN JUDUL	i
HALAMAN PENGESAHAN DOSEN PEMBIMBING.....	ii
HALAMAN PENGESAHAN DOSEN PENGUJI.....	iii
HALAMAN PERNYATAAN KEASLIAN TUGAS AKHIR Error! Bookmark not defined.	
HALAMAN PERSEMBAHAN	v
HALAMAN MOTO	vi
KATA PENGANTAR	vii
SARI	ix
GLOSARIUM.....	x
DAFTAR ISI.....	xi
DAFTAR TABEL.....	xiii
DAFTAR GAMBAR	xiv
BAB I PENDAHULUAN.....	1
1.1 Latar Belakang.....	1
1.2 Rumusan Masalah	3
1.3 Batasan Masalah.....	4
1.4 Tujuan Penelitian.....	4
1.5 Manfaat Penelitian.....	4
1.6 Metodologi Penelitian.....	4
1.7 Sistematika Penulisan.....	5
BAB II LANDASAN TEORI.....	7
2.1 Penelitian Terdahulu.....	7
2.2 Open Web Application Security Project (OWASP).....	8
2.3 Pengertian Vulnerability.....	9
2.4 Jenis-jenis Serangan	11
2.4.1 <i>SQL Injection</i>	11
2.4.2 <i>Phishing</i>	11
2.4.3 <i>Cross-Site Scripting (XSS)</i>	14
2.5 Peninjauan Aplikasi Sejenis	15
BAB III METODOLOGI PENELITIAN	17
3.1 Metode Penelitian.....	17
3.2 Analisa Kebutuhan Aplikasi.....	17

3.2.1 Analisis Kebutuhan Perangkat Lunak.....	17
3.2.2 Analisis Kebutuhan Masukan	18
3.2.3 Analisis Kebutuhan Proses.....	18
3.2.4 Analisis Kebutuhan Keluaran	19
3.3 Perancangan Aplikasi	19
3.3.1 Flowchart	19
3.3.2 Pseudocode.....	20
3.3.3 Desain Antarmuka.....	23
3.4 Implementasi	24
3.5 Pengujian dan Evaluasi.....	25
3.5.1 Pengujian <i>White Box</i>	25
3.5.2 Pengujian <i>Black Box</i>	27
3.5.3 Evaluasi.....	27
BAB IV IMPLEMENTASI DAN PENGUJIAN.....	28
4.1 Implementasi Aplikasi.....	28
4.1.1 Implementasi Antarmuka	28
4.1.2 Implementasi Proses Serangan <i>Phising</i>	29
4.1.3 Implementasi Proses Serangan <i>SQL Injection</i>	30
4.1.4 Implementasi Proses Serangan <i>Cross-Site Scripting (XSS)</i>	31
4.2 Pengujian Aplikasi.....	32
4.2.1 Pengujian <i>White Box</i>	33
4.2.2 Pengujian <i>Black Box</i>	37
4.2.3 Pengujian Secara Manual	39
4.2.4 Perbandingan terhadap Aplikasi Sejenis.....	41
4.2.5 Pengujian terhadap beberapa Jenis <i>Website</i>	45
BAB V KESIMPULAN DAN SARAN	47
5.1 Kesimpulan.....	47
5.2 Saran.....	47
DAFTAR PUSTAKA	48
LAMPIRAN.....	50

DAFTAR TABEL

Tabel 2.1 Penelitian Terdahulu	8
Tabel 2.2 Tabel Perbandingan <i>OWASP TOP</i> 10-2013 dengan <i>OWASP TOP</i> 10-2017.....	9
Tabel 3.1 Tabel <i>Library</i>	21
Tabel 4.1 Daftar Kelompok dan Alamat Target	32
Tabel 4.2 Hasil Pengujian terhadap Seluruh Target	35
Tabel 4.3 Tabel Hasil Perhitungan Kuisisioner.....	44
Tabel 4.4 Hasil Pengujian terhadap beberapa Jenis <i>Website</i>	46

DAFTAR GAMBAR

Gambar 2.1 Contoh Skenario Serangan <i>Phising</i>	13
Gambar 2.2 Kerja Serangan <i>Cross-Site Scripting (XSS)</i>	15
Gambar 3.1 <i>Flowchart</i> Aplikasi Aplikasi Celah Keamanan pada Aplikasi Berbasis Web	20
Gambar 3.2 <i>Pseudocode</i> Aplikasi Pengujian Celah Keamanan	22
Gambar 3.3 Aplikasi Aplikasi Celah Keamanan pada Aplikasi Berbasis Web	24
Gambar 4.1 Halaman Awal Aplikasi Pengujian Celah Keamanan.....	29
Gambar 4.2 Kode Program Serangan <i>Phising</i>	30
Gambar 4.3 Kode Program Serangan <i>SQL Injection</i>	31
Gambar 4.4 Kode Program Serangan <i>Cross-Site Scripting (XSS)</i>	32
Gambar 4.5 Hasil Pengujian terhadap Alamat <i>http://nightgallery.ca</i>	35
Gambar 4.6 Hasil <i>file</i> “Pengujian1.html”	35
Gambar 4.7 Hasil Pengujian terhadap Alamat <i>http://yutub.com</i>	37
Gambar 4.8 Hasil Pengujian terhadap Alamat <i>http://kemenhan.com</i>	37
Gambar 4.9 Hasil Pengujian Tombol <i>Select Folder</i>	38
Gambar 4.10 Hasil Pengujian Tombol <i>Attack !</i>	38
Gambar 4.11 Hasil Pengujian Performa Aplikasi.....	39
Gambar 4.12 Hasil Inspeksi pada Halaman <i>Login http://ittaherl.com</i>	40
Gambar 4.13 Hasil <i>Phising</i> Halaman <i>Login http://ittaherl.com</i>	40
Gambar 4.14 Hasil Serangan <i>SQL Injection</i>	40
Gambar 4.15 Hasil Serangan <i>Cross-Site Scripting (XSS)</i>	41
Gambar 4.16 Hasil Serangan <i>Cross-Site Scripting (XSS)</i>	41
Gambar 4.17 Duplikasi Halaman <i>http://readyicons.com</i> menggunakan <i>SET</i>	43
Gambar 4.18 Kesalahan pada <i>Tools SET</i>	43

BAB I

PENDAHULUAN

1.1 Latar Belakang

Keamanan merupakan salah satu aspek yang sangat penting dari sebuah sistem informasi (Dharmawan, Eka Adhitya , dkk, 2013). Keamanan komputer adalah sebuah tindakan pencegahan yang dilakukan untuk melindungi sebuah sistem dari serangan yang dilakukan oleh pihak-pihak yang tidak bertanggung jawab. Tanpa keamanan yang baik, maka data-data pengguna belum dapat dikatakan aman. Hal tersebut pastinya akan mempengaruhi kepercayaan pengguna dalam menggunakan sistem tersebut.

Di era teknologi informasi saat ini, sistem memiliki peran yang besar bagi perusahaan ataupun institusi, baik untuk mendapatkan keuntungan secara finansial maupun untuk memberikan pelayanan yang baik kepada pengguna sistem. Hal tersebut membuat perusahaan-perusahaan ataupun instansi bersaing dalam menyediakan layanan terbaik. Namun, seringkali yang menjadi prioritas mereka adalah tampilan sistem dan layanan yang dapat memikat pengguna dengan cepat, sedangkan masalah keamanan berada di urutan bawah atau tidak dianggap begitu penting (Dharmawan, Eka Adhitya , dkk, 2013).

Padahal keamanan data sangat dibutuhkan untuk menjaga dan melindungi privasi pengguna. Ketika data privasi pengguna bocor atau jatuh ke tangan pihak-pihak yang tidak bertanggung jawab, maka rasa kepercayaan pengguna terhadap penyedia layanan pun akan runtuh. Banyak peristiwa yang telah terjadi terkait peretasan atau pembobolan sistem karena minimnya perhatian penyedia layanan terhadap keamanan sistem tersebut, seperti kasus peretasan yang terjadi pada situs Komisi Pemilihan Umum (KPU) Kota Yogyakarta yang terjadi pada bulan Februari 2017 dan kasus peretasan situs Telkomsel yang terjadi beberapa bulan lalu.

Dilansir dari berita elektronik pada situs *kompas.com*, Kepala Subdirektorat IV *Cyber Crime* Ditreskrimsus Polda Metro Jaya, Ajun Komisaris Audie Latuheru menyebutkan bahwa kasus kejahatan peretasan (*hacking*) yang terjadi di Indonesia telah mencapai 30% (Gesita, 2013). Selain itu, dilansir dari berita elektronik pada situs *tempo.co*, Wakil Direktur Tindak Pidana Ekonomi Khusus Bareskrim Kepolisian Indonesia, Komisaris Besar Polisi Agung Setya mengatakan bahwa sejak tahun 2012 hingga Agustus 2015 tercatat ada 36,6 juta serangan yang terjadi didalam jaringan Indonesia. Jumlah serangan tersebut telah merugikan negara hingga Rp 33,29 Milliar (Nurrahman, 2015).

Salah satu sistem yang umumnya menjadi sasaran *hacker* dan *cracker* adalah aplikasi berbasis *website*. Hal tersebut dikarenakan pemanfaatan aplikasi mengalami pertumbuhan yang sangat pesat saat ini. *Hacker* adalah seseorang yang melakukan peretasan dengan mencari celah keamanan dari sebuah sistem, kemudian memberikan gagasan dan solusi kepada administrator sistem bila terdapat celah keamanan. Sedangkan *cracker* adalah seseorang yang melakukan peretasan dengan mencari celah keamanan dari sebuah sistem, kemudian menggunakan celah tersebut untuk kepentingan individu seperti mencuri data, menghapus data, merusak data, dan melakukan hal-hal lain yang merugikan pemilik sistem.

Banyak jenis serangan yang umumnya digunakan untuk mencari celah keamanan pada sebuah *website*. Beberapa jenis serangan tersebut diantaranya *SQL Injection*, *Phising*, dan *Cross-Site Scripting (XSS)*. *SQL Injection* merupakan jenis serangan yang dilakukan dengan cara memodifikasi perintah *SQL*. Berdasarkan dokumen *OWASP (Open Web Application Security Project)* tentang 10 resiko kewanamanan aplikasi *website* yang paling kritis, serangan *SQL Injection* berada diposisi teratas. Di Indonesia, teknik ini pun pernah digunakan oleh seorang *hacker* dengan kode nama “Xnuxer” untuk melakukan peretasan pada hasil Tabulasi Nasional Pemilu (TNP) KPU pada tahun 2004 (detikinet, 2014). Dengan menggunakan teknik *SQL Injection*, “Xnuxer” berhasil mengubah nama-nama partai peserta Pemilu menjadi Partai Jambu, Partai Nanas, dan nama-nama asal lainnya.

Phising merupakan sebuah teknik yang digunakan untuk mendapatkan data penting seperti *username*, *password*, dan data penting lainnya dari pengguna dengan cara mengelabui pengguna untuk menggunakan situs palsu yang menyerupai situs asli. Biasanya, *hacker* yang menggunakan teknik ini mengelabui korban dengan cara mengirimkan *e-mail* atau *chat* yang berisi tautan alamat situs palsu. Secara umum tampilan situs palsu dengan situs asli memang terlihat sama, namun alamat dari situs palsu tersebut biasanya berbeda dengan situs yang asli.

Cross-Site Scripting (XSS) adalah jenis serangan yang dilakukan oleh *hacker* dengan cara menyisipkan *script* kedalam sebuah *website*. *Cross-Site Scripting (XSS)* berada diposisi ketujuh dalam dokumen *OWASP* tentang 10 resiko kewanamanan aplikasi *website* yang paling kritis tahun 2017. Beberapa hal yang dapat dilakukan oleh *Cross-Site Scripting (XSS)*, seperti dapat mengalihkan halaman pengguna secara otomatis ketika pengguna membuka sebuah halaman *website* dan menampilkan *script* peringatan (*alert*) ketika pengguna membuka sebuah alamat *website*. Selain itu, serangan *Cross-Site Scripting (XSS)* juga dapat digunakan untuk mengambil sesi (*session*) dari pengguna.

Untuk menjaga dan meningkatkan keamanan dari sebuah aplikasi berbasis *website*, maka perlu dilakukan pengujian terhadap aplikasi tersebut secara berkala. Hal tersebut

penting dilakukan karena tanpa pengujian secara berkala, maka tidak ada jaminan bahwa aplikasi tersebut akan aman dalam waktu jangka panjang dan terhindar dari serangan *hacker* maupun *cracker*. Ditambah perkembangan teknologi yang begitu pesat diiringi dengan pembaruan terhadap jenis-jenis serangan.

(Kiezun, Guo, dkk., 2009) melakukan penelitian dengan membuat aplikasi bernama *Ardilla*. Aplikasi ini dapat digunakan untuk melakukan pengujian terhadap sebuah *website* dengan menggunakan jenis serangan *SQL Injection* dan *Cross-Site Scripting (XSS)*. Pada uji coba yang dilakukan, *Ardilla* berhasil menemukan 68 celah keamanan pada 5 *website*. Setiap *website* memiliki kerentanan yang berbeda-beda. Namun, masih terdapat beberapa kekurangan pada penelitian ini, salah satunya adalah aplikasi ini hanya menggunakan 2 jenis serangan, yaitu *SQL Injection* dan *Cross-Site Scripting (XSS)*.

Selain itu, (Siagian, Akbar, dkk., 2011) juga telah melakukan penelitian mengenai pengujian yang dilakukan terhadap *website* Universitas Binadarma. Dari hasil penelitian yang dilakukan, *website* Universitas Binadarma memiliki celah keamanan terhadap tingkat akses kontrol. Celah tersebut memungkinkan penyerang untuk mengunggah *file backdoor* dan menggunakan *file backdoor* tersebut untuk mengakses dan mengubah data-data yang berada didalam *server* tersebut. Namun, proses pengujian yang dilakukan terhadap *website* Universitas Binadarma masih bersifat manual. Manual yang dimaksud disini adalah peneliti masih mengetikkan kode sintaks secara manual dalam proses pengujiannya. Hal ini tentunya kurang efektif untuk dilakukan.

Penelitian ini dilakukan untuk mengatasi kekurangan pada aplikasi *Ardilla* dan penelitian yang dilakukan oleh (Siagian, Akbar, dkk., 2011), yakni dengan membuat sebuah aplikasi yang dapat digunakan untuk melakukan pengujian terhadap aplikasi berbasis *website*. Aplikasi tersebut berisi jenis serangan yang umumnya dilakukan oleh *hacker* ataupun *cracker* untuk mencari celah keamanan dari sebuah *website*, seperti *SQL Injection*, *Phising*, dan *Cross-Site Scripting (XSS)*. Hal ini diharapkan mampu membantu administrator untuk melakukan pengujian secara berkala dan membuat pengujian tersebut menjadi lebih efektif dan efisien.

1.2 Rumusan Masalah

Berdasarkan latar belakang diatas, perumusan masalah pada penelitian ini adalah bagaimana membuat sebuah aplikasi pengujian terhadap sebuah aplikasi berbasis *web* sehingga proses pengujian menjadi lebih efektif dan lebih efisien.

1.3 Batasan Masalah

Agar pengerjaan lebih terarah, dalam penelitian ini terdapat beberapa batasan masalah, diantaranya :

- a. Data masukan yang diterima oleh aplikasi berupa alamat *URL (Uniform Resource Locator)* atau *ip address*.
- b. Proses yang terdapat didalam aplikasi diantaranya jenis serangan *SQL Injection*, *Cross-Site Scripting (XSS)*, dan *Phising*.
- c. Hasil keluaran yang akan dihasilkan berupa informasi apakah target rentan terhadap jenis serangan yang terdapat didalam aplikasi atau tidak.
- d. Target yang dijadikan bahan untuk uji coba telah ditentukan.

1.4 Tujuan Penelitian

Adapun tujuan yang ingin dicapai pada penelitian ini yaitu membuat sebuah aplikasi yang dapat melakukan pengujian aplikasi berbasis *web* menjadi lebih efektif dan lebih efisien dengan menggunakan beberapa jenis serangan.

1.5 Manfaat Penelitian

Manfaat dari penelitian ini antara lain :

- a. Bagi peneliti :
 1. Dapat mengimplementasikan ilmu pengetahuan yang telah diperoleh selama kuliah.
 2. Mendapatkan pembelajaran baru mengenai bahasa pemrograman *Python*.
 3. Dapat menambah pengetahuan dan pemahaman mengenai keamanan sistem terutama keamanan aplikasi berbasis *website*.
- b. Bagi Masyarakat :
 1. Membantu administrator dalam melakukan pengujian sistem secara berkala.
 2. Tidak perlu mengeluarkan biaya lebih untuk menyewa penyedia jasa keamanan sistem.

1.6 Metodologi Penelitian

Adapun metodologi penelitian yang digunakan dalam pembuatan aplikasi pengujian celah keamanan ini, yaitu :

- a. Persiapan. Yaitu diawali dengan penyusunan proposal, kemudian mencari dan melakukan instalasi alat-alat yang diperlukan untuk membangun aplikasi.

- b. Studi literatur. Mengumpulkan data sesuai topik dengan mencari bahan dari internet, artikel, jurnal ilmiah, maupun tugas akhir.
- c. Perancangan aplikasi. Setelah mendapatkan data dari bahan yang didapatkan, tahap selanjutnya adalah melakukan perancangan aplikasi yang terdiri dari analisa kebutuhan aplikasi dan perancangan alur aplikasi.
- d. Pembuatan aplikasi. Dalam proses ini dilakukan pembuatan antarmuka aplikasi, dan penyusunan kode program.
- e. Pengujian aplikasi. Dalam tahap ini, aplikasi yang telah dibuat diuji dengan melakukan pengujian terhadap target yang telah ditentukan. Jika terjadi kesalahan, dilakukan pemeriksaan kembali mulai dari perancangan aplikasi hingga mendapatkan hasil yang diinginkan.

1.7 Sistematika Penulisan

Sistematika penulisan penelitian ini disusun untuk memberikan gambaran umum tentang penelitian yang dijalankan. Secara garis besar, sistematika penelitian ini adalah sebagai berikut :

BAB I PENDAHULUAN

Bab ini memuat latar belakang permasalahan, rumusan masalah, menentukan batasan masalah yang akan dibahas, menjelaskan tujuan dan manfaat dari penelitian ini, asumsi metodologi penelitian, dan penjelasan mengenai sistematika penulisan.

BAB II LANDASAN TEORI

Pada bagian ini berisi berbagai teori yang digunakan sebagai landasan untuk menyelesaikan permasalahan yang diangkat pada penelitian ini. Bahasan dalam bagian ini, diantaranya pembahasan teori dasar terkait *SQL Injection*, *Phising*, dan *Cross-Site Scripting (XSS)*.

BAB III METODOLOGI

Bab ini berisi tentang objek dan jenis penelitian, teknik pengumpulan data, analisis terhadap kebutuhan aplikasi yang akan dibangun, dan perancangan alur aplikasi.

BAB IV HASIL DAN PEMBAHASAN

Bab ini membahas tentang pembahasan berupa uraian, penjelasan, dan hasil pembuatan aplikasi yang telah dibuat sebelumnya. Pengujian dilakukan untuk memastikan bahwa hasil akhir yang dibuat sesuai dengan kebutuhan pengguna.

BAB V KESIMPULAN DAN SARAN

Pada bagian ini berisi kesimpulan dan saran yang menjelaskan tujuan penelitian dapat tercapai serta menjelaskan kelebihan dan kekurangan yang terdapat pada aplikasi yang telah dibuat. Sedangkan saran berisi hal-hal yang dapat dikembangkan lagi mengenai kekurangan dari aplikasi yang telah dibuat.

BAB II

LANDASAN TEORI

2.1 Penelitian Terdahulu

(Kiezun, Guo, dkk., 2009) melakukan penelitian dengan membuat aplikasi bernama *Ardilla*. Aplikasi ini dapat digunakan untuk melakukan pengujian terhadap sebuah *website* dengan menggunakan jenis serangan *SQL Injection* dan *Cross-Site Scripting (XSS)*. Aplikasi ini akan menghasilkan vektor serangan yang konkret dan ditulis dalam bentuk bahasa pemrograman *PHP (Hypertext Preprocessor)*. Untuk menggunakan aplikasi *Ardilla* ini, pengguna harus menentukan jenis serangan yang akan digunakan antara serangan *SQL Injection* atau serangan *Cross-Site Scripting (XSS)*. Nantinya, hasil keluaran dari program *Ardilla* adalah vektor serangan. Pada uji coba yang dilakukan, *Ardilla* berhasil menemukan 68 celah keamanan pada 5 program. Masing-masing program memiliki kerentanan yang berbeda-beda. Namun, masih terdapat beberapa kekurangan pada penelitian ini salah satunya adalah program ini hanya menggunakan 2 jenis serangan, yaitu *SQL Injection* dan *Cross-Site Scripting (XSS)*. Sampai saat ini aplikasi *Ardilla* belum terpublikasi untuk publik karena aplikasi tersebut masih dalam tahap pengembangan.

Berbeda dengan aplikasi *Ardilla* yang menggunakan 2 jenis serangan, (Elu, 2013) melakukan sebuah penelitian dengan membuat sebuah aplikasi pendeteksi celah *SQL Injection* untuk keamanan *website*. Untuk menjalankan aplikasi tersebut, pertama-tama pengguna harus memasukkan alamat *website* yang ingin diuji. Kemudian, pengguna memilih jenis pengujian yang akan digunakan. Aplikasi ini menyediakan 8 jenis pengujian, diantaranya uji komentar baris, uji komentar sebaris, uji perintah bertumpuk, uji kalimat jika, uji bilangan bulat, uji untaian, uji penggabungan hasil *query*, dan uji kesalahan. Setelah itu, program tinggal dijalankan dengan menekan tombol mulai. Pada saat proses pengujian dijalankan, aplikasi ini nantinya akan menampilkan semua halaman-halaman yang ada di dalam *website* tersebut dengan menggunakan teknik *crawling web*. Selain itu, aplikasi ini juga memungkinkan pengguna untuk melihat kode *HTML (Hypertext Markup Language)* atau *script website* dari halaman yang sedang dianalisa. Hasil akhir yang akan ditampilkan oleh aplikasi ini adalah menampilkan halaman apa saja yang rentan terhadap *SQL Injection*. Aplikasi pendeteksi celah *SQL Injection* ini telah diuji secara sistem dan telah diuji oleh beberapa pengguna yang ditentukan oleh peneliti. Beberapa penelitian terdahulu lainnya dapat dilihat pada Tabel 2.1.

Tabel 2.1 Penelitian Terdahulu

No	Penulis	Teknik	Target	Tools
1.	(Elu, 2013)	Uji komentar baris, uji komentar sebaris, uji perintah bertumpu, uji kalimat jika, uji bilangan bulat, uji untaian, uji penggabungan hasil <i>query</i> , dan uji kesalahan	-	-
2.	(Dharmawan, Eka Adhitya , Erni Yudaningtyas, 2013)	Algoritma <i>Rijndael</i>	Halaman <i>Login</i> pada sebuah <i>website</i> .	-
3.	(Budiman & Jati, 2017)	<i>SQL Injection</i>	<i>Server Windows 7</i> yang dijalankan menggunakan <i>VMWare</i>	<i>SQLMap</i>
4.	(Siregar, 2013)	-	-	-
5.	(Tolle, Kurniawan, & Zakaria, 2008)	<i>Cross-Site Scripting</i>	<i>AuraCMS</i>	-
6.	(Rachmawati, 2014)	<i>Phising</i>	Halaman <i>login Facebook</i> (https://www.facebook.com)	<i>MS-Front Page, XAMPP, dan Notepad</i>
7.	(Siagian & Akbar, 2011)	<i>SQL Injection, Cross-Site Scripting, DOS Attack, Missing Function Level Access Control, dan Insecure Direct Reference Object</i>	Web Server Binadarma	<i>Acunetix, Nikto, OpenVas, dan Retina Web Scanner</i>
8.	(Saputra Ahad & Akbar, 2015)	-	Website PDAM Tirta Musi Palembang	<i>Acunetix, Netsparker, dan Nessus</i>
9.	(Yadi & Herdiansyah, 2012)	-	<i>Digital Library Universitas Binadarma</i>	<i>Acunetix Vulnerability Scanner dan Shadow Database Scanner</i>

2.2 Open Web Application Security Project (OWASP)

OWASP (*Open Web Application Security Project*) merupakan komunitas terbuka diseluruh dunia yang berfokus pada peningkatan keamanan aplikasi perangkat lunak (Meucci & Muller, 2014). Hingga saat ini, panduan pengujian OWASP telah mencapai versi 4.0. Panduan pengujian OWASP versi 1.0 pertama kali dipublikasikan pada bulan Desember tahun 2004. Pada tanggal 25 Desember 2006, panduan pengujian OWASP versi 2.0 pun dirilis. Selang 2 tahun tepatnya pada tanggal 15 September 2008, OWASP merilis panduan pengujian OWASP versi 3.0. Dan OWASP kembali merilis panduan pengujian OWASP versi 4.0 pada tahun 2014.

Pada tahun 2003, untuk pertama kalinya Yayasan OWASP merilis 10 daftar resiko keamanan aplikasi website yang paling kritis. Daftar tersebut disebut *OWASP TOP 10*. Pada tahun 2004, untuk kedua kalinya yayasan OWASP merilis *OWASP TOP 10 – 2004*. Seiring dengan perkembangan teknologi dan meningkatnya kejahatan siber, Yayasan OWASP kembali merilis *OWASP TOP 10* pada tahun 2007, 2010, 2013, dan terakhir pada tahun 2017. *OWASP TOP 10* tidak hanya berisi daftar resiko keamanan aplikasi *website* yang paling kritis, namun juga berisi tentang penjelasan daftar resiko tersebut, cara mencegah celah keamanan tersebut, dan contoh skenario dari serangan tersebut.

Setiap *OWASP TOP 10* diterbitkan, terdapat beberapa perubahan pada daftar resiko keamanan, baik seperti urutan daftar resiko yang berubah hingga munculnya jenis-jenis serangan baru yang lebih berbahaya dibandingkan jenis serangan sebelumnya. Sebagai contoh, perbandingan daftar *OWASP TOP 10 – 2013* dengan daftar *OWASP TOP 10 – 2017* dapat dilihat pada Tabel 2.2.

Tabel 2.2 Tabel Perbandingan *OWASP TOP 10-2013* dengan *OWASP TOP 10-2017*

Sumber : (Owasp, 2017)

OWASP TOP 10 – 2013	OWASP TOP 10 – 2017
A1 – Injeksi	A1 – Injeksi
A2 – Otentikasi dan manajemen sesi yang buruk	A2 – Otentikasi yang buruk
A3 – <i>Cross-Site Scripting (XSS)</i>	A3 – Data sensitif yang terekspos
A4 – Referensi obyek langsung yang tidak aman	A4 – <i>XML External Entities (XXE)</i>
A5 – Kesalahan konfigurasi keamanan	A5 – Akses kontrol yang buruk
A6 – Data sensitif yang terekspos	A6 – Kesalahan konfigurasi keamanan
A7 – Kehilangan fungsi kontrol tingkatan akses	A7 – <i>Cross-Site Scripting (XSS)</i>
A8 – <i>Cross-Site Request Forgery (CSRF)</i>	A8 – Deserialisasi yang tidak aman
A9 – Menggunakan komponen rentan yang diketahui	A9 – Menggunakan komponen rentan yang diketahui
A10 – Redireksi dan <i>Forward</i> yang tidak tervalidasi	A10 – Pencatatan dan pemantauan yang tidak cukup

2.3 Pengertian Vulnerability

Vulnerability adalah suatu titik lemah atau kelemahan dalam sebuah prosedur keamanan kontrol administratif, kontrol internet, dan lain-lain sebagai yang dapat dieksploitasi melalui suatu trik untuk memperoleh akses yang tidak sah terhadap informasi atau untuk mengganggu proses secara kritis (Elu, 2013). Dalam sebuah sistem, *vulnerability* dapat dikatakan sebagai sebuah kelemahan yang dapat dimanfaatkan oleh pihak tidak

bertanggung jawab untuk melanggar kebijakan keamanan sistem. Setiap *website* berpotensi memiliki *vulnerability*, maka dari itu tidak ada jaminan bahwa suatu *website* benar-benar aman dari segala celah.

Penguji (*tester*) harus melakukan pengujian (*penetration testing*) terlebih dahulu untuk menemukan *vulnerability* atau kelemahan dalam suatu *website*. Menurut (Pangalila & Andjarwirawan, 2015), terdapat 2 metode untuk melakukan pengujian, yakni *white box testing* dan *black box testing*. *White box testing* adalah sebuah metode pengujian dimana penguji mengetahui semua informasi yang diperlukan dari sebuah *website* untuk melakukan pengujian. Sedangkan *black box testing* adalah sebuah metode pengujian dimana penguji tidak mengetahui sama sekali tentang informasi dan infrastruktur dari sebuah target. Dengan demikian pada *black box testing*, penguji harus menggali informasi yang dibutuhkan dari awal kemudian melakukan analisis serta menentukan jenis serangan yang akan digunakan dalam proses pengujian.

Pengujian terhadap sebuah *website* dibagi kedalam 2 fase, yakni fase pasif (*Passive mode*) dan fase aktif (*Active mode*). Pada fase pasif, penguji mencoba untuk memahami tentang logika sistem. Selain itu, penguji harus dapat memahami semua jalur akses dari sistem seperti *cookies* dan parameter. Dalam fase ini, penguji dapat menggunakan *tools* untuk menggali dan mengumpulkan informasi. Fase selanjutnya adalah fase aktif, dimana pada fase ini penguji mulai melakukan pengujian menggunakan metodologi yang telah dibagi menjadi 11 sub-kategori (Meucci & Muller, 2014), diantaranya :

- a. Pengumpulan Informasi.
- b. Pengujian Konfigurasi dan Manajemen Penerapan.
- c. Pengujian Manajemen Identitas.
- d. Pengujian Otentikasi.
- e. Pengujian Otorisasi.
- f. Pengujian Manajemen Sesi.
- g. Pengujian Validasi Masukan.
- h. Penanganan Kesalahan.
- i. Kriptografi.
- j. Pengujian Logika Bisnis.
- k. Pengujian Sisi Klien.

2.4 Jenis-jenis Serangan

2.4.1 SQL Injection

SQL (Structured Query Language) Injection merupakan suatu teknik peretasan yang memungkinkan penyerang mendapatkan akses yang tidak sah kedalam *database* kemudian menyerang atau mengubah data-data yang berada didalam *database* (Kiezun, Guo, dkk., 2009). Berdasarkan data dari Akamai Q2 pada tahun 2016 (Fernandez, Arteaga, dkk., 2016), *SQL Injection* merupakan ancaman yang sering terjadi pada aplikasi *web server* setelah *Local File Inclusion (LFI)* dengan presentasi sebesar 44.11%, diikuti dengan *Cross-Site Scripting (XSS)* dengan presentasi sebesar 5.91% dan *Remote File Inclusion (RFI)* 2.27%. Teknik *SQL Injection* ini digunakan dengan cara memasukkan perintah-perintah *SQL* melalui alamat *URL (Uniform Resource Locator)* atau melalui kolom masukan yang nantinya akan dieksekusi oleh *server* ketika meminta data ke dalam *database*.

Dalam teknik *SQL Injection*, diperlukan karakter-karakter khusus untuk memanipulasi perintah *SQL*. Karakter-karakter tersebut yakni *Double Minus (--)* dan *Union*. *Double Minus (--)* merupakan tanda akhir suatu perintah *SQL*, sehingga perintah yang berada dibelakang *double minus* tidak akan terbaca dan tereksekusi oleh *MySQL* (Irwan & Arif, 2014). *Union* merupakan perintah untuk menggabungkan dua atau lebih perintah *SQL*. *Union* sendiri digunakan untuk memberikan perintah *error* dalam *query*. Salah satu penyebab utama suatu *website* rentan terhadap jenis serangan ini adalah minimnya penyaringan (*filter*) terhadap variabel.

Beberapa hal yang dapat dilakukan untuk mengetahui apakah sebuah *website* rentan terhadap *SQL Injection* atau tidak, seperti memeriksa kode apakah *website* telah menggunakan *compiler* dengan aman atau tidak. Selanjutnya dengan melakukan verifikasi bahwa semua penggunaan *compiler* secara tegas memisahkan data yang tidak dapat dipercaya dari perintah atau *query* (Owasp, 2017). Dan memastikan *website* memiliki *error handling* yang baik, karena *error handling* yang buruk akan membuat celah untuk serangan *injection* semakin mudah ditemukan.

2.4.2 Phising

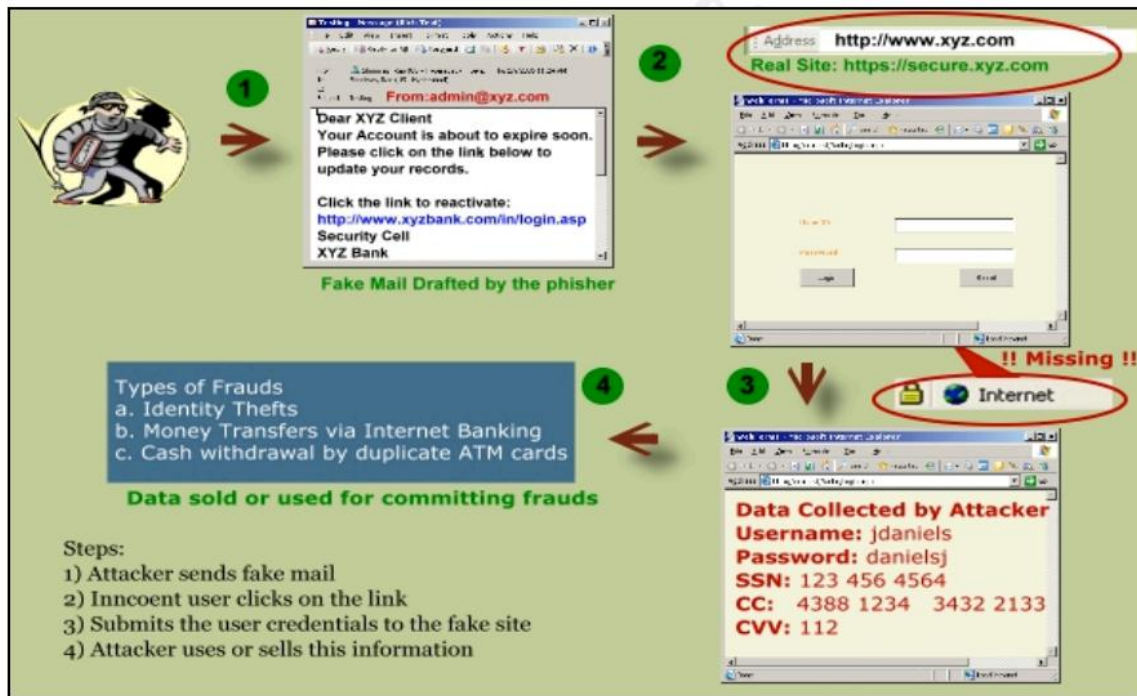
Phising merupakan salah satu jenis serangan dalam bentuk penipuan, dimana penyerang akan mengirimkan sebuah tautan yang akan mengarahkan korban ke sebuah halaman *website*. Halaman tersebut memiliki tampilan seperti tampilan *website* yang dapat dipercaya, namun sesungguhnya halaman tersebut merupakan jebakan untuk korban. Tujuan

dari *phishing* ini sendiri adalah untuk mendapatkan informasi sensitif seperti *e-mail*, *username*, *password*, dan data penting lainnya dari korban.

A. Cara Kerja *Phishing*

Informasi yang dibutuhkan oleh penyerang adalah informasi sensitif seperti *e-mail*, *username*, *password*, hingga nomor kartu kredit. Oleh karena itu, penyerang akan menjebak korban dengan mengirimkan *e-mail* atau mengirimkan tautan kepada korban melalui pesan instan. Cara kerja dari *phishing* sendiri yaitu pertama-tama penyerang membuat halaman *website* yang menyerupai *website* yang sesungguhnya. Selanjutnya untuk mengelabui korban, penyerang memanipulasi alamat *website* yang telah dibuat agar mirip dengan alamat *website* yang sesungguhnya. Misalnya untuk mendapatkan *e-mail* dan *password* akun *facebook* seseorang, penyerang akan memanipulasi alamat *website* yang telah dibuat dengan nama “*facebok.com*”, “*favebook.com*”, atau “*focebook.com*”. Setelah memanipulasi alamat *website* yang telah dibuat, penyerang akan mengirimkan alamat *website* tersebut kepada korban melalui *e-mail* ataupun melalui pesan instan.

Selanjutnya, menunggu kecerobohan dari korban. Ketika korban membuka alamat *website* yang diberikan oleh penyerang, maka korban akan masuk ke halaman *website* yang telah dibuat oleh penyerang tadi. Kemudian, ketika korban memasukkan data-data yang dibutuhkan dalam *website* tersebut, maka data-data tersebut secara otomatis akan terkirim ke komputer penyerang. Dari data-data tersebut, penyerang dapat mengakses situs yang digunakan oleh korban tanpa harus meretas situs tersebut. Contoh skenario *phishing* dapat dilihat pada Gambar 2.1. Pada contoh skenario tersebut, pertama-tama penyerang mengirimkan *e-mail* palsu yang berisi tautan alamat *phishing* kepada korban. Ketika korban membuka tautan yang diberikan oleh penyerang, maka korban akan diarahkan ke *website* yang telah dibuat oleh penyerang sebelumnya. *Website* tersebut memiliki tampilan yang menyerupai *website* resmi pada umumnya. Hal tersebut dilakukan untuk membuat korban percaya bahwa *website* yang dibuat oleh penyerang adalah asli. Ketika korban memasukkan data-data yang dibutuhkan, seluruh data tersebut nantinya akan terkirim ke dalam komputer penyerang. Setelah penyerang mendapatkan data-data yang dikirimkan oleh korban, penyerang dapat mengakses akun tersebut atau menjualnya.



Gambar 2.1 Contoh Skenario Serangan *Phishing*

Sumber : (Srivastava, 2007)

B. Teknik *Phishing*

Untuk memancing dan menjebak korban, penyerang dapat melakukan beberapa teknik *Phishing* (Rachmawati, 2014), diantaranya :

1. E-mail Spoofing

Teknik ini merupakan teknik yang dilakukan dengan cara mengirimkan *e-mail* berisi tautan atau *malware* kepada target dengan mengatasnamakan institusi atau organisasi resmi. Biasanya *e-mail* tersebut berisi permintaan nomor kartu kredit, *password* akun, atau nomor induk kependudukan.

2. Pesan Instan (*Chatting*)

Teknik ini digunakan oleh penyerang dengan cara mengirimkan tautan berisi *malware* atau alamat *website* palsu kepada target melalui pesan instan, seperti *facebook*, *whatsapp*, *line*, dan pesan instan lainnya. Untuk melakukan teknik ini, penyerang harus memiliki kemampuan komunikasi yang baik agar korban dapat masuk kedalam jebakan yang telah dibuat oleh penyerang.

3. Manipulasi Tautan

Teknik ini adalah memanipulasi tautan *website* yang akan digunakan untuk menjebak korban, lalu mengirimkannya ke sebuah blog atau *website*. Bila pengguna menekan (*click*)

tautan tersebut, maka pengguna tersebut secara otomatis akan diarahkan ke *website* yang telah dibuat oleh penyerang.

4. Malware Phising

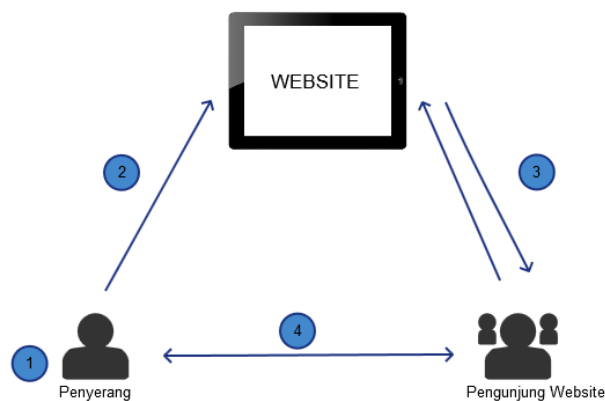
Teknik ini melibatkan *malware* yang nantinya akan berjalan pada komputer target. *Malware* tersebut akan disisipkan oleh penyerang kedalam sebuah *e-mail*, lalu penyerang mengirimkannya kepada target. Ketika target membuka *e-mail* tersebut, secara otomatis *malware* tersebut akan terunduh ke dalam komputer target dan akan langsung berjalan.

2.4.3 Cross-Site Scripting (XSS)

Cross-Site Scripting (XSS) merupakan sebuah jenis serangan yang memanfaatkan kelemahan *server* dalam melakukan validasi terhadap masukan yang diberikan oleh pengguna. XSS memungkinkan penyerang mengeksekusi *script-script* didalam *browser* korban, sehingga dapat mengubah tampilan *website* atau mengarahkan pengguna ke situs-situs jahat (Owasp, 2013). Selain itu, jenis serangan ini juga memungkinkan penyerang untuk mencuri *cookies* pengguna lain. Ketika penyerang telah mendapatkan *cookies* dari pengguna lain, maka penyerang dapat memuat nilai sesi tersebut dan menggunakannya untuk masuk ke *website* yang digunakan oleh *cookies* tersebut. Banyak penyedia layanan yang tidak mengakui dan menganggap bahwa celah dari serangan *Cross-Site Scripting (XSS)* tidak begitu berbahaya, sehingga banyak dari mereka yang tidak melakukan langkah pencegahan terhadap serangan ini (Tolle, Kurniawan, dkk., 2008).

Menurut (Irwan & Arif, 2014), *Cross-Site Scripting (XSS)* memiliki 2 jenis, yaitu *Reflected XSS* dan *Stored XSS*. *Reflected XSS* adalah jenis serangan XSS yang hanya berjalan pada halaman klien (*client page*). Jenis serangan ini biasanya digunakan oleh penyerang untuk mencari celah keamanan sebuah *website* dari sisi klien. Berbeda dengan *Reflected XSS*, *Stored XSS* merupakan jenis serangan XSS yang dapat berjalan disisi klien dan *server*. Hal tersebut dikarenakan *script* yang dimasukkan oleh penyerang akan tersimpan didalam *database*. Jenis serangan ini lebih berbahaya dibandingkan *Reflected XSS*, karena *script* yang telah dimasukkan oleh penyerang akan terus berjalan setiap pengguna memanggil halaman yang terinjeksi. Untuk memanfaatkan *Stored XSS* lebih lanjut, penyerang dapat menggabungkan jenis serangan ini dengan serangan *phising*. Contoh penggunaan jenis serangan *Stored XSS* yakni dengan mengisi kolom komentar atau kolom kritik dan saran pada sebuah *website* atau *blog* dengan *script javascript* yang berisi tautan ke *website* jebakan yang telah dibuat. Ketika pengguna lain membuka halaman yang telah terinjeksi, maka secara langsung pengguna akan diarahkan ke *website* jebakan.

Cara kerja dari XSS sendiri dapat dilihat pada Gambar 2.2. Pertama-tama penyerang mencari *website* yang rentan terhadap serangan XSS. Selanjutnya, penyerang akan menyisipkan *script* untuk mencuri sesi pengguna didalam *website* tersebut. Ketika ada pengguna yang mengunjungi *website* yang telah diinjeksi oleh penyerang, maka *script* yang telah disisipkan akan berjalan. Setelah *script* tersebut berhasil mencuri sesi pengguna, maka sesi tersebut akan dikirim ke penyerang. Selanjutnya, penyerang dapat mengakses akun pengguna (korban) dengan sesi yang telah didapatkan sebelumnya.



Gambar 2.2 Kerja Serangan *Cross-Site Scripting (XSS)*

Ada beberapa cara untuk mencegah jenis serangan ini, yang pertama adalah menyaring dan menguraikan semua data yang dimasukkan oleh pengguna ke dalam sistem berdasarkan konteks *HTML*. Cara kedua adalah menggunakan metode *POST* untuk mengirimkan data masukan dari pengguna ke dalam sistem. Metode ini menjamin kode tidak dapat diinjeksi melalui tautan (*link*) yang telah didesain oleh penyerang (Tolle, Kurniawan, dkk., 2008). Dan cara terakhir adalah dengan melakukan validasi panjang karakter, format karakter, dan karakter terhadap masukan yang dilakukan oleh pengguna. Namun, perlu diketahui bahwa cara tersebut bukanlah cara pencegahan menyeluruh karena banyak aplikasi yang membutuhkan karakter khusus dalam masukan (*input*) mereka (Owasp, 2017).

2.5 Peninjauan Aplikasi Sejenis

Aplikasi Pengujian Celah Keamanan pada Aplikasi Berbasis *Web* ini memiliki 3 jenis serangan, yakni *SQL Injection*, *Cross-Site Scripting (XSS)*, dan *Phising*. Pada dasarnya telah banyak aplikasi-aplikasi yang memiliki fungsi untuk menguji sebuah sistem dengan jenis serangan yang juga dimiliki oleh Aplikasi Pengujian Celah Keamanan pada Aplikasi Berbasis *Web*, seperti *sqlmap* dan *SET (Social Engineer Toolkit)*. *Sqlmap* merupakan sebuah alat

pengujian terhadap aplikasi berbasis *website* yang bersifat terbuka (*open source*). *Sqlmap* memiliki fungsi untuk mendeteksi dan mengeksploitasi kelemahan sebuah sistem terhadap serangan *SQL Injection*. Kelebihan *sqlmap* dibanding dengan Aplikasi Pengujian Celah Keamanan pada Aplikasi Berbasis Web adalah *sqlmap* mampu mengambil alih *server database* sebuah sistem yang rentan terhadap serangan *SQL Injection*. Dengan kelebihan itu, pengguna *sqlmap* memiliki kesempatan untuk mengakses dan mengubah isi dari *database* yang berada pada *server* dari sebuah sistem. Namun, kekurangan yang dimiliki *sqlmap* adalah pengguna harus memasukkan alamat *URL* yang rinci. Rinci disini adalah alamat *URL* yang memiliki kata “*id=nomor_id*” dibagian akhirnya. Dan kekurangan itulah yang menjadi kelebihan dari Aplikasi Pengujian Celah Keamanan pada Aplikasi Berbasis Web, karena aplikasi tersebut mampu menerapkan teknik *web crawling*, sehingga pengguna cukup memasukkan alamat *URL* umumnya saja.

SET (Social Engineer Toolkit) merupakan sebuah alat pengujian yang dapat melakukan duplikasi terhadap *website* yang diinginkan pengguna. Pengguna cukup memasukkan *ip address* tujuan untuk penyimpanan hasil duplikasi dan memasukkan alamat *URL website* yang akan diduplikasi. Kekurangan alat pengujian *SET* ini adalah waktu yang dibutuhkan untuk melakukan duplikasi cukup lama. Berbeda dengan Aplikasi Pengujian Celah Keamanan pada Aplikasi Berbasis Web yang tidak membutuhkan waktu yang begitu lama untuk melakukan proses duplikasi. Namun perlu diketahui bahwa Aplikasi Pengujian Celah Keamanan pada Aplikasi Berbasis Web tidak dapat menduplikasi semua *website* karena proses duplikasi pada aplikasi ini hanya dapat berjalan ketika seluruh sumber yang digunakan oleh target tidak berasal dari dalam *server*.

BAB III

METODOLOGI PENELITIAN

3.1 Metode Penelitian

Dalam pembuatan Aplikasi Pengujian Celah Keamanan pada Aplikasi Berbasis Web ini, terdapat tahapan pengumpulan data. Teknik yang digunakan pada tahapan pengumpulan data adalah studi literatur atau biasa disebut dengan studi pustaka. Studi literatur atau studi pustaka merupakan sebuah proses pengumpulan data dan informasi berupa teori-teori yang berhubungan dengan masalah yang diteliti. Studi literatur dilakukan dengan mencari jurnal ilmiah, paper ilmiah, dan tugas akhir di internet. Adapun data-data yang dibutuhkan untuk membangun aplikasi ini, diantaranya teori pengujian celah keamanan pada aplikasi atau sistem berbasis *website*, teori pembuatan aplikasi pengujian celah keamanan menggunakan bahasa pemrograman *python*, dan praktek pembuatan tampilan (*interface*) aplikasi dengan menggunakan modul *PyQT5* pada bahasa pemrograman *python*.

3.2 Analisa Kebutuhan Aplikasi

Pembuatan Aplikasi Pengujian Celah Keamanan pada Aplikasi Berbasis Web ini membutuhkan beberapa aplikasi dan modul yang dapat mendukung perancangan aplikasi serta pembuatan desain aplikasi. Untuk mengetahui aplikasi dan modul yang akan digunakan, maka diperlukan analisa terhadap kebutuhan aplikasi. Metode analisa yang digunakan dalam merancang dan membangun aplikasi ini, diantaranya analisis kebutuhan perangkat lunak, analisis kebutuhan masukan (*input*), analisis kebutuhan proses, dan analisis kebutuhan keluaran (*output*).

3.2.1 Analisis Kebutuhan Perangkat Lunak

Pada proses perancangan dan pembuatan aplikasi terdapat beberapa komponen perangkat lunak yang digunakan. Dalam perangkat lunak sistem, peneliti menggunakan sistem operasi *Windows 7*. Untuk merancang tampilan (*interface*) dari aplikasi, digunakan *tools Pencil* karena *tools* tersebut telah memiliki *library GUI (Graphic User Interface)* dan mudah untuk digunakan. Dalam pembuatan aplikasi, digunakan bahasa pemrograman *python* versi 3.6 dan modul *PyQt5* karena hasil dari modul *PyQt5* lebih baik dan lebih jernih dibandingkan modul *Tkinter*. Untuk *text editor*, digunakan *JetBrain PyCharm*. Dan untuk melakukan pengujian secara manual dibutuhkan browser *Mozilla Firefox*.

Sistem operasi	: Windows 7 (64 bit).
Tools	: Pencil versi 2.0.5.
Text Editor	: JetBrains PyCharm Community Edition 2017 (64 bit)
Browser	: Mozilla Firefox versi 57.0 (64 bit)
Bahasa pemrograman	: Python versi 3.6.0
Modul	: PyQt5

3.2.2 Analisis Kebutuhan Masukan

Pada proses analisis kebutuhan masukan, pengguna akan diminta untuk memasukkan alamat *website* target berupa *URL* atau *ip address website* tersebut. Setelah memasukkan alamat *website* target, selanjutnya aplikasi akan melakukan pengujian terhadap target terserbut dengan 3 jenis serangan. 3 Jenis serangan tersebut adalah *SQL Injection*, *Cross-Site Scripting (XSS)*, dan *Phising*.

3.2.3 Analisis Kebutuhan Proses

Proses yang dibutuhkan oleh aplikasi pengujian celah keamanan ini berupa analisa terhadap *website* yang akan diuji apakah rentan terhadap jenis serangan yang dimiliki oleh aplikasi atau tidak. Pertama-tama aplikasi akan melakukan *crawling*. Tujuan dari *crawling* adalah mencari tautan halaman-halaman yang terdapat pada halaman awal target. Dari tautan-tautan yang telah dikumpulkan, aplikasi akan memilah dan memilih tautan yang memiliki kata “*id=*” didalamnya. Tautan tersebutlah yang nantinya akan digunakan dalam pengujian serangan *SQL Injection* dan *Cross-Site Scripting (XSS)*. Untuk serangan *SQL Injection*, aplikasi ini akan menyisipkan tanda petik satu (‘) pada akhir tautan yang telah dipilih oleh aplikasi. Setelah itu, aplikasi akan mengunjungi halaman tautan yang telah disisipkan tanda petik satu dan mendeteksi apakah terdapat pesan *error* berupa kalimat yang mengandung kata “*SQL syntax*” atau tidak pada halaman tautan tersebut.

Untuk serangan *Cross-Site Scripting (XSS)*, aplikasi pengujian celah keamanan ini akan menyisipkan *script* diakhir tautan yang telah ditentukan oleh aplikasi. Setelah itu, aplikasi akan mengunjungi halaman tautan tersebut dan mendeteksi apakah *script* yang disisipkan berjalan atau tidak pada halaman tersebut. Dan untuk serangan *Phising*, aplikasi ini pertama-tama akan melakukan *crawling* dan mencari tautan yang memiliki kata “*login*” didalamnya. Jika terdapat tautan yang memiliki kata “*login*”, aplikasi akan mengunjungi tautan tersebut. Selanjutnya aplikasi akan memeriksa apakah sumber (*source*) gambar yang digunakan pada halaman tautan tersebut berasal dari *file* dalam *server* atau berasal dari alamat lain. Jika

sumber gambar yang digunakan pada halaman tautan tersebut berasal dari alamat lain, maka aplikasi melakukan duplikasi terhadap halaman tautan tersebut.

3.2.4 Analisis Kebutuhan Keluaran

Keluaran akhir yang dihasilkan oleh Aplikasi Pengujian Celah Keamanan pada Sistem Informasi berbasis *Web* ini adalah informasi apakah aplikasi yang dijadikan target rentan terhadap jenis serangan yang terdapat pada aplikasi atau aplikasi tersebut tidak rentan. Apabila target rentan terhadap jenis serangan *SQL Injection* atau *Cross-Site Scripting (XSS)*, maka aplikasi akan memberikan keluaran informasi berupa alamat spesifik dari target yang rentan. Sedangkan, jika target rentan terhadap jenis serangan *Phising*, maka aplikasi akan memberikan keluaran informasi berupa alamat asli target.

3.3 Perancangan Aplikasi

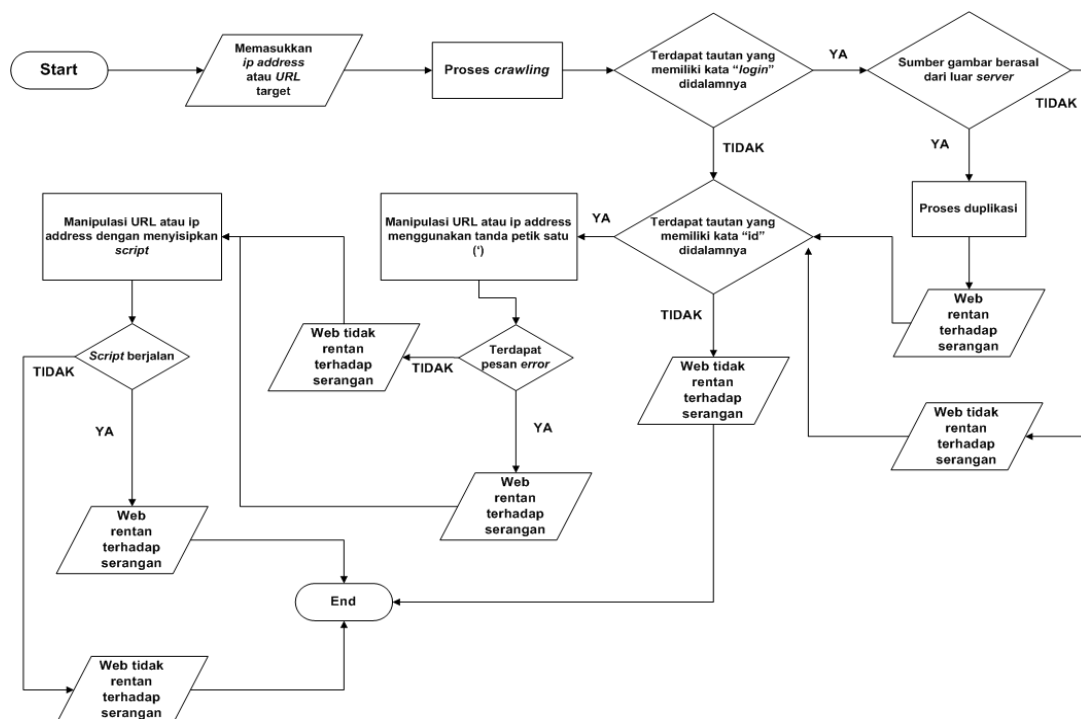
3.3.1 Flowchart

Alur (*flowchart*) Aplikasi Pengujian Celah Keamanan pada Aplikasi Berbasis *Web* dapat dilihat pada Gambar 3.1. Ketika pengguna membuka aplikasi ini, hal pertama yang dilakukan adalah aplikasi meminta pengguna untuk memasukkan alamat target, baik berupa alamat *URL* maupun *ip address*. Selanjutnya aplikasi akan melakukan *crawling* untuk mencari tautan-tautan yang berada di halaman awal target. Jika terdapat tautan yang memiliki kata "*login*" pada halaman awal *website*, maka aplikasi akan mengunjungi halaman tersebut. Selanjutnya, aplikasi akan melakukan pengecekan terhadap sumber gambar yang berada didalam halaman tersebut. Jika sumber gambar pada halaman tersebut berasal dari *file* yang berada didalam *server*, maka aplikasi akan membaca bahwa aplikasi *web* target tidak rentan terhadap serangan *Phising*. Namun, jika sumber gambar pada halaman tersebut berasal dari alamat lain, maka aplikasi akan membaca bahwa aplikasi *web* target rentan terhadap serangan *Phising*. Apabila aplikasi membaca bahwa target rentan terhadap serangan *Phising*, maka aplikasi akan melakukan duplikasi terhadap halaman *login* pada aplikasi *web* tersebut.

Setelah itu, aplikasi kembali melakukan *crawling*. Namun yang berbeda adalah aplikasi akan mencari tautan-tautan pada halaman awal target yang memiliki kata "*id=*". Jika tidak terdapat tautan yang memiliki kata "*id=*", maka aplikasi akan membaca bahwa aplikasi *web* tersebut tidak rentan terhadap serangan *SQL Injection* dan *Cross-Site Scripting (XSS)*. Dan aplikasi akan berhenti melakukan proses pengujian. Sedangkan, apabila terdapat tautan yang memiliki kata "*id=*", maka aplikasi akan menyisipkan tanda petik satu (') diakhir tautan tersebut. Setelah menyisipkan tanda petik satu, aplikasi akan mengunjungi tautan tersebut.

Jika terdapat pesan *error* pada halaman tautan yang dikunjungi, maka aplikasi akan membaca bahwa aplikasi *web* tersebut rentan terhadap serangan *SQL Injection*. Sebaliknya, apabila tidak terdapat pesan *error*, maka aplikasi akan membaca bahwa aplikasi *web* tersebut tidak rentan terhadap serangan *SQL Injection*.

Proses selanjutnya adalah menguji kerentanan aplikasi *web* terhadap serangan *Cross-Site Scripting* dengan cara menyisipkan *script* diakhir tautan yang memiliki kata “*id=*”. Proses ini kurang lebih sama halnya dengan proses pada pengujian *SQL Injection*, namun yang membedakan ada 2 hal, yakni yang disisipkan diakhir tautan bukan tanda petik satu (‘), melainkan *script*. Dan yang diperiksa oleh aplikasi pada proses ini adalah apakah *script* yang telah disisipkan berhasil berjalan atau tidak. Apabila *script* yang disisipkan berhasil berjalan, maka aplikasi akan membaca bahwa aplikasi *web* tersebut rentan terhadap serangan *Cross-Site Scripting*. Sedangkan, apabila *script* yang disisipkan tidak berjalan, maka aplikasi akan membaca bahwa aplikasi *web* tersebut tidak rentan terhadap serangan *Cross-Site Scripting*.



Gambar 3.1 *Flowchart* Aplikasi Celah Keamanan pada Aplikasi Berbasis Web

3.3.2 Pseudocode

Pseudocode adalah notasi yang menyerupai notasi bahasa pemrograman tingkat tinggi (Oktaviani, 2015). *Pseudocode* pada umumnya digunakan karena dapat mempermudah perubahan notasi kedalam bentuk bahasa pemrograman. *Pseudocode* Aplikasi Pengujian

Celah Keamanan pada Aplikasi Berbasis *Website* menggunakan 3 *library* yang memiliki fungsi berbeda-beda. Ketiga *library* tersebut dapat dilihat pada Tabel 3.1.

Tabel 3.1 Tabel *Library*

No	Nama <i>Library</i>	Fungsi
1.	BeautifulSoup4	Untuk mengambil data pada <i>file HTML</i> dan <i>XML</i> , lalu mengurai data tersebut.
2.	Get	Untuk mengambil atau menangkap data, baik berupa data <i>string</i> , <i>tag HTML</i> , maupun <i>atribut HTML</i> .
3.	Urllib	<i>Library</i> yang menyediakan beberapa modul, dimana modul-modul tersebut memiliki fungsi untuk mengerjakan pekerjaan yang berkaitan dengan <i>URL</i> .

Secara garis besar, algoritma Aplikasi Pengujian Celah Keamanan pada Aplikasi Berbasis *Web* dapat dijelaskan dengan *pseudocode* yang terdapat pada Gambar 3.2. Pada *pseudocode* dibawah, terdapat beberapa kamus yang digunakan, diantaranya alamatURL, soup, kondisiPhising, kondisiSQLXSS, kondisiSQL, kondisiXSS, URL, kata, cekSumber, dan copasSource. Variabel alamatURL memiliki tipe data *string* dan digunakan untuk menyimpan alamat target. Selanjutnya, variabel soup memiliki tipe data *string* dan digunakan untuk menyimpan nilai dari hasil pengambilan data yang dilakukan oleh *library BeautifulSoup*. Sedangkan, variabel kata yang memiliki tipe data *string* digunakan untuk menyimpan nilai yang akan dicari seperti kata *login*, *html*, dan lain-lain ketika aplikasi berjalan.

Variabel kondisiPhising, kondisiSQLXSS, kondisiSQL, dan kondisiXSS berfungsi untuk menyimpan nilai dari hasil pencarian yang dilakukan oleh *library BeautifulSoup*. Variabel URL memiliki tipe data *string* dan digunakan untuk menyimpan alamat-alamat yang telah ditangkap oleh *library get*. Selanjutnya, variabel cekSumber digunakan untuk menampung sumber-sumber yang didapatkan oleh *library BeautifulSoup*. Dan variabel copasSource memiliki tipe data *string* dan bertugas untuk menampung kode-kode yang telah disalin oleh *library BeautifulSoup*.

<p><u>Aplikasi</u>: Aplikasi Pengujian Celah Keamanan pada Aplikasi Berbasis <i>Web</i> {Aplikasi yang berfungsi untuk menguji celah keamanan pada aplikasi berbasis web dengan menggunakan jenis serangan <i>SQL Injection</i>, <i>Phising</i>, dan <i>Cross-Site Scripting (XSS)</i>}</p>
<p>Kamus :</p> <p>alamatURL : string</p> <p>soup : string</p> <p>kondisiPhising : string</p>

```

kondisiSQLXSS : string
kondisiSQL : string
kondisiXSS : string
URL : string
kata : string
cekSumber : string
copasSource : string
BeautifulSoup(response, 'html5lib') : library
get(value):library
urllib.request.open(value) : library

Deskripsi Algoritma :
input(alamatURL)
BeautifulSoup(response, 'html5lib') ← soup
soup.find(kata : 'login') ← kondisiPhising
if (kondisiPhising) then
    soup.find(kata : 'src') ← cekSumber
    if (cekSumber == "http") then
        soup.find(kata : 'html') ← copasSource
        write (copasSource) in 'index.html'
        output("Website rentan terhadap serangan Phising")
    else
        output("Website tidak rentan terhadap serangan Phising")
else
    output("Website tidak rentan terhadap serangan Phising")

soup.find(kata : 'id=') ← kondisiSQLXSS
if (kondisiSQLXSS) then
    get('href') ← URL
    urllib.request.open(URL + '%27')
    soup.find(kata : 'SQL syntax;') ← kondisiSQL
    if(kondisiSQL) then
        output("Website rentan terhadap serangan SQL Injection")
    else
        output("Website tidak rentan terhadap serangan SQL Injection")
    urllib.request.open(URL + '<script>alert(222)</script>')
    soup.find(kata : 'alert') ← kondisiXSS
    if(kondisiXSS) then
        output("Website rentan terhadap serangan XSS")
    else
        output("Website tidak rentan terhadap serangan XSS")
else
    output("Website tidak rentan")

```

Gambar 3.2 Pseudocode Aplikasi Pengujian Celah Keamanan

Untuk deskripsi algoritmanya, pertama-tama pengguna diminta untuk memasukkan alamat target dimana nilai tersebut akan dimasukkan kedalam variabel alamatURL. Selanjutnya dengan menggunakan *library BeautifulSoup*, aplikasi akan mengunjungi alamat target tersebut dan mengubah tampilannya kedalam bentuk *HTML*. Hal pertama yang dilakukan setelah aplikasi berhasil menampilkan halaman awal alamat target adalah mencari kata *login*. Hasil pencarian tersebut akan disimpan kedalam variabel kondisiPhising. Jika terdapat kata *login* pada halaman target, maka selanjutnya aplikasi akan mencari kata *src* pada halaman *login* target untuk mencari tau sumber yang digunakan dan menyimpan nilai tersebut pada variabel cekSumber. Jika terdapat kata *src*, selanjutnya aplikasi akan melakukan pengecekan apakah sumber yang digunakan mengandung kata *http* atau tidak.

Apabila sumber yang digunakan mengandung kata *http*, maka aplikasi akan menyalin kode halaman *login* target dan menyimpannya ke dalam *file index.html* dan memberikan informasi bahwa alamat target rentan terhadap serangan *Phising*. Namun, apabila sumber yang digunakan tidak terdapat memiliki kata *http*, maka aplikasi akan memberikan informasi bahwa alamat target tidak rentan terhadap serangan *Phising*. Dan jika tidak terdapat kata *login* pada halaman awal target, maka aplikasi akan memberikan informasi bahwa alamat target tidak rentan terhadap serangan *Phising*.

Selanjutnya aplikasi akan melakukan menguji alamat target dengan menggunakan jenis serangan *SQL Injection* dan *Cross-Site Scripting (XSS)*. Pertama-tama aplikasi akan melakukan pencarian kata *id=* pada halaman awal target. Jika terdapat kata *id=* pada halaman awal target, maka aplikasi akan melakukan injeksi dengan cara memberikan tanda petik satu (‘) pada akhir alamat yang mengandung kata *id=* dan mengunjunginya dengan menggunakan fungsi dari *library Urllib*. Selanjutnya aplikasi akan melakukan pencarian kata *SQL syntax*; pada halaman yang dikunjungi dan menyimpan hasilnya ke dalam variabel kondisiSQL. Jika terdapat kata *SQL syntax*; pada halaman yang dikunjungi, maka aplikasi akan memberikan informasi bahwa alamat target tersebut rentan terhadap serangan *SQL Injection*. Sedangkan, jika tidak terdapat kata *SQL syntax*;, maka aplikasi akan memberikan informasi bahwa alamat tersebut tidak rentan terhadap serangan *SQL Injection*.

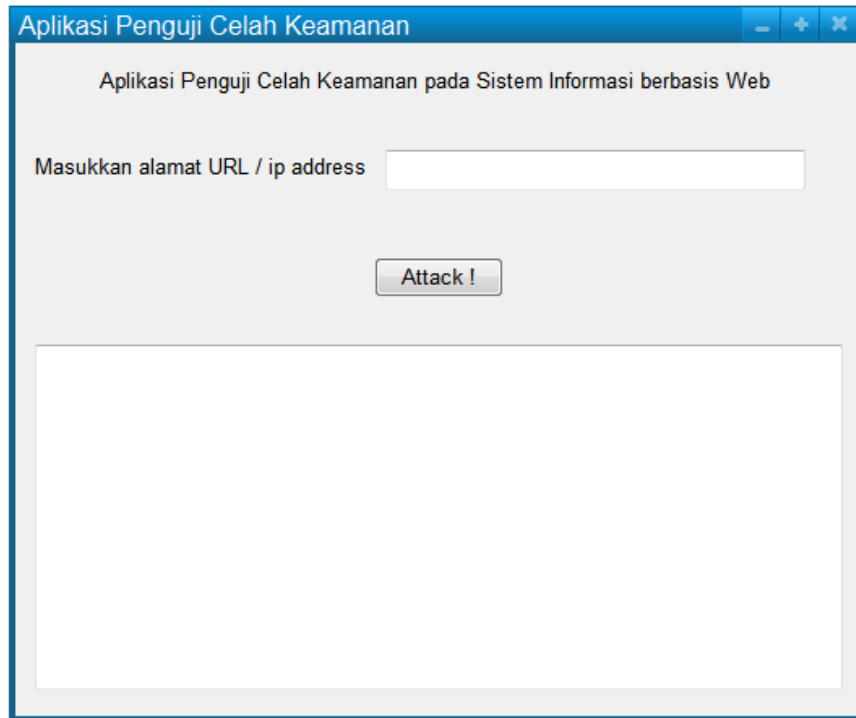
Terkahir, aplikasi akan menyisipkan *script* pada alamat yang mengandung kata *id=* di halaman awal target untuk mengujinya dengan jenis serangan *Cross-Site Scripting (XSS)*. Setelah disisipkan *script*, aplikasi akan mengunjungi alamat tersebut dan mencari tau apakah *script* tersebut berhasil berjalan atau tidak. Untuk mengetahui berjalan atau tidaknya *script* tersebut, maka aplikasi akan mencari kata *alert* pada *tag script* pada halaman tersebut. Jika terdapat kata *alert* pada *tag script*, maka aplikasi akan memberikan informasi bahwa alamat target rentan terhadap serangan *Cross-Site Scripting (XSS)*. Namun, jika tidak terdapat kata *alert* pada *tag script*, maka aplikasi akan memberikan informasi bahwa alamat target tidak rentan terhadap serangan *Cross-Site Scripting (XSS)*.

3.3.3 Desain Antarmuka

Desain antarmuka merupakan tahapan untuk memberikan gambaran mengenai antarmuka aplikasi yang akan dibangun. Dalam membuat desain antarmuka pada aplikasi ini, digunakan *tools Pencil*. Aplikasi Pengujian Celah Keamanan pada Aplikasi berbasis Web ini memiliki satu halaman antarmuka saja, yakni halaman awal.

Halaman awal merupakan halaman yang akan tampil ketika aplikasi dijalankan. Pada halaman awal, pengguna diminta untuk memasukkan alamat *URL* atau *ip address* target.

Setelah memasukkan alamat target, selanjutnya pengguna dapat menekan tombol “Attack !” untuk menjalankan aplikasi. Hasil akhirnya akan ditampilkan pada *textbox* yang berada dibawah tombol “Attack !”. Desain antarmuka halaman awal dapat dilihat pada Gambar 3.3.



Gambar 3.3 Desain Antarmuka Aplikasi Celah Keamanan pada Aplikasi Berbasis Web

3.4 Implementasi

Implementasi aplikasi dilakukan dengan menggunakan bahasa pemrograman *Python*. Untuk membuat *GUI* (*Graphic User Interface*), digunakan modul *PyQt5*. Modul *PyQt5* dipilih sebagai modul tampilan pengguna dibandingkan modul lainnya seperti *Tkinter* karena hasil tampilan yang dikeluarkan modul *PyQt5* lebih jernih dibandingkan modul lainnya. Hasil tampilan yang dikeluarkan modul *PyQt5* hampir sama seperti hasil tampilan yang digunakan oleh bahasa pemrograman *Java*. Karena semua jenis serangan yang terdapat pada aplikasi ini berhubungan dengan *website*, maka dibutuhkan teknik *web crawling* untuk menelusuri tautan-tautan yang terdapat pada *website* target. Keluaran yang diperoleh adalah informasi apakah halaman *website* yang dijadikan target rentan terhadap serangan yang digunakan pada aplikasi ini atau tidak.

3.5 Pengujian dan Evaluasi

Pengujian dan evaluasi merupakan sebuah tahapan untuk mengetahui apakah aplikasi yang dibangun telah sesuai dengan rancangan dan harapan yang diinginkan atau belum. Pengujian diperlukan untuk memastikan apakah aplikasi yang telah dibangun berjalan sesuai dengan yang diharapkan. Terdapat 2 jenis pengujian, yaitu pengujian *white box* dan pengujian *black box*. Pengujian *white box* adalah pengujian yang didasarkan pada pengecekan terhadap detail perancangan, sedangkan pengujian *black box* merupakan pengujian yang berfokus pada spesifikasi fungsional dari aplikasi (Mustaqbal, Firdaus, dkk., 2015). Evaluasi merupakan sebuah proses untuk menganalisis dan menilai apakah masih terdapat kesalahan atau *bugs* pada aplikasi yang telah dibangun.

3.5.1 Pengujian White Box

Pengujian *white box* merupakan pengujian yang dilakukan untuk menganalisa dan mengetahui apakah terdapat kesalahan pada kode program yang telah dibuat atau tidak. Jika kode program yang telah dibuat tidak menampilkan kesalahan (*error*), maka kode program sudah berjalan dengan baik. Namun, apabila terdapat *error* pada kode program seperti fungsi yang tidak berjalan dengan benar atau keluaran yang dihasilkan tidak sesuai dengan yang diharapkan, maka harus dilakukan pengecekan ulang terhadap kode tersebut. Beberapa hal yang akan diuji pada pengujian ini, yakni pengujian penggunaan percabangan dan pengujian penggunaan perulangan. Pengujian *white box* akan dilakukan dengan melakukan pengujian normal dan pengujian tidak normal.

A. Pengujian Normal

Pengujian normal adalah pengujian yang dilakukan terhadap aplikasi dengan memasukkan data yang berupa alamat target secara benar. Pada Aplikasi Pengujian Celah Keamanan pada Aplikasi berbasis Web ini terdapat 3 jenis serangan dan proses pengujian ini akan menilai apakah keluaran yang dikeluarkan oleh aplikasi tersebut sesuai dengan yang diharapkan atau masih terdapat *error*.

1. SQL Injection

Pengujian normal terhadap jenis serangan ini yakni dengan melihat apakah terdapat tautan yang memiliki kata “*id=*” didalamnya atau tidak dengan menggunakan teknik *web crawling*. Jika terdapat tautan yang memiliki kata “*id=*” didalamnya, maka proses injeksi akan berjalan. Apabila aplikasi *web* target tersebut rentan terhadap serangan ini, maka akan muncul keluaran bahwa aplikasi *web* tersebut rentan terhadap jenis serangan *SQL Injection*.

Sebaliknya, jika aplikasi *web* target tidak rentan terhadap serangan ini, maka akan muncul keluaran bahwa aplikasi *web* tersebut tidak rentan terhadap jenis serangan *SQL Injection*. Namun, apabila tidak terdapat tautan yang memiliki kata “*id=*” didalamnya, maka proses injeksi tidak akan dilakukan dan seluruh proses pada aplikasi akan berhenti.

2. *Cross-Site Scripting (XSS)*

Pengujian normal terhadap jenis serangan ini yakni dengan melihat apakah terdapat tautan yang memiliki kata “*id=*” didalamnya atau tidak dengan menggunakan teknik *web crawling*. Jika terdapat tautan yang memiliki kata “*id=*” didalamnya, maka proses menyisipkan *script* akan berjalan. Apabila *script* yang telah disisipkan berjalan pada tautan pada aplikasi *web* target, maka akan muncul keluaran bahwa aplikasi *web* tersebut rentan terhadap jenis serangan *Cross-Site Scripting (XSS)*. Sebaliknya, jika *script* yang telah disisipkan tidak berjalan pada halaman yang telah diinjeksi, maka akan muncul keluaran bahwa aplikasi *web* tersebut tidak rentan terhadap jenis serangan *Cross-Site Scripting (XSS)*. Namun, apabila tidak terdapat tautan yang memiliki kata “*id=*” didalamnya, maka proses penyisipan *script* tidak akan dilakukan dan seluruh proses pada aplikasi akan berhenti.

3. *Phising*

Pengujian normal terhadap jenis serangan ini yakni dengan melihat apakah terdapat tautan yang memiliki kata “*login*” didalamnya atau tidak dengan menggunakan teknik *web crawling*. Jika terdapat tautan yang memiliki kata “*login*” didalamnya, maka aplikasi akan mengunjungi halaman tautan tersebut. Selanjutnya aplikasi akan memeriksa apakah sumber gambar pada halaman tersebut berasal dari alamat lain atau tidak berasal dalam *server*, maka akan muncul keluaran bahwa aplikasi *web* tersebut rentan terhadap jenis serangan *Phising* dan aplikasi secara otomatis akan melakukan duplikasi terhadap halaman tautan tersebut. Sebaliknya, jika sumber gambar pada halaman target berasal dari *file* yang berada didalam *server*, maka akan muncul keluaran bahwa aplikasi *web* tersebut tidak rentan terhadap jenis serangan *Phising*. Namun, apabila tidak terdapat tautan yang memiliki kata “*login*” didalamnya, maka proses pencarian sumber gambar tidak akan dilakukan dan proses pada aplikasi akan berlanjut ke tahap pengujian serangan *SQL Injection*.

B. Pengujian Tidak Normal

Pengujian tidak normal adalah pengujian yang dilakukan terhadap aplikasi dengan memberikan masukan alamat *URL* atau *ip address* yang tidak dapat diakses atau salah. Ketika

pengguna memasukkan alamat *URL* atau *ip address* yang tidak dapat diakses atau salah, maka aplikasi akan memberi pemberitahuan kepada pengguna bahwa alamat yang dimasukkan tidak dapat diakses beserta dengan kode *error*. Selain itu, jika koneksi internet dalam proses pengujian tidak stabil ataupun koneksi internet terputus, maka aplikasi akan memberikan peringatan kepada pengguna untuk melakukan pengecekan terhadap koneksi internet yang digunakan.

3.5.2 Pengujian *Black Box*

Pengujian *black box* berfokus pada fungsional dari aplikasi yang telah dibangun. Beberapa hal yang akan diuji pada pengujian ini, diantaranya tombol yang terdapat pada aplikasi telah berfungsi dengan semestinya atau tidak, apakah terdapat kesalahan pada antarmuka aplikasi ketika proses pengujian celah keamanan berhasil dijalankan, dan pengujian performa dari aplikasi.

3.5.3 Evaluasi

Evaluasi merupakan tahapan untuk mengamati dan melihat kembali apakah aplikasi yang telah diuji telah sesuai dengan yang diharapkan atau belum. Jika terdapat kesalahan kode atau *error* pada proses pengujian aplikasi, maka akan dilakukan peninjauan ulang terhadap kode aplikasi. Sebaliknya, jika aplikasi sudah berjalan dan memberikan hasil keluaran sesuai dengan yang diharapkan, maka aplikasi telah berhasil dibangun.

Selain melihat apakah proses pengujian pada aplikasi sudah berjalan dan memberikan hasil keluaran yang sesuai atau belum, evaluasi terhadap ketepatan aplikasi juga dibutuhkan. Evaluasi terhadap ketepatan hasil dari aplikasi akan dilakukan dengan cara melakukan pengujian celah keamanan secara manual. Dan untuk mengetahui kelebihan dan kekurangan dari Aplikasi Pengujian Celah Keamanan pada Sistem Informasi Berbasis *Web*, maka akan dilakukan perbandingan antara Aplikasi Pengujian Celah keamanan pada Sistem Informasi Berbasis *Web* dengan aplikasi yang telah ada sebelumnya, yaitu *SQLMap* dan *SET (Social Engineer Toolkit)*.

BAB IV

IMPLEMENTASI DAN PENGUJIAN

Implementasi aplikasi menjelaskan tentang hasil yang diperoleh dari tahapan sebelumnya yakni tahap analisis dan perancangan. Pada implementasi aplikasi juga akan dijelaskan proses-proses apa saja yang berjalan didalam aplikasi ini. Pengujian Aplikasi Celah Keamanan pada Aplikasi Berbasis *Web* ini dilakukan dengan 2 tahap, yaitu pengujian normal dan pengujian tidak normal. Selain itu, untuk mengetahui keakuratan hasil yang ditampilkan oleh aplikasi, maka akan dilakukan pengujian secara manual terhadap *website* target. Dan untuk mengetahui kelemahan dan kelebihan dari Aplikasi Pengujian Celah Keamanan pada Aplikasi *Web*, maka akan dilakukan perbandingan terhadap aplikasi sejenis yang telah ada sebelumnya, yakni *SqlMap* dan *SET*.

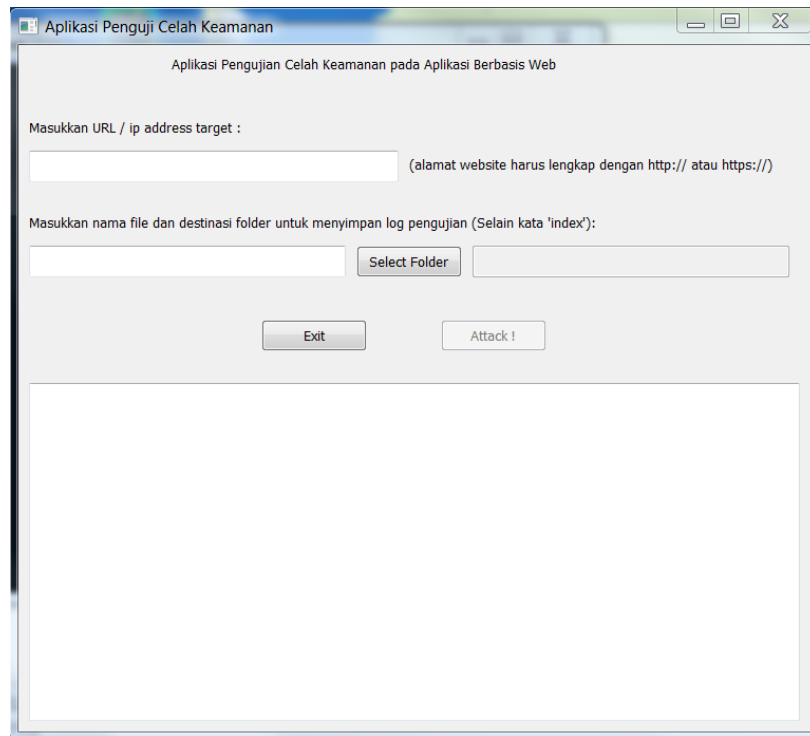
4.1 Implementasi Aplikasi

4.1.1 Implementasi Antarmuka

Pada Aplikasi Pengujian Celah Keamanan pada Aplikasi Berbasis *Web* ini hanya memiliki satu halaman, yakni halaman awal. Halaman awal tersebut berisi *form* untuk mengisi alamat target, *form* untuk mengisi nama *file* untuk menyimpan *log* pengujian, tombol untuk mengarahkan *folder* tujuan tempat penyimpanan *file log*, tombol *exit*, tombol *attack*, dan *form text area* yang berfungsi untuk menampilkan hasil laporan dari proses pengujian.

Pada *form* alamat target, pengguna diminta untuk mengisi alamat target dengan benar yaitu diawali dengan “*http://*” atau “*https://*”. Pada *form* nama *file*, pengguna diminta memasukkan nama *file* yang nantinya digunakan untuk menyimpan *log* pengujian. Nama *file* yang dimasukkan harus selain “*index*” karena kata “*index*” digunakan untuk menyimpan *source code* halaman *login* pada *website* yang rentan terhadap serangan *Phising*. Selanjutnya, pengguna juga diminta untuk memilih *folder* tujuan untuk menyimpan *file log* pengujian.

Pada halaman awal terdapat 2 proses, yaitu proses keluar dan proses pengujian. Proses keluar merupakan proses yang akan membuat pengguna keluar dari aplikasi. Sedangkan proses pengujian adalah proses yang digunakan untuk menjalankan proses pengujian terhadap target. Proses keluar dijalankan dengan menekan tombol *exit* dan proses pengujian dijalankan dengan menekan tombol *Attack !*. Implementasi halaman awal dapat dilihat pada Gambar 4.1.



Gambar 4.1 Halaman Awal Aplikasi Pengujian Celah Keamanan

4.1.2 Implementasi Proses Serangan *Phising*

Jenis serangan pertama yang akan diujikan kepada alamat target adalah serangan *Phising*. Kode program serangan *Phising* secara keseluruhan dapat dilihat pada Gambar 4.2. Pada baris 129 terdapat variabel dengan nama “*soupPhising*”. Variabel tersebut memiliki fungsi untuk membaca hasil respon dari *server* target dengan format “*html5lib*”. Selanjutnya, kode pada baris 130 dibuat sebuah kondisi untuk melakukan pencarian kata “*login*” pada *tag anchor*. Kode baris 132 hingga baris 137 menjelaskan apabila kondisi yang telah dibuat pada baris 130 benar, maka nilai dari kondisi tersebut akan dimasukkan kedalam *array* “*URLPhising*”. Namun jika kondisi tersebut salah, maka aplikasi akan memberikan keluaran informasi bahwa alamat target tidak rentan terhadap serangan *Phising*.

Selanjutnya, kode pada baris 149 hingga baris 165 menjelaskan bahwa aplikasi akan membaca nilai yang ada pada *array* “*URLPhising*” berupa halaman *login* target dan mengunjungi halaman tersebut. Selanjutnya, aplikasi akan melakukan pencarian kata “*http*” pada *tag image* dengan atribut *src (source)* pada *array* tersebut. Kode pada baris 169 hingga baris 201 menjelaskan jika halaman *login* target menggunakan sumber (*source*) yang mengandung kata “*http*”, maka aplikasi akan memberikan keluaran informasi bahwa alamat target rentan terhadap serangan *Phising*. Setelah itu aplikasi akan melakukan duplikasi terhadap halaman “*login*” target dan menyimpannya di dalam file “*index.html*” . Sebaliknya,

jika *source* yang digunakan tidak mengandung kata “*http*” sama sekali, maka aplikasi akan memberikan keluaran informasi bahwa alamat target tidak rentan terhadap serangan *Phising*.

```

129. soupPhising = BeautifulSoup(response, 'html5lib')
130. kondisiPhising = soupPhising.find_all('a', href=re.compile('login'))
132. if len(kondisiPhising) > 0:
133.     for b in kondisiPhising:
134.         URLPhising.append(b['href'])
136. else:
137.     print(datetime.now().strftime("%d-%m-%Y %H:%M:%S")+" " +URLAsli, "tidak
147. rentan terhadap serangan Phising")
148. intPhising = 0
149. tdkPhising = []
150. for phising in URLPhising:
151.     try:
152.         responsePhising = urllib.request.urlopen(URLPhising[intPhising]).read()
153.         soupCopasPhising = BeautifulSoup(responsePhising, 'html5lib')
155.         kondisicekPhising = soupCopasPhising.find_all('img', {"src": True})
156.         if len(kondisicekPhising) > 0:
157.             letter = ["http"]
158.             x = 0
159.             for c in kondisicekPhising:
160.                 srcPhising.append(c['src'])
161.                 first = srcPhising[x][:4]
162.                 if first in letter != -1:
163.                     resultPhising.append("rentan")
164.                 else:
165.                     tdkPhising.append("a")
169.             x = x + 1
170.             if len(resultPhising) > 0:
171.                 kondisiCopasPhising = soupCopasPhising.find_all('html')
172.                 saveFileHTML = os.path.join(lokasiPath, 'index.html')
172.                 filePhising = open(saveFileHTML, "w", encoding='utf-8')
173.                 filePhising.write(str(kondisiCopasPhising))
174.                 filePhising.close()
175.                 print(datetime.now().strftime("%d-%m-%Y %H:%M:%S") + " " + URLAsli,
176. "rentan terhadap serangan Phising !")
187.             else:
188.                 print(datetime.now().strftime("%d-%m-%Y %H:%M:%S") + " " + URLAsli, "
189. tidak rentan terhadap serangan Phising")
198.             intPhising = intPhising + 1
199.             except (ValueError, URLError) as e:
200.                 print(datetime.now().strftime("%d-%m-%Y %H:%M:%S") + " " + URLAsli,
201. "tidak rentan terhadap serangan Phising")

```

Gambar 4.2 Kode Program Serangan *Phising*

4.1.3 Implementasi Proses Serangan *SQL Injection*

Jenis serangan kedua yang akan diujikan kepada alamat target adalah serangan *SQL Injection*. Kode program serangan *SQL Injection* secara keseluruhan dapat dilihat pada Gambar 4.3. Kode baris 203 terdapat variabel dengan nama “*Soup*” yang memiliki fungsi untuk membaca hasil respon dari *server* target dengan format “*html5lib*”. Selanjutnya, kode pada baris 204 terdapat kondisi untuk melakukan pencarian kata “*id=*” pada *tag anchor*. Kode pada baris 207 hingga baris 213 menjelaskan jika kondisi pada baris 204 benar, maka nilai dari kondisi tersebut akan dimasukkan kedalam *array* “*URLSql*” dan *array* “*URLXSS*”.

Selanjutnya, kode pada baris 218 hingga baris 222 menjelaskan bahwa aplikasi akan membaca nilai berupa alamat *website* yang ada pada *array* “URLSql” dan melakukan injeksi dengan menyisipkan tanda petik satu (‘) atau dalam *encoding* adalah “%27” pada bagian akhir alamat. Setelah itu, aplikasi akan mengunjungi halaman yang telah diinjeksi pada *array* “URLSql” dan akan mencari kata “SQL syntax;” pada *tag body*. Kode pada baris 224 hingga 245 menjelaskan jika terdapat kata “SQL syntax;” pada halaman tersebut, maka aplikasi akan memberikan keluaran informasi bahwa alamat target rentan terhadap serangan *SQL Injection*. Sebaliknya, jika tidak terdapat kata “SQL syntax” pada halaman tersebut, maka aplikasi akan memberikan keluaran informasi bahwa alamat target tidak rentan terhadap serangan *SQL Injection*.

```

203. Soup = BeautifulSoup(response, 'html5lib')
204. kondisi = soup.find_all('a', href=re.compile('id='))
207. if len(kondisi) > 0:
208.     y = urlparse(URLAsli)
209.     URLFix = y.scheme + "://" + y.netloc + y.path
211.     for a in kondisi:
212.         URLSql.append(URLFix + a['href'])
213.         URLXss.append(URLFix + a['href'])
218. for sql in URLSql:
219.     try:
220.         responseSQL = urllib.request.urlopen(URLSql[intSQL] + "%27").read(25373)
221.         soupSQL = BeautifulSoup(responseSQL, 'html5lib')
222.         kondisiSQL = soupSQL.find_all('body', string=re.compile('SQL syntax;'))
224.         if len(kondisiSQL) > 0:
225.             print(datetime.now().strftime("%d-%m-%Y %H:%M:%S") + " " + " " +
226.                   URLSql[intSQL] + " rentan terhadap serangan SQL Injection")
235.             tdkSQL.append("a")
236.             intSQL = intSQL + 1
241.         except http.client.IncompleteRead as e:
242.             responseSQL = e.partial
243.         if len(tdkSQL) == 0:
244.             print(datetime.now().strftime("%d-%m-%Y %H:%M:%S") + " " + URLAsli, "tidak
245.             rentan terhadap serangan SQL Injection")

```

Gambar 4.3 Kode Program Serangan *SQL Injection*

4.1.4 Implementasi Proses Serangan *Cross-Site Scripting (XSS)*

Jenis serangan terakhir yang akan diujikan kepada alamat target adalah serangan *Cross-Site Scripting (XSS)*. Kode program serangan *Cross-Site Scripting* secara keseluruhan dapat dilihat pada Gambar 4.4. Pertama-tama aplikasi akan membaca nilai berupa alamat *website* yang ada pada *array* “URLXSS” seperti yang terlihat pada kode dibaris 258. Selanjutnya, kode dibaris 260 menjelaskan bahwa aplikasi akan menyisipkan *script alert* (pemberitahuan) pada akhir alamat dan mengunjungi alamat yang telah disisipkan *script* tersebut. Setelah itu, kode pada baris 262 menjelaskan sebuah variabel yang berfungsi untuk mencari kata “alert” pada *tag script*. Kode pada baris 264 hingga baris 283 menjelaskan sebuah kondisi, dimana

jika terdapat kata “*alert*” pada *tag script* di halaman target, maka aplikasi akan memberikan keluaran informasi bahwa alamat target rentan terhadap serangan *Cross-Site Scripting (XSS)*. Sebaliknya, jika kondisi tersebut salah, maka aplikasi akan memberikan keluaran informasi bahwa alamat target tidak rentan terhadap serangan *Cross-Site Scripting (XSS)*. Kode pada baris 279 hingga baris 280 berfungsi untuk menjalankan fungsi dari titik terakhir (bukan dari awal) ketika proses pada aplikasi membutuhkan waktu yang lebih lama. Untuk keseluruhan kode program aplikasi dapat dilihat pada halaman lampiran.

```

258. for sql in URLXSS:
259.     try:
260.         responseXSS = urllib.request.urlopen(URLXss[intXSS]+
260. "%22%3E%3Cscript%3Ealert%28222%29%3C%2Fscript%3E").read()
261.         soupXSS = BeautifulSoup(responseXSS, 'html5lib')
262.         kondisiXSS = soupXSS.find_all('script', string=re.compile('alert'))
264.         if len(kondisiXSS) > 0:
265.             print(datetime.now().strftime("%d-%m-%Y %H:%M:%S") + " " +
266. URLXss[intXSS] + " rentan terhadap serangan Cross-Site Scripting")
274.             tdkXSS.append("tidak")
275.             intXSS = intXSS + 1
279.         except http.client.IncompleteRead as e:
280.             responseXSS = e.partial
281.         if len(tdkXSS) == 0:
282.             print(datetime.now().strftime("%d-%m-%Y %H:%M:%S") + " " + URLAsli, "tidak
283. rentan terhadap serangan Cross-Site Scripting")

```

Gambar 4.4 Kode Program Serangan *Cross-Site Scripting (XSS)*

4.2 Pengujian Aplikasi

Pada tahapan ini, akan dilakukan pengujian terhadap aplikasi yang telah dibuat sebelumnya untuk mengetahui apakah hasil yang dikeluarkan oleh aplikasi telah sesuai dengan yang diharapkan atau belum. Target pengujian terdiri dari alamat *website* yang berada di Indonesia dan alamat *website* yang berasal dari luar negeri. Target tersebut telah ditentukan oleh peneliti dan dikelompokkan kedalam 4 kelompok, diantaranya *website e-commerce* (situs penjualan produk), *website pemerintahan*, *website pendidikan*, dan *website pariwisata* (situs pemberi informasi tempat wisata di suatu tempat atau situs objek wisata). Setiap kelompok masing-masing terdiri dari 3 alamat target, sehingga total target pengujian ada 24 alamat *website*. Daftar kelompok dan alamat target dapat dilihat pada Tabel 4.1.

Tabel 4.1 Daftar Kelompok dan Alamat Target

No	Kelompok	Alamat Target Situs Indonesia	Alamat Target Situs Luar Negeri
1.	Website E-commerce	http://diskon.com	http://readyicons.com

		http://benscrub.com	http://id.ee
		http://ittaherl.com	http://ampak.com.tw
2.	Website Pemerintahan	http://buku.kemdikbud.go.id	http://qld.gov.au
		http://aspirasi.kotabogor.go.id	https://gov.uk
		http://jkn.kemkes.go.id	https://evisa.moip.gov.mm
3.	Website Pendidikan	http://unla.ac.id	http://spu.edu
		http://stikespku.ac.id	http://assumption.edu
		http://widyagamahusada.ac.id	http://beverlyhills.co.in
4.	Website Pariwisata	http://tourism-makassar.id	http://nightgallery.ca
		http://bandungtourism.com	http://boothillcasino.com
		http://visitingjogja.com	http://adventuretours.com.au

4.2.1 Pengujian White Box

A. Pengujian Normal

Pada tahap ini akan dilakukan pengujian dengan memberikan masukan alamat target yang benar. Pada proses pengujian akan terdapat 4 tahapan, yakni tahap serangan *Phising*, tahap *crawling id=*, tahap serangan *SQL Injection*, dan tahap serangan *Cross-Site Scripting (XSS)*. Tahap serangan *Phising* merupakan tahap pertama yang akan dilakukan oleh aplikasi ketika proses pengujian dijalankan. Pada tahap ini, aplikasi akan melakukan *crawling* terhadap halaman awal pada target dan mencari kata “login” pada atribut “href” didalam *tag anchor* (< a >). Jika aplikasi menemukan kata “login” pada atribut “href” didalam *tag anchor* (< a >), maka selanjutnya aplikasi akan mengunjungi halaman tersebut. Namun, jika aplikasi tidak menemukan kata “login”, maka proses pengujian akan langsung berlanjut pada tahap *crawling id=*.

Selanjutnya, aplikasi kembali melakukan *crawling* pada halaman *login* yang dikunjungi. Pada proses *crawling* ini, aplikasi akan melakukan pemeriksaan terhadap sumber (*source*) gambar yang digunakan oleh target. Sumber gambar yang diperiksa berada dalam atribut “src” pada *tag image* (). Apabila sumber gambar yang digunakan oleh target berasal dari alamat lain dan bukan berasal dari *server*, maka aplikasi akan memberikan keluaran berupa informasi bahwa target rentan terhadap serangan *Phising*. Sedangkan, apabila sumber yang digunakan oleh target berasal dari *server*, maka aplikasi akan memberikan keluaran berupa informasi bahwa target tidak rentan terhadap serangan *Phising*. Dan apabila sumber gambar yang digunakan oleh target merupakan sumber gabungan dari alamat lain dan dari *server* sendiri, maka aplikasi akan memberikan keluaran berupa informasi bahwa target rentan terhadap serangan *Phising*. Ketika aplikasi memberikan informasi bahwa target rentan terhadap serangan *Phising*, maka aplikasi akan melakukan duplikasi terhadap halaman *login* target dan menyimpannya ke dalam *file* yang bernama

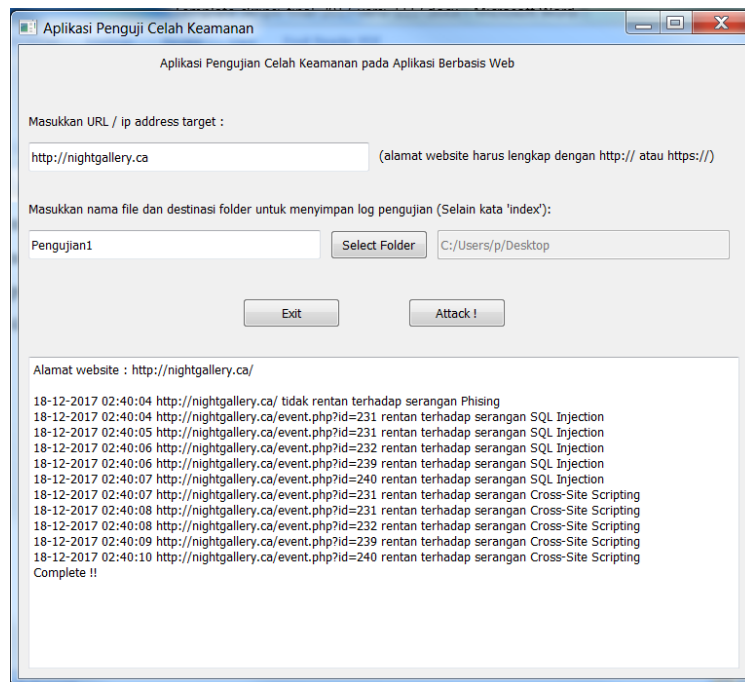
“*index.html*”. Setelah tahap serangan *Phising* selesai, selanjutnya target akan masuk ke dalam tahap *crawling id=*.

Pada tahap *crawling id=*, aplikasi akan melakukan *crawling* untuk mencari kata “*id=*” pada atribut “*href*” didalam *tag anchor* (< a >). Hasil dari proses *crawling* ini nantinya akan digunakan oleh tahap serangan *SQL Injection* dan tahap serangan *Cross-Site Scripting (XSS)*. Jika hasil *crawling* menunjukkan adanya kata “*id=*” pada atribut “*href*” didalam *tag anchor* (< a >), maka selanjutnya target akan masuk kedalam tahap proses serangan *SQL Injection*. Sedangkan, jika hasil *crawling* menunjukkan tidak adanya kata “*id=*” pada atribut “*href*” didalam *tag anchor* (< a >), maka aplikasi akan memberikan keluaran berupa informasi bahwa target tidak rentan terhadap serangan *SQL Injection* dan serangan *Cross-Site Scripting (XSS)*.

Ketika target masuk kedalam tahap *SQL Injection*, alamat hasil dari *crawling id=* akan disisipkan tanda petik satu (‘) dibagian belakang. Selanjutnya aplikasi akan mengunjungi alamat yang telah disisipkan tanda petik satu (diinjeksi). Jika terdapat kata “*SQL syntax;*” pada halaman tersebut, maka aplikasi akan memberikan keluaran berupa informasi bahwa target rentan terhadap serangan *SQL Injection*. Tetapi apabila tidak terdapat kata “*SQL syntax;*”, maka aplikasi akan memberikan informasi bahwa target tidak rentan terhadap serangan *SQL Injection*. Setelah tahap serangan *SQL Injection* selesai, maka target akan masuk kedalam tahap terakhir yaitu tahap serangan *Cross-Site Scripting (XSS)*.

Pada tahap serangan *Cross-Site Scripting (XSS)*, *script alert* akan disisipkan pada bagian akhir alamat yang berasal dari hasil *crawling id=*. Selanjutnya aplikasi akan mengunjungi alamat yang telah disisipkan *script alert* tersebut. Apabila *script* yang telah disisipkan berjalan, maka aplikasi akan memberikan keluaran berupa informasi bahwa target rentan terhadap serangan *Cross-Site Scripting (XSS)*. Namun, apabila *script* yang disisipkan tidak berjalan, maka aplikasi akan memberikan keluaran bahwa target tidak rentan terhadap serangan *Cross-Site Scripting (XSS)*. Setelah semua tahap dijalankan, maka hasil rentan atau tidak rentannya target terhadap serangan yang terdapat didalam aplikasi (*Phising*, *SQL Injecction*, *Cross-Site Scripting (XSS)*) akan ditampilkan pada *form text area*. Selain itu, aplikasi ini juga menyimpan hasil proses pengujian didalam *file* yang nama dan lokasi penyimpanannya telah ditentukan oleh pengguna (dalam contoh ini, *file* diberi nama “*Pengujian1*” dan lokasi penyimpanannya di *Desktop*). Hasil proses pengujian tersebut tersimpan pada *file* yang berekstensi “.html”, sehingga pengguna dapat melihat hasil tersebut pada *browser*. Hasil dari pengujian normal terhadap salah satu target dapat dilihat pada Gambar 4.5 dan hasil proses pengujian yang tersimpan pada *file* “*Pengujian1*” dapat dilihat

pada Gambar 4.6. Sedangkan, untuk hasil dari pengujian normal terhadap keseluruhan target yang telah ditentukan dapat dilihat pada Tabel 4.2.



Gambar 4.5 Hasil Pengujian terhadap Alamat <http://nightgallery.ca>

Log Pengujian

Alamat website : <http://nightgallery.ca>/

No.	Tanggal Pengujian	Alamat Target	Keterangan
1.	18-12-2017 02:40:04	http://nightgallery.ca/	Tidak rentan terhadap serangan Phishing
2.	18-12-2017 02:40:04	http://nightgallery.ca/event.php?id=231	Rentan terhadap serangan SQL Injection
3.	18-12-2017 02:40:05	http://nightgallery.ca/event.php?id=231	Rentan terhadap serangan SQL Injection
4.	18-12-2017 02:40:06	http://nightgallery.ca/event.php?id=232	Rentan terhadap serangan SQL Injection
5.	18-12-2017 02:40:06	http://nightgallery.ca/event.php?id=239	Rentan terhadap serangan SQL Injection
6.	18-12-2017 02:40:07	http://nightgallery.ca/event.php?id=240	Rentan terhadap serangan SQL Injection
7.	18-12-2017 02:40:07	http://nightgallery.ca/event.php?id=231	Rentan terhadap serangan Cross-Site Scripting (XSS)
8.	18-12-2017 02:40:08	http://nightgallery.ca/event.php?id=231	Rentan terhadap serangan Cross-Site Scripting (XSS)
9.	18-12-2017 02:40:08	http://nightgallery.ca/event.php?id=232	Rentan terhadap serangan Cross-Site Scripting (XSS)
10.	18-12-2017 02:40:09	http://nightgallery.ca/event.php?id=239	Rentan terhadap serangan Cross-Site Scripting (XSS)
11.	18-12-2017 02:40:10	http://nightgallery.ca/event.php?id=240	Rentan terhadap serangan Cross-Site Scripting (XSS)

Gambar 4.6 Hasil file “Pengujian1.html”

Tabel 4.2 Hasil Pengujian terhadap Seluruh Target

No	Alamat Target	Serangan Sql Injection	Serangan XSS	Serangan Phising
1.	http://diskon.com	-	-	√
2.	http://benscrub.com	-	-	√
3.	http://ittaherl.com	-	-	√
4.	http://buku.kemdikbud.go.id	-	-	√
5.	http://aspirasi.kotabogor.go.id	-	-	√

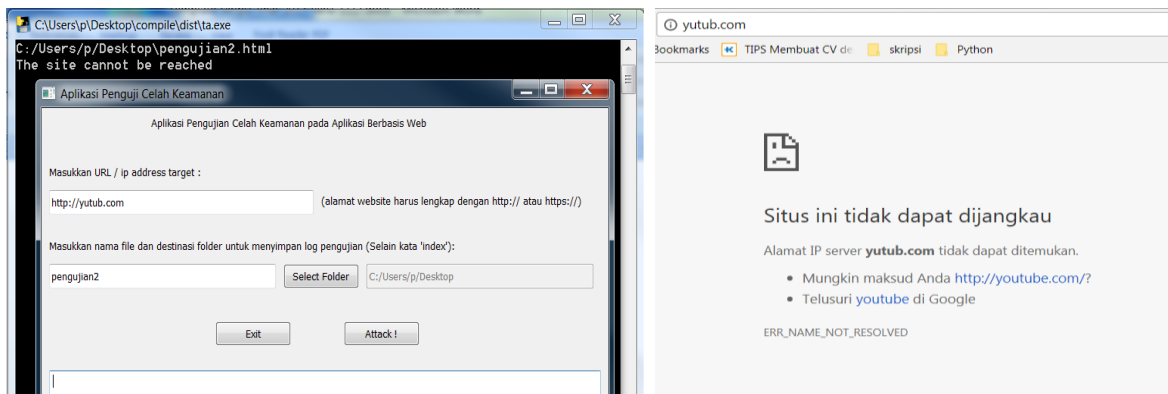
6.	http://pu.go.id	-	-	-
7.	http://unla.ac.id	-	-	√
8.	http://stikespku.ac.id	-	-	√
9.	http://widyagamahusada.ac.id	-	√	-
10.	http://tourism-makassar.id	-	-	-
11.	http://bandungtourism.com	-	-	-
12.	http://visitingjogja.com	-	-	-
13.	http://readyicons.com	-	√	-
14.	http://id.ee	-	-	-
15.	http://ampak.com.tw	√	√	-
16.	http://qld.gov.au	-	-	-
17.	https://gov.uk	-	-	-
18.	https://evisa.moip.gov.mm	-	-	-
19.	http://spu.edu	-	-	-
20.	http://assumption.edu	-	-	-
21.	http://beverlyhills.co.in	-	-	-
22.	http://nightgallery.ca	-	-	-
23.	http://boothillcasino.com	-	-	-
24.	http://adventuretours.com.au	-	-	-

B. Pengujian Tidak Normal

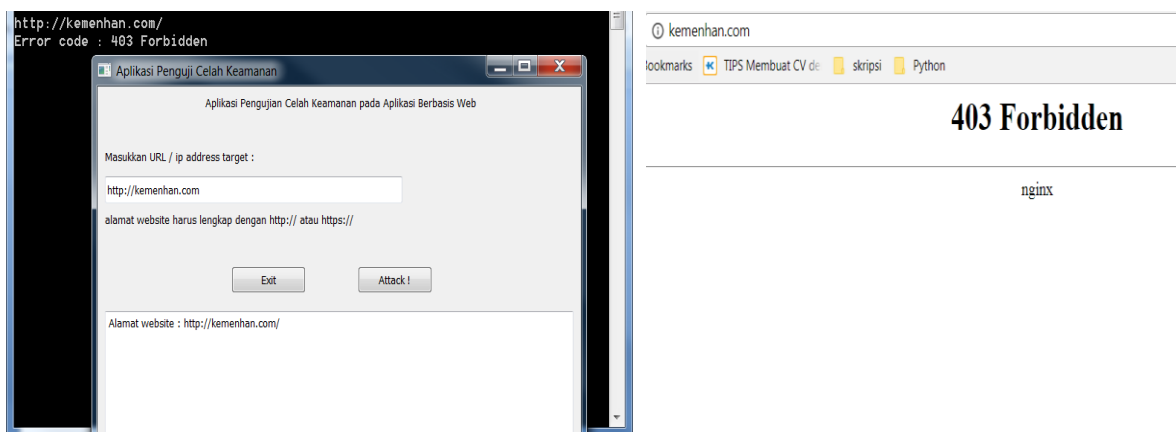
Pada tahap pengujian tidak normal, aplikasi akan diberikan masukan berupa alamat *website* yang salah. Ketika aplikasi menerima alamat *website* yang salah atau tidak dapat diakses, maka aplikasi akan memberikan keluaran berupa kode kesalahan dan informasi kesalahannya. Namun, kode kesalahan dan informasi kesalahan tersebut tidak dapat ditampilkan didalam *form text area* ataupun pada *file log*, melainkan menampilkannya di dalam *command prompt*. *Command prompt* sendiri memiliki tugas untuk memberikan keluaran informasi secara *real-time*. Hal tersebut terjadi dikarenakan untuk menangani kode kesalahan yang muncul, dibutuhkan penanganan dengan menggunakan *try exception* pada bahasa pemrograman *python*. Namun, penggunaan *try exception* tersebut akan membuat hasil *error* yang muncul hanya dapat dikeluarkan di dalam *command prompt*.

Pada pengujian tidak normal ini, peneliti memasukkan 2 alamat yang salah, yakni “*http://yutub.com*” dan “*http://kemenhan.com*”. Ketika kita mengakses “*http://yutub.com*” pada *browser*, maka informasi yang muncul adalah “*This site cannot be reached*” karena alamat *website* tersebut memang tidak ada, sedangkan ketika kita mengakses “*http://kemenhan.com*” di *browser*, maka informasi yang akan muncul adalah “*403 Forbidden*”. Informasi tersebut nantinya akan tampil di dalam *command prompt*. Selain informasi tersebut, kode *error* (kesalahan) yang terdapat pada halaman tersebut juga akan ditampilkan seperti kode *error* 403 yang muncul ketika kita mengakses *website*

“<http://kemenhan.com>”. Hasil dari pengujian tidak normal dapat dilihat pada Gambar 4.7 dan Gambar 4.8.



Gambar 4.7 Hasil Pengujian terhadap Alamat <http://yutub.com>



Gambar 4.8 Hasil Pengujian terhadap Alamat <http://kemenhan.com>

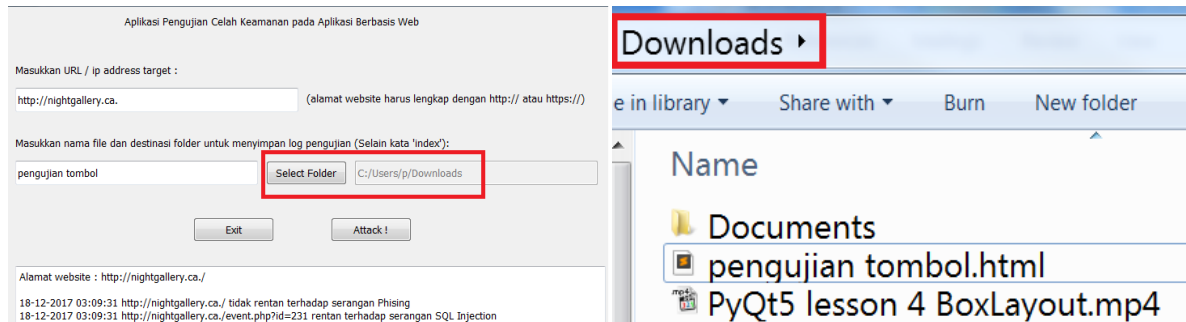
4.2.2 Pengujian *Black Box*

Pada tahap ini, fungsional aplikasi yang telah dibangun akan diuji. Adapun hal-hal yang akan diuji, diantaranya pengujian tombol pada aplikasi, pemuan kesalahan antarmuka aplikasi ketika proses pengujian celah keamanan dijalankan, dan pengujian performa dari aplikasi.

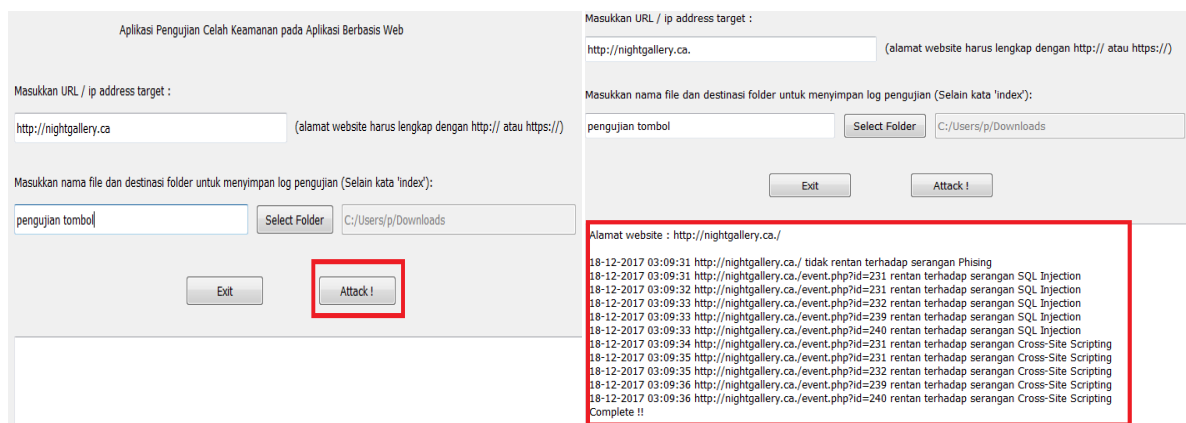
1. Pengujian Tombol pada Aplikasi

Pada pengujian ini, ketiga tombol yang terdapat pada aplikasi akan diuji apakah berjalan sesuai fungsinya atau belum. Pertama, tombol yang akan diuji adalah tombol “*Select Folder*”. Untuk melakukan pengujian pada tombol tersebut, peneliti mengarahkan penyimpanan *log* pengujian ke *folder* “*Downloads*”. Setelah dilakukan pengujian, tombol “*Select Folder*” berfungsi dengan baik. Hasil pengujian dapat dilihat pada Gambar 4.9.

Selanjutnya, tombol “Exit” yang akan diuji. Setelah dilakukan pengujian, tombol “Exit” berfungsi dengan baik. Dan terakhir pengujian pada tombol “Attack !”. Setelah dilakukan pengujian, tombol “Attack !” berfungsi dengan baik. Hasil pengujian tersebut dapat dilihat pada Gambar 4.10.



Gambar 4.9 Hasil Pengujian Tombol *Select Folder*



Gambar 4.10 Hasil Pengujian Tombol *Attack !*

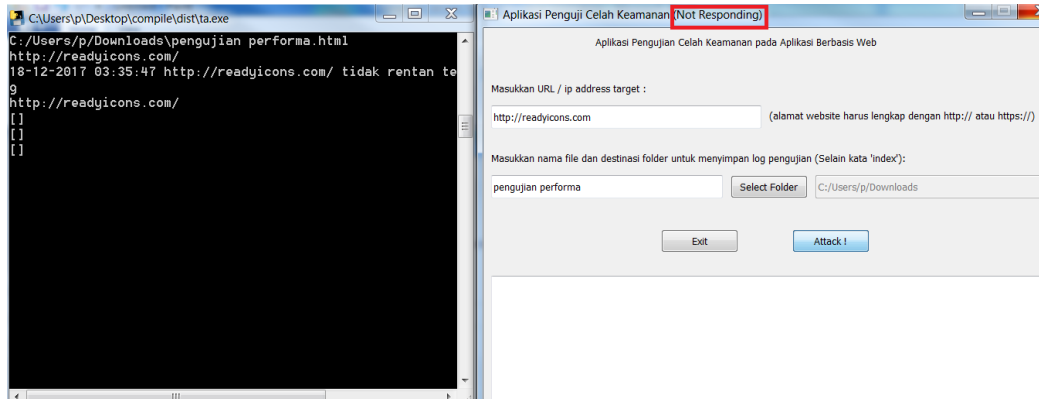
2. Penemuan Kesalahan Antarmuka

Pada tahap ini, pengujian akan dilakukan terhadap aplikasi dengan cara menjalankan aplikasi seperti biasa. Selanjutnya akan dilihat apakah terdapat kesalahan antarmuka pada aplikasi atau tidak. Setelah menjalankan aplikasi dari awal hingga proses pengujian selesai, tidak didapatkan kesalahan pada antarmuka aplikasi.

3. Pengujian Performa dari Aplikasi

Pada tahap ini, pengujian akan dilakukan dengan cara menjalankan fungsi pengujian terhadap sebuah alamat target. Selanjutnya akan dilihat apakah performa dari aplikasi terlihat normal atau tidak. Target alamat yang diuji adalah “*http://readyicons.com*”. Setelah

dilakukan pengujian, ternyata performa aplikasi kurang begitu bagus. Dapat dilihat pada garis merah yang berada di Gambar 4.11 bahwa aplikasi mengalami “Not Responding”. Namun, sebenarnya proses pengujian celah keamanan tetap berjalan. Hal tersebut dapat dilihat pada *command prompt* aplikasi.



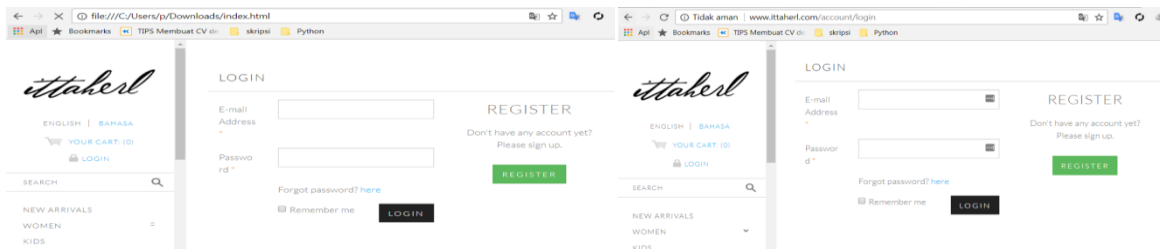
Gambar 4.11 Hasil Pengujian Performa Aplikasi

4.2.3 Pengujian Secara Manual

Pengujian secara manual merupakan tahap pengujian untuk membuktikan apakah hasil yang dikeluarkan oleh Aplikasi Pengujian Celah Keamanan pada Aplikasi Berbasis Web tersebut akurat atau tidak. Pengujian secara manual ini akan dilakukan pada 3 alamat *website*, yaitu “<http://ittaherl.com>”, “<http://nightgallery.com>”, dan “<http://widyagamahusada.ac.id>”. Aplikasi Pengujian Celah Keamanan pada Aplikasi Berbasis Web, mengatakan bahwa *website* “<http://ittaherl.com>” rentan terhadap serangan *Phising*. Setelah dilakukan pengujian secara manual, *website* <http://ittaherl.com> memang rentan terhadap serangan *Phising*. Dapat dilihat pada Gambar 4.12 bahwa sumber gambar yang digunakan oleh alamat *website* “<http://ittaherl.com>” berasal dari alamat lain. Dan hasil *phising* halaman *login* dari *website* target dapat dilihat pada Gambar 4.13. Tampilan halaman *login* dari hasil *phising* secara umum persis dengan halaman *login* aslinya. Hal ini tentunya dapat dimanfaatkan oleh pihak-pihak yang tidak bertanggung jawab.

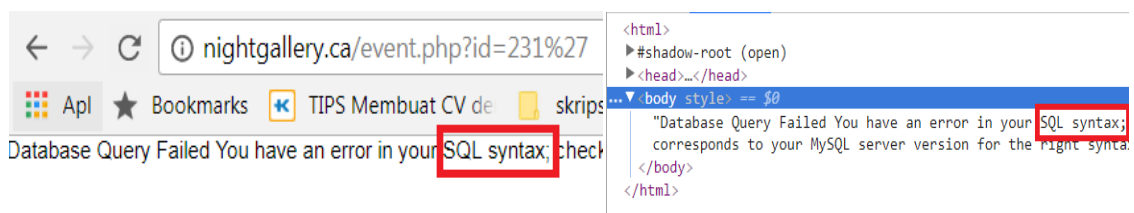


Gambar 4.12 Hasil Inspeksi pada Halaman Login <http://ittaherl.com>

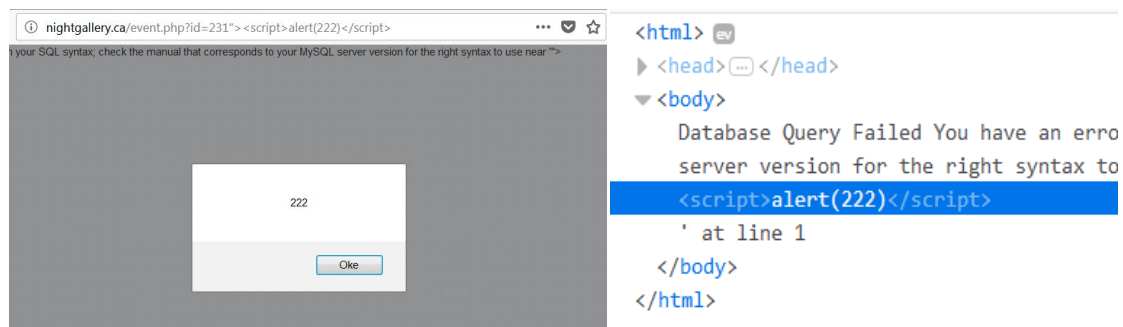


Gambar 4.13 Hasil Phising Halaman Login <http://ittaherl.com>

Selanjutnya, pengujian secara manual akan dilakukan pada alamat *website* “<http://nightgallery.ca>”. Aplikasi pengujian mengatakan bahwa alamat *website* tersebut rentan terhadap serangan *SQL Injection* dan serangan *Cross-Site Scripting (XSS)*. Setelah dilakukan pengujian secara manual terhadap *website* tersebut, terbukti bahwa alamat *website* “<http://nightgallery.ca>” rentan terhadap serangan *SQL Injection* dan serangan *Cross-Site Scripting (XSS)*. Dapat dilihat pada Gambar 4.14 bahwa setelah alamat tautan yang mengandung kata “*id=*” disisipkan tanda petik satu, terlihat kalimat *error* yang mengandung kata “*SQL syntax;*”. Selanjutnya pada Gambar 4.15 dapat dilihat bahwa setelah alamat tautan yang mengandung kata “*id=*” disisipkan *script*, *script* dapat berjalan pada halaman tersebut.

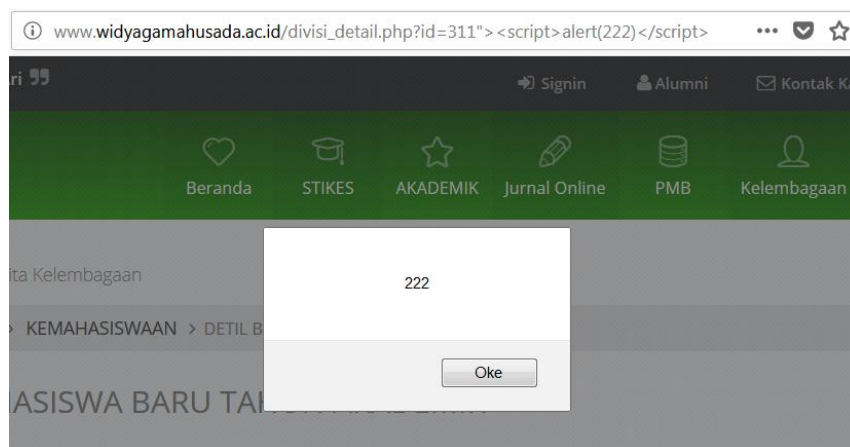


Gambar 4.14 Hasil Serangan *SQL Injection*



Gambar 4.15 Hasil Serangan *Cross-Site Scripting (XSS)*

Terakhir, pengujian secara manual dilakukan terhadap alamat *website* “<http://widyagamahusada.ac.id>”. Aplikasi mengatakan bahwa alamat tersebut rentan terhadap serangan *Cross-Site Scripting (XSS)*. Setelah melakukan pengujian manual terhadap alamat *website* tersebut, terbukti bahwa alamat *website* “<http://widyagamahusada.ac.id>” rentan terhadap serangan *Cross-Site Scripting (XSS)*. Pembuktian tersebut dapat dilihat pada Gambar 4.16. Terlihat jelas bahwa *script* yang disisipkan pada tautan alamat berjalan dengan sempurna.



Gambar 4.16 Hasil Serangan *Cross-Site Scripting (XSS)*

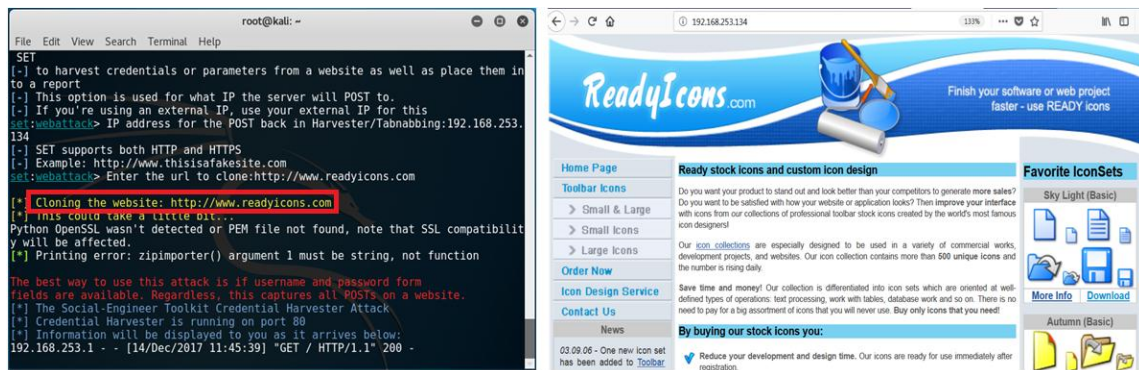
4.2.4 Perbandingan terhadap Aplikasi Sejenis

Pada tahap ini, aplikasi akan dibandingkan dengan aplikasi sejenis yang telah ada sebelumnya, yaitu *SqlMap* dan *SET (Social Engineer Toolkit)*. Perbandingan ini dilakukan untuk mengetahui kelebihan dan kekurangan dari Aplikasi Pengujian Celah Keamanan pada Aplikasi Berbasis *Web*. Walaupun algoritma yang digunakan oleh kedua aplikasi tersebut berbeda dengan algoritma yang digunakan oleh Aplikasi Pengujian Celah Keamanan pada Aplikasi Berbasis *Web*, namun secara garis besar kedua aplikasi tersebut memiliki fungsi

untuk mencari informasi apakah alamat *website* yang dimasukkan oleh pengguna rentan terhadap serangan yang dimiliki masing-masing aplikasi atau tidak.

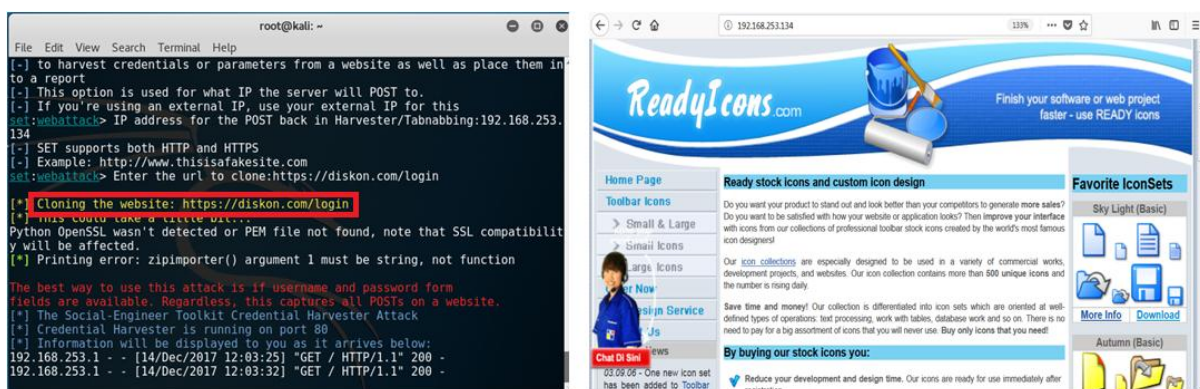
SqlMap merupakan salah satu *tools* yang sering digunakan oleh para *hacker* untuk mencari celah keamanan pada sebuah sistem. Pada dasarnya *tools* ini menggunakan teknik *SQL Injection* untuk menyerang targetnya. *Tools* ini memiliki beberapa kelebihan, diantaranya *tools* ini dapat melakukan *SQL Injection* dengan beberapa metode, seperti *Union Query SQL Injection*, *Error-Based SQL Injection*, *Boolean-Based SQL Injection*, dan *Time-Based SQL Injection*. Selain itu, *tools* ini juga dapat berjalan untuk beberapa jenis *database*, seperti *MySQL*, *Oracle*, *PostgreSQL*, dan masih banyak lagi. Kelebihan tersebut tentunya tidak semuanya dimiliki oleh Aplikasi Pengujian Celah Keamanan pada Aplikasi Berbasis *Web* karena aplikasi tersebut hanya menggunakan metode *Error-Based SQL Injection* pada proses penyerangannya. Namun, kelebihan Aplikasi Pengujian Celah Keamanan pada Aplikasi Berbasis *Web* yang belum tentu dimiliki oleh *SqlMap* adalah aplikasi tersebut dapat menerima target dari pengguna berupa alamat utama sebuah *website*. Tidak seperti *SqlMap* yang mengharuskan pengguna memasukkan alamat target secara spesifik (mengandung kata “*id=*”).

SET (Social Engineer Toolkit) merupakan sebuah *tools* pengujian penetrasi seputar *Social-Engineering*. *Tools* ini bersifat terbuka (*open source*) dan dibuat menggunakan bahasa pemrograman *python*. Selain itu *tools* ini telah menjadi standar untuk melakukan sebuah pengujian penetrasi seputar *Social-Engineering*. Salah satu yang dapat dilakukan oleh *tools* ini adalah melakukan duplikasi terhadap sebuah *website* atau biasa disebut dengan *Phising*. Proses pada *tools* ini sangat cepat dalam hal menduplikasi halaman sebuah *website*. Penggunaan *tools* ini pun terbilang cukup mudah karena pengguna tinggal memilih jenis pengujian, lalu memberikan *ip address* untuk memberikan alamat pada *website* yang telah terduplikasi nantinya, dan terakhir pengguna tinggal memasukkan alamat *website* yang akan diduplikasi. Dapat dilihat pada Gambar 4.17 *tools* tersebut berhasil menduplikasi halaman awal alamat *website* “*http://readyicons.com*” dengan sempurna. Berbeda dengan Aplikasi Pengujian Celah Keamanan pada Aplikasi Berbasis *Web* yang hanya dapat menduplikasi halaman *login* pada sebuah aplikasi *web*.



Gambar 4.17 Duplikasi Halaman *http://readyicons.com* menggunakan SET

Meskipun *tools SET* ini mampu menduplikasi halaman *website* apa saja, namun *tool* ini juga memiliki kekurangan yaitu pengguna harus keluar dari *terminal* dan masuk ulang lagi jika ingin mengganti halaman duplikasi. Apabila pengguna memberhentikan proses duplikasi dengan menekan tombol pada *keyboard* (misalnya *Ctrl + C*) dan mencoba memasukkan halaman baru yang ingin diduplikasi, maka halaman duplikasi yang muncul di dalam *browser* adalah halaman yang telah diduplikasi pertama kali (awal). Sebagai bukti, pertama-tama penulis melakukan duplikasi terhadap halaman awal pada *website* “*http://readyicons.com*” seperti yang terlihat pada Gambar 4.17. Selanjutnya, penulis menekan tombol *Ctrl + C* pada *keyboard* (papan ketik) untuk memasukkan alamat baru yang ingin diduplikasi. Alamat baru yang ingin diduplikasi adalah “*https://diskon.com/login*”. Setelah penulis memasukkan alamat baru tersebut dan kembali menjalankan proses duplikasi, yang terjadi adalah halaman yang terbuka pada *browser* masih halaman “*http://readyicons.com*” bukan halaman “*https://diskon.com/login*”. Kesalahan tersebut dapat dilihat pada Gambar 4.18



Gambar 4.18 Kesalahan pada *Tools SET*

Aplikasi Pengujian Celah Keamanan pada Aplikasi Berbasis *Web* tentunya mampu menangani kekurangan pada *tools SET* tersebut. Hal tersebut dikarenakan proses duplikasi pada Aplikasi Pengujian Celah Keamanan pada Aplikasi Berbasis *Web* akan mengubah kode sebelumnya yang ada pada *file index.html*. Akan tetapi perlu diketahui bahwa Aplikasi Pengujian Celah Keamanan pada Aplikasi Berbasis *Web* ini hanya mampu melakukan duplikasi terhadap aplikasi *website* yang memiliki halaman *login*.

Untuk mengetahui perbandingan Aplikasi Pengujian Celah Keamanan pada Aplikasi Berbasis *Web* terhadap aplikasi sejenis lebih rinci, maka pengguna lain akan dilibatkan secara langsung untuk menggunakan ketiga aplikasi tersebut dan mengisi kuisioner yang telah disediakan. Kuisioner ini mengambil sampel sebanyak 5 responden (pengguna lain). Setiap pengguna nantinya akan diminta untuk memberikan respon dari 9 pernyataan yang terdapat pada kuisioner. Rincian tabel hasil perhitungan kuisioner dapat dilihat pada Tabel 4.3

Tabel 4.3 Tabel Hasil Perhitungan Kuisioner

No	Pernyataan	Sangat Setuju	Setuju	Cukup	Tidak Setuju	Presentasi
1.	Aplikasi Pengujian Celah Keamanan pada Aplikasi Berbasis <i>Web</i> mudah digunakan.	2	3	-	-	88%
2.	<i>SqlMap</i> mudah digunakan.	-	-	3	2	52%
3.	<i>SET</i> mudah digunakan.	-	-	1	4	44%
4.	Aplikasi Pengujian Celah Keamanan pada Aplikasi Berbasis <i>Web</i> membutuhkan waktu yang lama dalam melakukan pengujian.	-	-	2	3	48%
5.	<i>SqlMap</i> membutuhkan waktu yang lama dalam melakukan pengujian.	1	2	2	-	76%
6.	<i>SET</i> membutuhkan waktu yang lama dalam melakukan pengujian.	-	1	3	1	60%
7.	Kemampuan Aplikasi Pengujian Celah Keamanan pada Aplikasi Berbasis <i>Web</i> dalam melakukan pengujian sangat baik.	-	2	3	-	56%
8.	Kemampuan <i>SqlMap</i> dalam melakukan pengujian sangat baik.	-	4	1	-	76%
9.	Kemampuan <i>SET</i> dalam melakukan pengujian sangat baik.	2	3	-	-	88%

Dari hasil perhitungan kuisioner, kesimpulan hasil perbandingan dari ketiga aplikasi, yaitu Aplikasi Pengujian Celah Keamanan pada Aplikasi Berbasis Web lebih mudah digunakan dibandingkan *tools SqlMap* dan *SET* dengan presentasi sebesar 88%. Selanjutnya, *tools SqlMap* membutuhkan waktu paling lama dalam melakukan pengujian dibandingkan *tools SET* dan Aplikasi Pengujian Celah Keamanan pada Aplikasi Berbasis Web. Dan *tools SET* memiliki kemampuan yang paling baik dalam melakukan pengujian dibandingkan *tools SqlMap* dan Aplikasi Pengujian Celah Keamanan pada Aplikasi Berbasis Web.

4.2.5 Pengujian terhadap beberapa Jenis Website

Pengujian terhadap beberapa jenis merupakan tahap pengujian untuk membuktikan apakah Aplikasi Pengujian Celah Keamanan pada Aplikasi Berbasis Web dapat digunakan untuk menguji segala jenis *website* (bersifat *universal*) atau tidak. Jenis-jenis *website* yang akan diuji, diantaranya *website* yang dibangun dengan struktur *HTML* dan *PHP* biasa (*website* umum), *website* yang dibangun dengan menggunakan *CMS* (*Content Management System*) *Wordpress*, dan *website* yang dibangun dengan menggunakan *framework* terkini seperti *Laravel* dan *CodeIgniter*.

Perbedaan dasar dari ketiga jenis *website* tersebut dapat dilihat dari pemanggilan terhadap *source* gambar, *source css*, dan *hyperlink*. *Website* yang menggunakan struktur *HTML* dan *PHP* biasa, pada umumnya memanggil *source* gambar dan *css* langsung dari folder gambar dan *css* itu sendiri atau menambahkan tanda titik 2 kali (..) diawal, seperti “*img/logo.png*” atau “*../css/style.css*”. Selain itu, alamat *URL* pada *website* yang menggunakan struktur *HTML* dan *PHP* biasa identik dengan kata “*id=*” diakhir alamat.

Lain halnya dengan *website* yang dibangun dengan menggunakan *CMS*. *Website* yang dibangun dengan menggunakan *CMS* pada umumnya memiliki kata “*wp*” pada alamat *source* yang digunakan, seperti “*www.abc.xyz/wp-content/image/logo.png*”. Sedangkan, untuk mengetahui *website* yang dibangun dengan menggunakan *framework Laravel* atau *CodeIgniter* adalah dengan melihat *source* gambar dan *hyperlink* yang digunakan. *Source* gambar dan *hyperlink* yang digunakan pada umumnya diawali dengan *base path*. *Base path* ini merupakan *path* yang namanya mirip seperti alamat *URL*. Contoh *source* gambar pada *framework Laravel* adalah “*https://www.abc.xyz/storage/product-variants/logo.jpeg*”.

Website yang akan dijadikan target pada pengujian ini, diantaranya “*http://www.f10products.co.za*” yang dibangun menggunakan *HTML* dan *PHP* biasa (*website* umum), “*http://www.tobiweb.id/*” yang dibangun menggunakan *Framework CodeIgniter*, dan

“<https://www.needrom.com/>” yang dibangun menggunakan *CMS Wordpress*. Untuk hasil pengujian terhadap beberapa jenis *website* dapat dilihat pada Tabel 4.4.

Tabel 4.4 Hasil Pengujian terhadap beberapa Jenis *Website*

No	Alamat <i>Website</i>	Jenis <i>Website</i>	Kerentanan
1.	http://www.f10products.co.za	<i>Website umum</i>	Rentan terhadap serangan <i>SQL Injection</i>
2.	https://www.needrom.com/	<i>Website CMS</i>	Rentan terhadap serangan <i>Phising</i>
3.	http://www.tobiweb.id/	<i>Website Framework</i>	Tidak rentan terhadap semua serangan

Dari hasil pengujian diatas, Aplikasi Pengujian Celah Keamanan pada Aplikasi Berbasis *Web* menemukan celah pada “<http://www.f10products.co.za>” . Celah tersebut dapat ditemukan karena *website* tersebut masih menggunakan *HTML* dan *PHP* biasa dalam pembangunan struktur *web* nya. Dengan struktur *HTML* dan *PHP* biasa, Aplikasi Pengujian Celah Keamanan pada Aplikasi Berbasis *Web* mampu dengan mudah mendapatkan alamat yang mengandung kata “*id=*”. Berbeda dengan “<https://www.needrom.com/>” yang dibangun dengan *CMS Wordpress*. Kerentanan yang ditemukan pada *website* tersebut adalah kerentanan terhadap serangan *Phising*. Hal tersebut dapat terjadi karena *source* gambar yang digunakan halaman *login* pada *website* yang dibangun dengan *CMS Wordpress* dapat diakses publik, sehingga memudahkan orang lain untuk menduplikasi halaman tersebut.

Dan pada “<http://www.tobiweb.id/>” tidak ditemukan kerentanan dari Aplikasi Pengujian Celah Keamanan pada Aplikasi Berbasis *Web*. Hal tersebut dapat terjadi karena dengan menggunakan struktur *framework* (baik *Laravel*, maupun *CodeIgniter*) dalam membangun sebuah *website*, *developer* (pengembang) dapat mengatur alamat *link* agar kata “*id=*” dapat disembunyikan. Selain itu, pengembang juga memiliki hak penuh dalam mengatur keamanan dari *website* tersebut. Jadi, Aplikasi Pengujian Celah Keamanan pada Aplikasi Berbasis *Web* dapat digunakan untuk melakukan pengujian terhadap *website* yang dibangun dengan *HTML* biasa dan *website* yang dibangun dengan *CMS Wordpress*.

BAB V

KESIMPULAN DAN SARAN

5.1 Kesimpulan

Setelah melakukan penelitian dan implementasi, hasil yang didapatkan adalah Aplikasi Pengujian Celah Keamanan pada Aplikasi Berbasis *Web* dapat membuat pengujian terhadap aplikasi *website* menjadi lebih efisien dan efektif. Hal tersebut juga diperkuat oleh hasil dari pengujian yang dilakukan oleh pengguna lain. Namun, Aplikasi Pengujian Celah Keamanan pada Aplikasi Berbasis *Web* hanya dapat digunakan untuk melakukan pengujian terhadap *website* yang dibangun dengan *HTML* biasa dan *website* yang dibangun dengan *CMS Wordpress*. Hal tersebut dikarenakan dengan struktur *HTML* dan *PHP* biasa, Aplikasi Pengujian Celah Keamanan pada Aplikasi Berbasis *Web* mampu dengan mudah mendapatkan alamat yang mengandung kata “*id=*” dan struktur *website* yang dibangun dengan *CMS Wordpress* menggunakan *source* gambar yang dapat diakses publik, sehingga memudahkan orang lain untuk menduplikasi halaman tersebut.

5.2 Saran

Dari Aplikasi Pengujian Celah Keamanan pada Aplikasi Berbasis *Web* yang telah dibuat, masih terdapat beberapa kekurangan dan kelemahan yang dapat dikembangkan lebih lanjut. Hal yang dapat dikembangkan dari Aplikasi Pengujian Celah Keamanan pada Aplikasi Berbasis *Web* yaitu :

- a. Pengembangan dari aplikasi ini diharapkan menggunakan bahasa pemrograman atau *library* yang dapat menampilkan kode atau informasi *error* pada halaman *GUI* aplikasi.
- b. Pengembangan dari aplikasi ini diharapkan menggunakan bahasa pemrograman atau *library* yang dapat menampilkan proses yang berjalan secara *real-time* pada halaman *GUI* aplikasi.
- c. Pengembangan pada teknik *web crawling* yang tidak hanya dapat melakukan *crawling* pada halaman awal dari sebuah *website*, melainkan mampu melakukan *crawling* secara mendalam pada sebuah alamat *website*.
- d. Pengembangan dari aplikasi ini juga diharapkan dapat menambahkan jenis serangan untuk melakukan pengujian pada aplikasi berbasis *web*.

DAFTAR PUSTAKA

- Budiman, E., & Jati, hario. (2017). Teknik Hacking Web Server dengan SQLMap di Kali Linux.pdf. 2017.
- detikinet. (2014). Xnuxer, “Hacker Partai Jambu” Situs KPU. Diambil 15 Desember 2017, dari <https://inet.detik.com/cyberlife/d-2643201/xnuxer-hacker-partai-jambu-situs-kpu>
- Dharmawan, Eka Adhitya , Erni Yudaningtyas, M. S. (2013). Perlindungan Web pada Login Sistem Menggunakan Algoritma Rijndael. *Universitas Brawijaya*, 7(1), 77–84.
- Elu, A. M. (2013). (SQL) Injection untuk Keamanan Website, 111–124.
- Fernandez, D., Arteaga, J., & Caltum, E. (2016). Akamai’s State of Internet / Security Q2 2016 Report.
- Gesita, N. (2013). Kasus Penipuan Dominasi Kejahatan Cyber - Kompas.com. Diambil 27 Agustus 2017, dari [http://tekno.kompas.com/read/2013/04/15/22095149/~Megapolitan~Crime Story](http://tekno.kompas.com/read/2013/04/15/22095149/~Megapolitan~Crime%20Story)
- Irwan, S., & Arif, N. (2014). Investigasi Web Attack Menggunakan Intrusion Detection System (IDS) dan Access Log, 5–17.
- Kiezun, A., Guo, P. J., Jayaraman, K., & Ernst, M. D. (2009). Automatic creation of SQL injection and cross-site scripting attacks. *Proceedings - International Conference on Software Engineering*, 199–209. <https://doi.org/10.1109/ICSE.2009.5070521>
- Meucci, M., & Muller, A. (2014). Testing Guide 4.0, (Cc).
- Mustaqbal, M. S., Firdaus, R. F., & Rahmadi, H. (2015). Pengujian Aplikasi Menggunakan Black Box Testing Boundary Value Analysis (Studi Kasus : Aplikasi Prediksi Kelulusan SNMPTN). *Jurnal Ilmiah Teknologi Informasi Terapan*, 1(3), 31–36. Diambil dari <http://jitter.widyatama.ac.id/index.php/jitter/article/view/70>
- Nurrahman, Y. (2015). Cyber Crime, Lebih dari Rp 33 M Melayang Gara-gara Hacker - Teknologi Tempo.co. Diambil 27 Agustus 2017, dari <https://tekno.tempo.co/read/695105/cyber-crime-lebih-dari-rp-33-m-melayang-gara-gara-hacker>
- Oktaviani, D. M. (2015). Sistem Rekomendasi Penyewaan Sound System Pada Ud . Dyah Audio Berbasis Web Menggunakan Metode Euclidean Distance. *Artikel Skripsi*, 1–15.
- Owasp. (2013). OWASP Top 10 - 2013. *OWASP Top 10*, 22. <https://doi.org/10.1109/OWASP.2013.6616171>
- Owasp. (2017). OWASP Top 10 - 2017.
- Pangalila, R., & Andjarwirawan, J. (2015). Penetration Testing Server Sistem Informasi

Manajemen dan Website Universitas Kristen Petra.

Rachmawati. (2014). Phising sebagai salah satu bentuk ancaman dalam dunia cyber, 209–216.

Saputra Ahad, D., & Akbar, M. (2015). Serangan Pada Website Pdam Tirta Musi.

Siagian, H. P., & Akbar, M. (2011). Vulnerability Assessment Pada Web Server Wwww.Binadarma.Ac.Id, (12).

Siregar, J. J. (2013). Web Denial of Service Attack, 4(9), 1199–1205.

Srivastava, T. V. (2007). The Evil Twins Phishing and Pharming – The Deadly Duo.

Tolle, H., Kurniawan, T. A., & Zakaria, A. (2008). Peningkatan Keamanan Web terhadap Serangan Cross-Site Scripting (XSS).

Yadi, I. Z., & Herdiansyah, M. I. (2012). Analisis Web Vulnerability Pada Digital Library, (12). <https://doi.org/Ilman Zuhri Yadi1 , Muhammad Izman Herdiansyah2>

LAMPIRAN

No.	Kode Program
1.	import sys
2.	from bs4 import BeautifulSoup
3.	from urllib.parse import *
4.	import urllib.request # library untuk HTTP request
5.	import urllib.parse # library untuk memberikan balasan kepada client
6.	from urllib.error import * # library untuk penanganan Error
7.	import requests
8.	import re
9.	import http.client
10.	from PyQt5.QtWidgets import *
11.	from PyQt5.QtGui import *
12.	from time import *
13.	from datetime import *
14.	from webbrowser import open_new_tab
15.	import os.path
16.	
17.	class Aplikasi(QMainWindow):
18.	def __init__(self):
19.	super().__init__()
20.	self.setWindowTitle('Aplikasi Penguji Celah Keamanan') # Mengatur
21.	judul window
22.	self.interface()
23.	
24.	def interface(self):
25.	self.setGeometry(320, 50, 750, 650) # Mengatur ukuran layar (posisi
26.	x, y, lebar, panjang)
27.	self.labelJudul = QLabel('Aplikasi Pengujian Celah Keamanan pada
28.	Aplikasi Berbasis Web', self) # Membuat label
29.	self.label = QLabel('Masukkan URL / ip address target :',self)
30.	self.pemberitahuanSave = QLabel('Masukkan nama file dan destinasi
31.	folder untuk menyimpan log pengujian (Selain kata \'index\'):', self)
32.	self.url = QLineEdit(self) # membuat input box
33.	self.attack_btn = QPushButton('Attack !', self) # membuat button
34.	attack
35.	self.exit_btn = QPushButton('Exit', self) # membuat button
36.	exit
37.	self.pemberitahuan = QLabel(self)
38.	self.NotifRentan = QLabel(self)
39.	self.destination_btn = QPushButton('Select Folder', self)
40.	self.namaFile = QLineEdit(self)
41.	self.dest = QLineEdit(self)
42.	
43.	# membuat text area
44.	self.textArea = QTextEdit(self)
45.	self.textArea.setText("")
46.	self.textArea.setDisabled(False)
47.	self.textArea.setGeometry(10, 320, 730, 320)
48.	
49.	
50.	self.url.setFixedWidth(350) # mengatur ukuran lebar form
51.	self.namaFile.setFixedWidth(300)
52.	self.pemberitahuan.setText('(alamat website harus lengkap dengan
53.	http:// atau https://)')
54.	
55.	self.labelJudul.setGeometry(145, 10, 430, 16) # Mengatur posisi x,y
56.	dan panjang lebar labelJudul
57.	self.label.setGeometry(10, 70, 211, 16)
58.	self.dest.move(430, 190)
59.	self.dest.setFixedWidth(300)
60.	self.dest.setDisabled(True)
61.	self.url.move(10, 100)
62.	self.namaFile.move(10,190)

```

63.         self.pemberitahuan.setGeometry(370, 105, 480, 16)
64.         self.pemberitahuanSave.setGeometry(10,160,550,16)
65.         self.NotifRentan.setGeometry(10, 180, 480, 16)
66.         self.attack_btn.move(400, 260)
67.         self.exit_btn.move(230, 260)
68.         self.destination_btn.move(320,190)
69.         self.show()
70.
71.         # membuat tombol Attack ! disable ketika aplikasi pertama kali
72.         dijalankan
73.         self.attack_btn.setDisabled(True)
74.
75.         self.url.textChanged.connect(self.disableButton)
76.         self.namaFile.textChanged.connect(self.disableButton)
77.         self.dest.textChanged.connect(self.disableButton)
78.
79.         self.attack_btn.clicked.connect(self.btn_clk)
80.         self.exit_btn.clicked.connect(self.exit_clk)
81.         self.destination_btn.clicked.connect(self.dest_clk)
82.
83.         def disableButton(self):                                # fungsi untuk disable
84.         button
85.             if len(self.url.text() and self.namaFile.text() and self.dest.text())
86.             > 0:                                             # kalo ada string didalam line edit, maka...
87.                 self.attack_btn.setDisabled(False)         # tombol attack tidak
88.         disable
89.             else:
90.                 self.attack_btn.setDisabled(True)
91.         def exit_clk(self):
92.             sys.exit()
93.
94.         def dest_clk(self):
95.             self.folderPath = QFileDialog.getExistingDirectory(self, 'Select
96. Folder')
97.             self.dest.setText(self.folderPath)
98.             # self.filename = QFileDialog.getSaveFileName(self.folderPath)
99.
100.        def btn_clk(self):
101.            self.textArea.clear()
102.            alamatURL = self.url.text()
103.            lokasiPath = self.dest.text()
104.            saveFileLog = os.path.join(lokasiPath, self.namaFile.text()+'.html')
105.            print(saveFileLog)
106.            try:
107.                redirect = requests.get(alamatURL, timeout=5) # melakukan
108.        request terhadap URL
109.                URLAsli = redirect.url # mengambil URL yang telah di-redirect /
110.        history
111.                Nomor = 1
112.                buatLog = open(saveFileLog, "w")
113.                buatLog.write("<html>\n"
114.                             "<head>\n<title>Report</title>\n"
115.                             "<style>\n"
116.                             "table, th, td {\n"
117.                             "border: 1px solid black;\n"
118.                             "border-collapse: collapse;\n"
119.                             "}\n"
120.                             "</style>\n"
121.                             "</head>\n"
122.                             "<body>\n"
123.                             "<h2 style=\"text-align:center;\"> Log Pengujian
124.        </h2>\n")
125.                print(URLAsli)
126.                self.textArea.insertPlainText("Alamat website : "+ URLAsli +
127.        '\n\n')
128.                buatLog.write("<h3> Alamat website : " + URLAsli + "</h3>\n\n")
129.                buatLog.write("<table style = \"width:100%;\">\n"

```

```

130.         "<tr>\n"
131.         "<th>No.</th>\n"
132.         "<th>Tanggal Pengujian</th>\n"
133.         "<th>Alamat Target</th>\n"
134.         "<th>Keterangan</th>\n"
135.         "</tr>\n")
136.     URLSql = []
137.     URLXss = []
138.     URLPhising = []
139.     srcPhising = []
140.     resultPhising = []
141.
142.     response = urllib.request.urlopen(URLAsli).read()
143.     # print(response.read().decode('utf-8'))
144.
145.     # --Proses serangan Phising--
146.     soupPhising = BeautifulSoup(response, 'html5lib')
147.     kondisiPhising = soupPhising.find('a', href=re.compile('login'))
148.     # print(kondisiPhising)
149.     if len(kondisiPhising) > 0:
150.         for b in kondisiPhising:
151.             URLPhising.append(b['href'])
152.             print(URLPhising)
153.     else:
154.         print(datetime.now().strftime("%d-%m-%Y %H:%M:%S") + " " +
155. URLAsli, "tidak rentan terhadap serangan Phising")
156.         self.textArea.insertPlainText(datetime.now().strftime("%d-%m-
157. %Y %H:%M:%S") + " " + URLAsli + " tidak rentan terhadap serangan Phising\n")
158.         # buatLog.write(datetime.now().strftime("%d-%m-%Y %H:%M:%S")
159. + " " + URLAsli + " tidak rentan terhadap serangan Phising\n")
160.         buatLog.write("<tr>\n")
161.         buatLog.write("<td>" + str(Nomor) + ". " + "</td>\n")
162.         buatLog.write("<td>" + datetime.now().strftime("%d-%m-%Y
163. %H:%M:%S") + "</td>\n")
164.         buatLog.write("<td>" + "<a href='" + URLAsli + "'>" + URLAsli
165. + "</td>\n")
166.         buatLog.write("<td>" + "Tidak rentan terhadap serangan
167. Phising" + "</td>\n")
168.         buatLog.write("</tr>\n")
169.         Nomor = Nomor + 1
170.         intPhising = 0
171.         tdkPhising = []
172.         for phising in URLPhising:
173.             try:
174.                 responsePhising =
175. urllib.request.urlopen(URLPhising[intPhising]).read()
176.                 soupCopasPhising = BeautifulSoup(responsePhising,
177. 'html5lib')
178.                 kondisicekPhising = soupCopasPhising.find_all('img',
179. {"src": True})
180.                 # print(kondisicekPhising)
181.                 if len(kondisicekPhising) > 0:
182.                     letter = ["http"]
183.                     x = 0
184.                     for c in kondisicekPhising:
185.                         srcPhising.append(c['src'])
186.                         first = srcPhising[x][:4]
187.                         if first in letter != -1 :
188.                             resultPhising.append("rentan")
189.                         else:
190.                             tdkPhising.append("a")
191.                             x = x + 1
192.                             print(srcPhising)
193.                             print(len(resultPhising))
194.                             print(len(tdkPhising))
195.                 if len(resultPhising) > 0:
196.                     kondisiCopasPhising =

```

```

197. soupCopasPhising.find_all('html')
198.         saveFileHTML = os.path.join(lokasiPath, 'index.html')
199.         filePhising = open(saveFileHTML, "w", encoding='utf-
200. 8')
201.         filePhising.write(str(kondisiCopasPhising))
202.         filePhising.close()
203.         print(datetime.now().strftime("%d-%m-%Y %H:%M:%S") +
204. " " + URLAsli,
205.                 "rentan terhadap serangan Phising")
206.
207. self.textArea.insertPlainText(datetime.now().strftime(
208.         "%d-%m-%Y %H:%M:%S") + " " + URLAsli + " rentan
209. terhadap serangan Phising\n")
210.         # buatLog.write(datetime.now().strftime("%d-%m-%Y
211. %H:%M:%S") + " " + URLAsli + " rentan terhadap serangan Phising\n")
212.         buatLog.write("<tr>\n")
213.         buatLog.write("<td>" + str(Nomor) + ". " + "</td>\n")
214.         buatLog.write("<td>" + datetime.now().strftime("%d-
215. %m-%Y %H:%M:%S") + "</td>\n")
216.         buatLog.write("<td>" + "<a href='" + URLAsli + "'>"
217. + URLAsli + "</td>\n")
218.         buatLog.write("<td>" + "Rentan terhadap serangan
219. Phising" + "</td>\n")
220.         buatLog.write("</tr>\n")
221.         Nomor = Nomor + 1
222.     else:
223.         print(datetime.now().strftime("%d-%m-%Y %H:%M:%S") +
224. " " + URLAsli, "tidak rentan terhadap serangan Phising !")
225.
226. self.textArea.insertPlainText(datetime.now().strftime("%d-%m-%Y %H:%M:%S") +
227. " " + URLAsli + " tidak rentan terhadap serangan Phising\n")
228.         # buatLog.write(datetime.now().strftime("%d-%m-%Y
229. %H:%M:%S") + " " + URLAsli + " tidak rentan terhadap serangan Phising\n")
230.         buatLog.write("<tr>\n")
231.         buatLog.write("<td>" + str(Nomor) + ". " + "</td>\n")
232.         buatLog.write("<td>" + datetime.now().strftime("%d-
233. %m-%Y %H:%M:%S") + "</td>\n")
234.         buatLog.write("<td>" + "<a href='" + URLAsli + "'>"
235. + URLAsli + "</td>\n")
236.         buatLog.write("<td>" + "Tidak rentan terhadap
237. serangan Phising" + "</td>\n")
238.         buatLog.write("</tr>\n")
239.         Nomor = Nomor + 1
240.         intPhising = intPhising + 1
241.     except (ValueError, URLError) as e:
242.         print(datetime.now().strftime("%d-%m-%Y %H:%M:%S") + " "
243. + URLAsli, "tidak rentan terhadap serangan Phising")
244.         # -- Proses Serangan Phising end--
245.
246.         soup = BeautifulSoup(response, 'html5lib')
247.         kondisi = soup.find_all('a', href=re.compile('id=')) # mencari
248. tag <a> yang memiliki atribut "href". Kemudian mencari value href yang
249. memiliki kalimat "id=" (Menggunakan regex / regular expression)
250.         # print(kondisi)
251.
252.         if len(kondisi) > 0:
253.             y = urlparse(URLAsli)
254.             URLFix = y.scheme + "://" + y.netloc + y.path # mengatur URL
255. (menghilangkan query untuk case "atmaine.fi"
256.             print(URLFix)
257.             for a in kondisi:
258.                 URLSql.append(URLFix + a['href'])
259.                 URLXss.append(URLFix + a['href'])
260.
261.             # --Proses serangan SQL Injection--
262.             intSQL = 0
263.             tdkSQL = []

```

```

264.         for sql in URLSql:
265.             try:
266.                 responseSQL = urllib.request.urlopen(URLSql[intSQL] +
267. "%27").read(25373)
268.                 soupSQL = BeautifulSoup(responseSQL, 'html5lib')
269.                 kondisiSQL = soupSQL.find_all('body',
270. string=re.compile('SQL syntax;'))
271.                 print(kondisiSQL)
272.                 if len(kondisiSQL) > 0:
273.                     print(datetime.now().strftime("%d-%m-%Y %H:%M:%S") +
274. " " + URLSql[intSQL] + " rentan terhadap serangan SQL Injection")
275.
276. self.textArea.insertPlainText(datetime.now().strftime("%d-%m-%Y %H:%M:%S") +
277. " " + URLSql[intSQL] + " rentan terhadap serangan SQL Injection\n")
278.                 # print(URLSql)
279.                 buatLog.write("<tr>\n")
280.                 buatLog.write("<td>" + str(Nomor) + ". " + "</td>\n")
281.                 buatLog.write("<td>" + datetime.now().strftime("%d-
282. %m-%Y %H:%M:%S") + "</td>\n")
283.                 buatLog.write("<td>" + "<a href=\"'" + URLSql[intSQL]
284. + "\"'>" + URLSql[intSQL] + "</td>\n")
285.                 buatLog.write("<td>" + "Rentan terhadap serangan SQL
286. Injection" + "</td>\n")
287.                 buatLog.write("</tr>\n")
288.                 Nomor = Nomor + 1
289.                 tdkSQL.append("a")
290.                 intSQL = intSQL + 1
291.             except HTTPError as e:
292.                 print(datetime.now().strftime("%d-%m-%Y %H:%M:%S") + " "
293. + URLSql[intSQL], ":", e.code, e.reason)
294.                 # self.text.insertPlainText(datetime.now().strftime("%d-
295. %m-%Y %H:%M:%S") + " " + URLSql[intSQL] + " : " + e.code + e.reason)
296.                 intSQL = intSQL + 1
297.             except http.client.IncompleteRead as e:
298.                 responseSQL = e.partial
299.                 if len(tdkSQL) == 0:
300.                     print(datetime.now().strftime("%d-%m-%Y %H:%M:%S") + " " +
301. URLAsli, "tidak rentan terhadap serangan SQL Injection")
302.                     self.textArea.insertPlainText(datetime.now().strftime("%d-%m-
303. %Y %H:%M:%S") + " " + URLAsli + " tidak rentan terhadap serangan SQL
304. Injection\n")
305.                     buatLog.write("<tr>\n")
306.                     buatLog.write("<td>" + str(Nomor) + ". " + "</td>\n")
307.                     buatLog.write("<td>" + datetime.now().strftime("%d-%m-%Y
308. %H:%M:%S") + "</td>\n")
309.                     buatLog.write("<td>" + "<a href=\"'" + URLAsli + "\"'>" +
310. URLAsli + "</td>\n")
311.                     buatLog.write("<td>" + "Tidak rentan terhadap serangan SQL
312. Injection" + "</td>\n")
313.                     buatLog.write("</tr>\n")
314.                     Nomor = Nomor + 1
315.                 # --Proses serangan SQL Injection end--
316.
317.                 # --Proses serangan XSS--
318.                 intXSS = 0
319.                 tdkXSS = []
320.                 for xss in URLXss:
321.                     try:
322.                         responseXSS = urllib.request.urlopen(URLXss[intXSS] +
323. "%22%3E%3Cscript%3Ealert%28222%29%3C%2Fscript%3E").read()
324.                         soupXSS = BeautifulSoup(responseXSS, 'html5lib')
325.                         kondisiXSS = soupXSS.find_all('script',
326. string=re.compile('alert'))
327.                         print(kondisiXSS)
328.                         if len(kondisiXSS) > 0:
329.                             print(datetime.now().strftime("%d-%m-%Y %H:%M:%S") +
330. " " + URLXss[intXSS] + " rentan terhadap serangan Cross-Site Scripting")

```

```

331. self.textArea.insertPlainText(datetime.now().strftime("%d-%m-%Y %H:%M:%S") +
332. " " + URLXss[intXSS] + " rentan terhadap serangan Cross-Site Scripting\n")
333.         buatLog.write("<tr>\n")
334.         buatLog.write("<td>" + str(Nomor) + ". " + "</td>\n")
335.         buatLog.write("<td>" + datetime.now().strftime("%d-
336. %m-%Y %H:%M:%S") + "</td>\n")
337.         buatLog.write("<td>" + "<a href='" + URLSql[intXSS]
338. + "'>" + URLSql[intXSS] + "</td>\n")
339.         buatLog.write("<td>" + "Rentan terhadap serangan
340. Cross-Site Scripting (XSS)" + "</td>\n")
341.         buatLog.write("</tr>\n")
342.         Nomor = Nomor + 1
343.         tdkXSS.append("tidak")
344.         intXSS = intXSS + 1
345.     except HTTPError as e:
346.         print(datetime.now().strftime("%d-%m-%Y %H:%M:%S") + " "
347. + URLXss[intXSS], ":", e.code, e.reason)
348.         intXSS = intXSS + 1
349.     except http.client.IncompleteRead as e:
350.         responseXSS = e.partial
351.         if len(tdkXSS) == 0:
352.             print(datetime.now().strftime("%d-%m-%Y %H:%M:%S") + " " +
353. URLAsli, "tidak rentan terhadap serangan Cross-Site Scripting")
354.             self.textArea.insertPlainText(datetime.now().strftime("%d-%m-
355. %Y %H:%M:%S") + " " + URLAsli + " tidak rentan terhadap serangan Cross-Site
356. Scripting\n")
357.             buatLog.write("<tr>\n")
358.             buatLog.write("<td>" + str(Nomor) + ". " + "</td>\n")
359.             buatLog.write("<td>" + datetime.now().strftime("%d-%m-%Y
360. %H:%M:%S") + "</td>\n")
361.             buatLog.write("<td>" + "<a href='" + URLAsli + "'>" +
362. URLAsli + "</td>\n")
363.             buatLog.write("<td>" + "Tidak rentan terhadap serangan Cross-
364. Site Scripting (XSS)" + "</td>\n")
365.             buatLog.write("</tr>\n")
366.             buatLog.write("</table>\n")
367.             "</body>\n")
368.             "</html>\n")
369.             Nomor = Nomor + 1
370.             # Proses serangan XSS end--
371.             self.textArea.insertPlainText("Complete !!")
372.             buatLog.close()
373.         except HTTPError as e:
374.             # self.text.insertPlainText("Error code : " + e.code + e.reason)
375.             print("Error code :", e.code, e.reason)
376.         except URLError as e:
377.             print("Periksa koneksi internet anda : ", e.reason)
378.             # self.text.insertPlainText("Periksa koneksi internet anda : " +
379. e.reason)
380.         except ConnectionError as e:
381.             print("Error : ", e)
382.             # self.text.insertPlainText("Error : " + e)
383.         except Exception as e:
384.             print("The site cannot be reached")
385.
386.
387.
388. app = QApplication(sys.argv)
389. tampilan = Aplikasi()
390. sys.exit(app.exec_())

```