



# Petunjuk Teknis

Pelaksanaan IT Security Assessment (ITSA)  
Aplikasi Berbasis Web





## INFORMASI DOKUMEN

<b>JUDUL</b>	PETUNJUK TEKNIS PELAKSANAAN INFORMATION TECHNOLOGY SECURITY ASSESSMENT (ITSA) APLIKASI BERBASIS WEB
<b>VERSI</b>	v1.0
<b>TANGAL PENGESAHAN</b>	11 Oktober 2021
<b>TIM PENYUSUN</b>	Kelompok Fungsi Operasi Identifikasi dan Proteksi  Direktorat Operasi Keamanan Siber

## DAFTAR ISI

INFORMASI DOKUMEN.....	1
DAFTAR ISI.....	2
DAFTAR TABEL.....	4
DAFTAR GAMBAR.....	5
LATAR BELAKANG.....	7
1. Aplikasi Berbasis WEB.....	8
1.1. Terminologi pada Aplikasi Berbasis Web .....	8
1.2. Arsitektur Teknologi WEB .....	10
2. IT Security Assessment Aplikasi berbasis Web .....	10
2.1. Inspeksi dan Tinjauan Manual.....	11
2.2. Pemodelan Ancaman.....	12
2.3. Peninjauan Kode .....	13
2.4. Uji Penetrasi .....	14
3. Uji Penetrasi Aplikasi berbasis Web .....	15
3.1. Tahapan Pengujian.....	15
3.2. Information Gathering .....	17
3.2.1 Server Discovery .....	17
3.2.2 Search Engine Discovery .....	21
3.2.3 Fingerprint Framework Aplikasi .....	24
3.3. Configuration and Deployment Management .....	26
3.4. Identify Management .....	28
3.5. Authentication .....	32
3.5.1 Mekanisme Autentikasi.....	32
3.5.2 Pengujian Autentikasi .....	39
3.6. Authorization.....	45
3.6.1 Matrik Peran (Role Matrix) .....	47
3.6.2 Pengujian Otorisasi .....	48
3.7. Session Management .....	54
3.7.1 Client-Side Techniques.....	55
3.7.2 Server-Side Techniques .....	58
3.7.3 Pengujian Session Management .....	60

DRAFT PANDUAN TEKNIS  
PELAKSANAAN IT SECURITY ASSESSMENT (ITSA)



3.8	Input Validation .....	61
3.8.1	Mengharapkan yang Tidak Terduka .....	61
3.8.2	Pengujian Input Validation .....	63
3.9	Error Handling.....	65
3.10	Cryptographic .....	68
3.10.1.	Pengujian Cryptographic.....	68
3.11	Business Logic.....	69
PENUTUP .....		70
A.	KESIMPULAN.....	70
B.	SARAN .....	70
LEMBAR PENGESAHAN .....		71
REFERENSI .....		72
Lampiran I : Alat Pengujian ITSA Aplikasi Berbasis Web .....		73
Lampiran II : Pendeteksian dan Menghindari WAF .....		74
Lampiran III : <i>Websites Security Checklist Guide</i> .....		77

## DAFTAR TABEL

Tabel 1 Matrik Peran .....	47
Tabel 2 Tabel Hasil Pelacakan Menggunakan Teknik <i>Client-Side</i> .....	55
Tabel 3 Variable Session .....	59

## DAFTAR GAMBAR

Gambar 1 Contoh URL .....	9
Gambar 2 Arsitektur Komponen Web .....	10
Gambar 3 Tampilan Developer Tools .....	16
Gambar 4 Mode Pasif pada bssn.go.id Menggunakan Developer Tool.....	16
Gambar 5 Kategori Pengujian Uji Penetrasi.....	16
Gambar 6 Informasi IP Server Menggunakan netcraft .....	18
Gambar 7 Hasil Kueri Registrar Domain .....	19
Gambar 8 Hasil Enumerasi Subdomain .....	20
Gambar 9 Hasil Pemindaian Port dan Layanan .....	21
Gambar 10 Pencarian Menggunakan Google .....	23
Gambar 11 Pencarian Informasi Sertifikat SSL.....	24
Gambar 12 Hasil Identifikasi Framework Aplikasi .....	26
Gambar 13 Hasil Vulnerability Assessment menggunakan OpenVas .....	27
Gambar 14 Hasil Uji Kerentanan Menggunakan OWASP ZAP .....	28
Gambar 15 Respon Login Gagal .....	30
Gambar 16 Respon Detail Login Gagal .....	30
Gambar 17 Respon Login Sukses .....	31
Gambar 18 Respon Login Sukses .....	31
Gambar 19 HTTP Basic Authentication .....	33
Gambar 20 Request HTTP Basic Authentication .....	34
Gambar 21 Login Window NTLM Authentication .....	36
Gambar 22 Autentikasi Berbasis Form .....	39
Gambar 23 Hasil Password Guessing .....	40
Gambar 24 Burp Session Sequencer.....	42
Gambar 25 Burp Session ID Intruder .....	42
Gambar 26 Contoh XSS Menampilkan Cookies .....	43
Gambar 27 Burp Proxy Intercept .....	50
Gambar 28 Burp Melakukan Intersepsi Request Client.....	50
Gambar 29 Contoh Hidden Tag pada Source HTML .....	51
Gambar 30 Melakukan Perubahan HTTP <i>Header</i> .....	52

Gambar 31 Hasil Pengujian Directory Traversal .....	54
Gambar 32 Melihat Status Cookies .....	58
Gambar 33 OWASP ZAP Mendapatkan Kerentanan XSS .....	63
Gambar 34 Konfirmasi Kerentanan XSS.....	63
Gambar 35 OWASP ZAP Menemukan Kerentanan SQL Injection .....	64
Gambar 36 Hasil Pengujian Kerentanan SQL Injection.....	64
Gambar 37 OWASP ZAP Mendapatkan Kerentanan Command Injection.....	65
Gambar 38 Percobaan Command Injection Menggunakan Burp .....	65
Gambar 39 Respon Error Web Server .....	67
Gambar 40 Respon Error Aplikasi .....	67
Gambar 41 Respon Error Database.....	67
Gambar 42 Hasil Pengujian SSL/TLS .....	68
Gambar 43 Hasil Identifikasi Jenis WAF .....	75
Gambar 44 Hasil Pencarian Alamat Asli .....	76

## LATAR BELAKANG

Maraknya teknologi yang selalu berkembang berbanding lurus dengan kerentanan pada teknologi tersebut. Sehingga diperlukan sebuah pengamanan untuk menyeimbangkannya. Berbicara mengenai teknologi, saat ini bentuknya sudah berbagai macam yang bisa kita temui. Salah satunya adalah aplikasi berbasis *website*. Hingga saat ini teknologi tersebut masih relevan dan akan terus relevan. Mengingat mesin pencarian yang digunakan berupa *browser*.

Saat ini berbagai macam serangan siber pada *website* memiliki jumlah yang sangat banyak. Hal ini dibuktikan dengan rekapan yang dilakukan oleh BSSN setiap harinya. Oleh karena itu diperlukan upaya dalam rangka mengurangi atau mencegah serangan pada *website* yang memungkinkan terjadi.

Berdasarkan latar belakang tersebut, wujud pengamanan yang bisa dilakukan sebelum adanya serangan yang terjadi yakni dengan melaksanakan *Information Technology Security Assessment* (ITSA). Upaya tersebut mendorong pelayanan publik terkait pengujian kerentanan, pemberian saran dan rekomendasi terkait pengamanan, guna meminimalkan celah kerawanan yang terdapat pada semua sistem informasi pemerintah.

Kegiatan ITSA yang dilakukan tentu membutuhkan dokumen pendukung dalam melaksanakannya. Hal ini diperkuat dengan adanya perbedaan kompetensi dari masing-masing personil. Maka sangat penting dibuat sebuah petunjuk teknis dari Pelaksanaan Kegiatan ITSA *Web*. Tujuannya agar personil yang ada pada unit kerja tersebut dapat dengan mudah belajar dan memahaminya. Sehingga tidak perlu membutuhkan waktu yang lama untuk melakukan adaptasi.



## 1. Aplikasi Berbasis WEB

Web merupakan kumpulan halaman – halaman untuk menginformasikan sesuatu dalam internet yang dapat berisi teks, gambar, animasi, suara, video dan lainnya. Web merupakan suatu ruang informasi di mana sumber-sumber daya yang berguna diidentifikasi oleh pengenal global yang disebut *Uniform Resource Identifier* (URI).

Dalam membangun sebuah aplikasi berbasis web dikenal dengan pemrograman web dikelompokkan menjadi 2 kelompok, yaitu :

- *Client Side* : Informasi yang disampaikan akan langsung dieksekusi browser pada *client*. Contoh: HTML, Javaskrip dan lainnya.
- *Server Side* : Informasi yang dikirim akan dieksekusi di web server sebelum disampaikan ke browser pada *client*. Contoh: PHP, ASP, JSP dan lainnya.

### 1.1. Terminologi pada Aplikasi Berbasis Web

Untuk memperjelas apa yang ada pada panduan ini, berikut adalah beberapa definisi istilah yang akan kami gunakan sesuai dengan standar *industry* yang ada.

- *Server*  
Sistem komputer yang menjalankan layanan koneksi HTTP. Perangkat lunak server (seperti Apache dan Microsoft IIS) biasanya berjalan di sistem ini untuk menanganinya koneksi HTTP.
- *Client*  
Komputer atau perangkat lunak yang membuat koneksi ke server, meminta data. Perangkat lunak klien paling sering adalah browser web, tetapi ada banyak hal lain yang membuat permintaan. Misalnya pemutar Flash Adobe dapat membuat permintaan HTTP, seperti yang bisa Aplikasi Java, Adobe PDF Reader, dan sebagian besar perangkat lunak. Saat memikirkan tentang pengujian, itu penting perlu diingat bahwa browser web hanyalah salah satu dari banyak jenis program yang dibuat permintaan web.



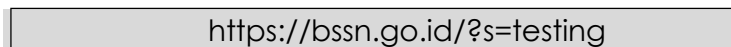
- *Request*

*Request* tersebut merangkum apa yang ingin diketahui *client*. *Request* terdiri dari beberapahal-hal, yang semuanya didefinisikan di sini: URL, parameter, dan metadata dalam formulirdari *header*.

- URL

*Universal Resource Locator* (URL) adalah jenis khusus dari *Universal Resource Identifier* (URI). Ini menunjukkan lokasi dari aplikasi web. URL terdiri dari protokol (http dan https). Protokol ini diikuti dengan token standar (: //) yang memisahkan protokol dari seluruh lokasi. Berikutnya adalah nama server yang akan dihubungi. Setelah nama server, ada jalur ke sumber daya di server itu. Adaparameter opsional untuk sumber daya itu. Akhirnya, dimungkinkan untuk menggunakan tanda *hash* (#) untuk mereferensikan fragmen internal atau link di dalam badan halaman.

Gambar 1 menunjukkan URL sederhana dari sebuah aplikasi berbasis WEB. Dimana https merupakan *protocol* yang digunakan adan bssn.go.id merupakan alamat sari server



https://bssn.go.id/?s=testing

Gambar 1 Contoh URL

- Parameter

Parameter adalah pasangan nilai dan isi dengan tanda sama dengan (=) di antara kunci dan nilai. Ada banyak di antaranya di URL dan dipisahkan oleh tanda. Parameter dapat diteruskan di URL, seperti yang ditunjukkan di Gambar 1 .

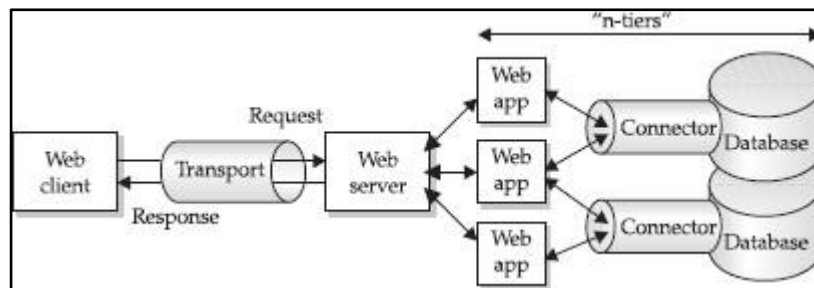
- *Method*

*Method* yang banyak digunakan sejauh ini, GET dan POST. Jika Anda mengetik URL ke browserweb Anda dan tekan *enter*, atau jika Anda mengklik *link*, Anda mengeluarkan permintaan GET. Sering kali Anda mengklik tombol pada formulir atau melakukan sesuatu yang relatif

rumit, sepertimengunggah gambar, Anda membuat permintaan POST. Metode lain (mis., PROPFIND, OPTIONS, PUT, DELETE).

## 1.2. Arsitektur Teknologi WEB

Arsitektur aplikasi web paling mendekati model komputasi terpusat, dengan banyak klien "*thick*" terdistribusi yang biasanya melakukan lebih sedikit daripada data presentasi yang menghubungkan ke server "*thin*" pusat yang melakukan sebagian besar pemrosesan. Apa yang membedakan arsitektur Web dari model komputasi terpusat tradisional (seperti sebagai komputasi *mainframe*) adalah bahwa mereka sangat bergantung pada teknologi yang dipopulerkan oleh World Wide Web, *Hypertext Markup Language* (HTML), dan bahasa utamanya , media transportasi, *Hypertext Transfer Protocol* (HTTP).



Gambar 2 Arsitektur Komponen Web

## 2. IT Security Assessment Aplikasi berbasis Web

IT Security Assessment (ITSA) merupakan pengukuran keamanan suatu sistem atau organisasi yang mana penilaian ini berbasis risiko, karena fokus dari kegiatan ITSA yakni terletak pada kerentanan dan dampak. Terdapat beberapa teknik untuk melakukan IT Security Assessment pada Aplikasi berbasis Web antaralain :

- Inspeksi dan Tinjauan Manual
- Pemodelan Ancaman
- Peninjauan Kode
- Uji Penetrasi

## 2.1. Inspeksi dan Tinjauan Manual

Inspeksi manual adalah tinjauan manusia yang biasanya menguji keamanan terhadap implikasi orang, kebijakan, dan proses suatu aplikasi. Inspeksi manual juga dapat mencakup pemeriksaan keputusan teknologi tersebut sebagai desain arsitektur. Kegiatan tersebut biasanya dilakukan dengan menganalisis dokumentasi atau melakukan wawancara dengan perancang atau pemilik sistem.

Sedangkan konsep inspeksi manual dan human *review* yang sederhana menjadi salah satu teknik yang paling kuat dan efektif yang tersedia. Dengan menanyakan seseorang bagaimana suatu aplikasi bekerja dan mengapa itu diterapkan dengan cara tertentu, penguji dapat dengan cepat menentukan apakah ada masalah keamanan yang mungkin terlihat. Manual inspeksi dan ulasan adalah salah satu dari sedikit cara untuk menguji proses siklus hidup pengembangan (SDLC) perangkat lunak itu sendiri dan untuk memastikan kebijakan atau keterampilan yang memadai dijalankan.

Ulasan secara manual sangat bagus untuk menguji apakah orang memahami proses keamanan, telah mengetahui kebijakan, dan memiliki keterampilan yang sesuai untuk merancang atau mengimplementasikan keamanan pada aplikasi.

Kegiatan lain pada tahapan ini termasuk meninjau dokumentasi secara manual, kebijakan pengkodean yang aman, persyaratan keamanan, dan arsitektur desain, semua harus diselesaikan dengan menggunakan inspeksi manual.

Keuntungan:

- Tidak membutuhkan teknologi pendukung
- Dapat diterapkan pada berbagai situasi
- Fleksibel
- Di awal SDLC

Kekurangan :

- Bisa memakan waktu yang banyak
- Materi pendukung tidak selalu tersedia

- Membutuhkan analisa dan keterampilan manusia yang signifikan agar efektif

## 2.2. Pemodelan Ancaman

Pemodelan ancaman telah menjadi teknik yang populer untuk membantu perancang sistem berpikir tentang ancaman keamanan pada sistem dan aplikasi yang mungkin hadapi. Oleh karena itu, pemodelan ancaman dapat dilihat sebagai penilaian risiko untuk aplikasi. Bahkan, memungkinkan perancang untuk melakukannya pengembangan strategi mitigasi untuk potensi kerentanan. Model ancaman harus dibuat sedini mungkin di SDLC, dan harus ditinjau kembali saat aplikasi berkembang.

Untuk mengembangkan model ancaman berdasarkan penilaian risiko menggunakan pendekatan antara lain :

- Mengurai aplikasi - gunakan proses manual pemeriksaan untuk memahami cara kerja aplikasi, asetnya, fungsionalitas, dan konektivitas.
- Mendefinisikan dan mengklasifikasikan aset - mengklasifikasikan aset ke dalam aset berwujud dan tidak berwujud dan memberi peringkat sesuai dengan kepentingan bisnis.
- Menjelajahi potensi kerentanan - baik teknis, operasional, atau manajemen.
- Menjelajahi potensi ancaman - kembangkan pandangan potensi yang realistis, vektor serangan dari sudut pandang penyerang, dengan menggunakan skenario ancaman.
- Membuat strategi mitigasi - mengembangkan kontrol mitigasi untuk setiap ancaman dianggap realistis.

Keluaran dari model ancaman itu sendiri dapat bervariasi tetapi biasanya terdiri kumpulan daftar dan diagram.

Keuntungan :

- Pandangan praktis penyerang tentang sistem
- Fleksibel
- Diawali SDLC

Kekurangan :

- Teknik yang relatif baru
- Model ancaman yang baik tidak secara otomatis berarti perangkat lunak yang baik

### 2.3. Peninjauan Kode

Peninjauan *source code* adalah proses memeriksa sumber secara manual kode aplikasi web untuk masalah keamanan. Banyak kerentanan yang serius yang tidak dapat dideteksi dengan bentuk analisis lainnya atau uji penetrasi.

Semua informasi untuk mengidentifikasi masalah keamanan apakah ada di kode. Berbeda dengan pengujian tertutup pihak ketiga pada perangkat lunak seperti sistem operasi, saat menguji aplikasi web (terutama jika dikembangkan sendiri) sumbernya kode harus tersedia untuk tujuan pengujian.

Banyak masalah keamanan yang tidak disengaja dan parah sulit ditemukan dengan bentuk analisis atau pengujian lain, seperti pengujian penetrasi, membuat analisis sumber kode teknik pilihan untuk pengujian teknis. Dengan *source code*, seorang penguji dapat secara akurat menentukan apa yang terjadi (atau yang diharapkan terjadi) dan menghapus pekerjaan menebak pengujian secara *blackbox*.

Analisis *source code* juga bisa sangat efisien menemukan masalah implementasi seperti tempat validasi *input* tidak dilakukan atau ketika prosedur kontrol terbuka gagal mungkin menyajikan. Namun perlu diingat bahwa prosedur operasional perlu dilakukan peninjauan juga, karena *source code* yang diterapkan mungkindidak sama dengan yang dianalisis di sini.

Keuntungan:

- Kelengkapan dan efektivitas
- Akurasi
- Cepat

Kekurangan :

- Membutuhkan pengembang keamanan yang sangat terampil
- Bisa melewatkan masalah di *library* yang dikompilasi
- Tidak dapat mendeteksi kesalahan *run-time* dengan mudah
- *Source code* yang sebenarnya diterapkan mungkin berbeda dari yang satu sedang dianalisis.

## 2.4. Uji Penetrasi

Uji Penetrasi telah menjadi teknik umum yang digunakan untuk menguji keamanan jaringan selama bertahun-tahun. Uji penetrasi pada dasarnya adalah "seni" untuk menguji aplikasi yang berjalan dari jarak jauh untuk menemukan kerentanan keamanan, tanpa mengetahui cara kerja bagian dalam aplikasi itu sendiri. Biasanya, tim uji penetrasi akan melakukannya akan memiliki akses ke aplikasi seolah-olah mereka adalah pengguna. Penguji bertindak seperti penyerang dan mencoba menemukan dan mengeksploitasi kerentanan. Dibanyak kasus penguji akan diberikan akun yang valid di sistem. Sedangkan uji penetrasi terbukti efektif dalam keamanan jaringan , teknik ini tidak secara alami diterjemahkan ke aplikasi. Saat uji penetrasi dilakukan pada jaringan dan sistem operasi, sebagian besar pekerjaan terlibat dalam pencarian dan kemudian mengeksploitasi kerentanan yang diketahui dalam teknologi tertentu.

Alat uji penetrasi telah banyak dikembangkan yang mengotomatiskan proses, tetapi dengan sifat aplikasi web keefektifannya biasanya kurang baik.

Uji penetrasi terfokus (yaitu, pengujian yang mencoba mengeksploitasi kerentanan yang diketahui terdeteksi di tinjauan sebelumnya) dapat berguna dalam mendeteksi jika ada kerentanan tertentu sebenarnya diperbaiki dalam *source code* yang digunakan di situs web.

Keuntungan:

- Bisa cepat (dan karenanya murah)
- Membutuhkan keahlian yang relatif lebih rendah daripada tinjauan *source code*

- Menguji kode yang sebenarnya sedang diekspos

Kekurangan:

- Terlambat dalam SDLC
- Hanya uji benturan di aplikasi depan

### 3. Uji Penetrasi Aplikasi berbasis Web

Uji Penetrasi digunakan mengevaluasi keamanan aplikasi dengan memvalidasi dan memverifikasi efektivitas kontrol keamanan aplikasi. Proses ini melibatkan analisis aktif terhadap aplikasi untuk setiap kelemahan atau kerentanan. Masalah keamanan apa pun yang ditemukan akan dilaporkan kepada pemilik sistem, bersama dengan penilaian risiko terhadap dampaknya, rekomendasi untuk mitigasi atau solusi teknis. Pada uji penetrasi harus mengikuti kerangka kerja IT *Security Assessment* , bagian khusus dari tahapan uji penetrasi pada aplikasi berbasis web dijelaskan pada panduan ini.

#### 3.1. Tahapan Pengujian

Pada Uji Penetrasi dilakukan dalam 2 (dua) bagian antara lain :

- Mode Pasif

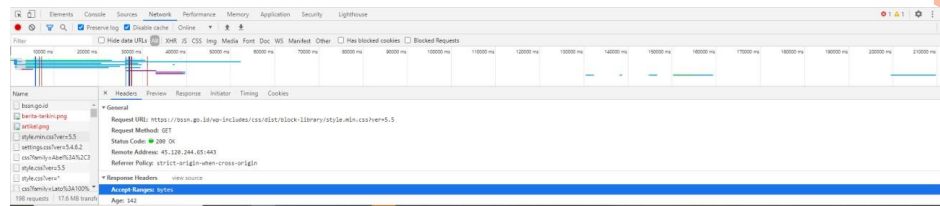
Dalam mode pasif, penguji mencoba memahami logika aplikasi dan bermain dengan aplikasi tersebut. Alat yang dapat digunakan untuk pencarian informasi. Misalnya, HTTP *proxy* dapat digunakan untuk mengamati semua permintaan dan tanggapan pada protokol HTTP. Di akhir fase ini, penguji harus memahami semua titik akses (gerbang) aplikasi (mis., HTTP *Header* , parameter, dan *cookie*). Pengumpulan Informasi bagian menjelaskan cara melakukan pengujian pada mode pasif.

Pada gambar 3 merupakan contoh dari kegiatan mode pasif untuk melihat *request* dan *response* terkait aplikasi WEB. Alat yang bisa digunakan adalah *addons Developer Tool* dari browser (chrome<sup>[1]</sup> , mozilla firefox<sup>[2]</sup> dan Microsoft Edge<sup>[3]</sup>) yang secara *default* sudah ada pada browser tersebut.

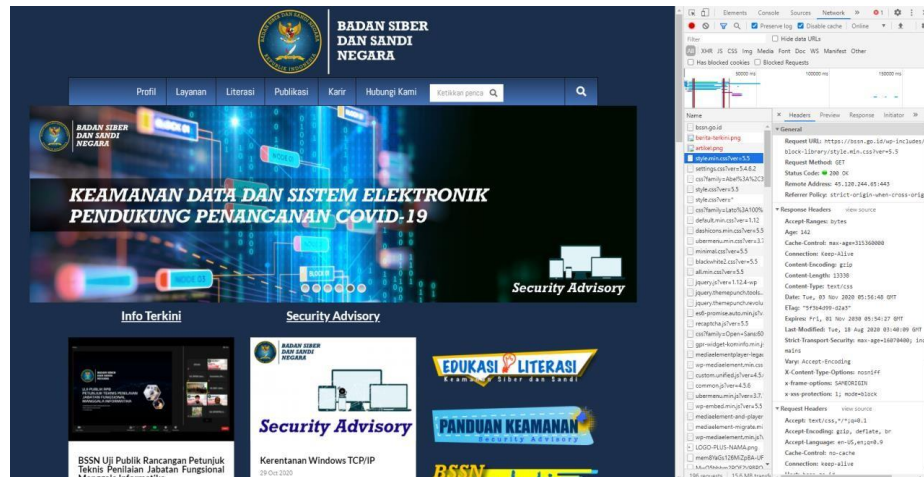


# DRAFT PANDUAN TEKNIS PELAKSANAAN IT SECURITY ASSESSMENT (ITSA)

• • •



Gambar 3 Tampilan Developer Tools



Gambar 4 Mode Pasif pada bssn.go.id Menggunakan Developer Tools

## • Mode Aktif

Uji penetrasi dengan mode aktif dilakukan langsung terhadap aplikasi web yang dijadikan target dalam *IT Security Assessment*.

Beberapa pengujian dalam uji penetrasi terhadap aplikasi berbasis web meliputi beberapa tahapsesuai dengan gambar di bawah.



Gambar 5 Kategori Pengujian Uji Penetrasi

### 3.2. *Information Gathering*

Memahami konfigurasi yang diterapkan dari server yang menghosting aplikasi web hampir sama pentingnya dengan dalam pengujian keamanan aplikasi. Platform aplikasi sangat luas dan bervariasi, tetapi ada beberapa kunci kesalahan konfigurasi platform dapat membahayakan aplikasi di cara yang sama aplikasi tidak aman dapat membahayakan server.

Beberapa informasi yang perlu diketahui dalam tahapan ini adalah sebagai berikut :

- *Server IP address* , termasuk virtual *IP address*
- Port dan Layanan beserta versi layanan pada server
- *Operating Sistem* yang dipergunakan
- Jenis *framework* aplikasi

#### 3.2.1 *Server Discovery*

Tidak sulit untuk tidak menemukan server Web di Internet saat ini. Cukup tambahkan *www.* dan *.com* (atau *.org* atau *.edu* atau *.id*) ke hampir semua istilah, nama, atau frasa yang bisa dibayangkan dan Anda berdiri kesempatan yang sangat bagus untuk menemukan server Web. Penyerang yang menargetkan organisasi Anda mungkin akan mengambil pendekatan ini terlebih dahulu karena hampir tidak membutuhkan usaha. Mereka bahkan mungkin mencoba menghitung server atau situs Web lain dengan menebak nama *host* yang umum, seperti *www1.victim.com* atau *shopping.victim.com*.

Dalam melakukan *server discovery* terdapat beberapa *website* dan alat yang bisa digunakan untuk menemukan informasi-informasi yang dibutuhkan dalam tahapan ini.

# DRAFT PANDUAN TEKNIS PELAKSANAAN IT SECURITY ASSESSMENT (ITSA)



### Informasi IP Server

Untuk mendapatkan Informasi terkait IP server bisa menggunakan netcraft.com

#### Network

Site	<a href="https://bssn.go.id">https://bssn.go.id</a>	Domain	<a href="https://bssn.go.id">bssn.go.id</a>
Netblock Owner	<a href="#">BADAN SIBER DAN SANDI NEGARA (BSSN)</a>	Nameserver	<a href="#">ns1.bssn.go.id</a>
Hosting company	PT, Cyber Network Indonesia	Domain registrar	unknown
Hosting country	<a href="#">ID</a>	Nameserver organisation	unknown
IPv4 address	<a href="#">45.120.244.65</a> (VirusTotal ID)	Organisation	unknown
IPv4 autonomous systems	<a href="#">AS135457</a>	DNS admin	<a href="#">pusdatik@bssn.go.id</a>
IPv6 address	Not Present	Top Level Domain	Indonesia (.go.id)
IPv6 autonomous systems	Not Present	DNS Security Extensions	Enabled
Reverse DNS	<a href="#">bssn.go.id</a>		

#### IP delegation

IPv4 address (45.120.244.65)

IP range	Country	Name	Description
0.0.0.0-255.255.255.255	N/A	IANA-BLK	The whole IPv4 address space
<a href="#">45.0.0.0-45.255.255.255</a>	<a href="#">United States</a>	NET45	American Registry for Internet Numbers
<a href="#">45.119.212.0-45.127.255.255</a>	<a href="#">Australia</a>	APNIC	Asia Pacific Network Information Centre
<a href="#">45.120.244.0-45.120.247.255</a>	<a href="#">Indonesia</a>	IDNIC-BSSN-ID	BADAN SIBER DAN SANDI NEGARA (BSSN)
<a href="#">45.120.244.65</a>	<a href="#">Indonesia</a>	IDNIC-BSSN-ID	BADAN SIBER DAN SANDI NEGARA (BSSN)

Gambar 6 Informasi IP Server Menggunakan netcraft

#### Informasi Pemilik Domain

Informasi terkait pemilik domain merupakan informasi yang bisa digunakan untuk mengetahui organisasi yang mendapatkan domain dan juga bisa digunakan untuk mencari informasi yang terkait terhadap target dari *IT Security Assessment*. Untuk domain .id bisa menggunakan *whois.pandi.id* untuk mengetahui terkait pendaftar dari sebuah domain dan untuk domain lain bisa menggunakan *whois.domaintools.com*.

#### Result for bssn.go.id

```
Domain ID: PANDI-DO633781
Domain Name:bssn.go.id
Created On:2017-07-07 08:22:17
Last Updated On:2019-08-22 01:12:04
Expiration Date:2021-07-07 23:59:59
Status:ok
=====
Sponsoring Registrar PANDI ID:H4964483
Sponsoring Registrar Organization:Kementerian Komunikasi dan Informati
Sponsoring Registrar City:Jakarta Pusat
Sponsoring Registrar State/Province:Jakarta
Sponsoring Registrar Postal Code:10110
Sponsoring Registrar Country:ID
Sponsoring Registrar Phone:622138433507
Sponsoring Registrar Contact Email:hostmaster@pandi.id
Name Server:ns1.bssn.go.id
Name Server:ns2.bssn.go.id
DNSSEC:Signed
```

Gambar 7 Hasil Kueri Registrar Domain

## DRAFT PANDUAN TEKNIS PELAKSANAAN IT SECURITY ASSESSMENT (ITSA)



### Enumerasi *Subdomain*

Enumerasi *subdomain* digunakan untuk mendapatkan *host* lain dari domain yang masuk dalam target ITSA. Pada uji penetrasi semua *vector* serangan terhadap target harus bisa diuji untuk mencari kerentanan yang mungkin terjadi dari sisi lain selain dari target ITSA untuk itu diperlukan enumerasi terhadap *subdomain*. Enumerasi secara pasif tanpa menyentuh target ITSA bisa menggunakan *securitytrails.com* dan alat bantu *dnsrecon*.

#### bssn.go.id subdomains



Domain	Rank	Hosting Provider	Mail Provider
bssn.go.id	3,835,962	BADAN SIBER DAN SANDI NEGARA (BSSN)	BADAN SIBER DAN SANDI NEGARA (BSSN)
esign-ui-bsre.bssn.go.id	-	-	-
esign-dev.bssn.go.id	-	BADAN SIBER DAN SANDI NEGARA (BSSN)	-
pusmanas.bssn.go.id	-	BADAN SIBER DAN SANDI NEGARA (BSSN)	-
siab-bsre.bssn.go.id	-	BADAN SIBER DAN SANDI NEGARA (BSSN)	-
vpn-bsre.bssn.go.id	-	BADAN SIBER DAN SANDI NEGARA (BSSN)	-
esign2-dev.bssn.go.id	-	BADAN SIBER DAN SANDI NEGARA (BSSN)	-
misp.bssn.go.id	-	BADAN SIBER DAN SANDI NEGARA (BSSN)	-
sihs.bssn.go.id	-	-	-
ns2.bssn.go.id	-	PT Cyber Network Indonesia	-
zmmta3.bssn.go.id	-	-	-
gitlab-bsre.bssn.go.id	-	BADAN SIBER DAN SANDI NEGARA (BSSN)	-
ams-ui-bsre.bssn.go.id	-	BADAN SIBER DAN SANDI NEGARA (BSSN)	-
cpns.bssn.go.id	-	BADAN SIBER DAN SANDI NEGARA (BSSN)	-

Gambar 8 Hasil Enumerasi Subdomain

#### Pencarian Port dan versi layanan pada server

Setelah server diidentifikasi, sekarang saatnya untuk mencari tahu *port* apa yang menjalankan HTTP(atau SSL sebagai kasusnya) dan juga *port* lainnya pada server. Kami menyebutnya proses penemuanlayanan, dan itu dilakukan dengan menggunakan pemindaian *port* untuk daftar *port* server Web umum. Alat yang bisa digunakan untuk memindai *port* dan layanan adalah nmap<sup>[4]</sup>.

Perintah

```
$nmap -sV -A target
```

```
root@sungai:/# nmap -sV -A localhost
Starting Nmap 7.80 ( https://nmap.org ) at 2020-11-03 16:18 WIB
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000026s latency).
Other addresses for localhost (not scanned): ::1
Not shown: 997 closed ports
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.2p1 Debian 4 (protocol 2.0)
80/tcp    open  http      nginx 1.18.0
|_ http-server-header: nginx/1.18.0
3306/tcp  open  mysql     MySQL 5.5.5-10.3.22-MariaDB-1
|_ mysql-info:
|   Protocol: 10
|   Version: 5.5.5-10.3.22-MariaDB-1
|   Thread ID: 123
|   Capabilities flags: 63486
|   Some Capabilities: LongColumnFlag, Support41Auth, DontAllowDatabaseTableColumn, SupportsTransac
tions, IgnoreSigpipes, ODBCClient, Speaks41ProtocolOld, FoundRows, Speaks41ProtocolNew, Interacti
veClient, ConnectWithDatabase, IgnoreSpaceBeforeParenthesis, SupportsLoadDataLocal, SupportsCompre
ssion, SupportsAuthPlugins, SupportsMultipleStatements, SupportsMultipleResults
|   Status: Autocommit
|   Salt: .jj!^;I@Bkt(fJ~^9B)q
|_ Auth Plugin Name: mysql_native_password
Device type: general purpose
Running: Linux 2.6.X
OS CPE: cpe:/o:linux:linux_kernel:2.6.32
OS details: Linux 2.6.32
Network Distance: 0 hops
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel

OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit
/ .
Nmap done: 1 IP address (1 host up) scanned in 68.78 seconds
```

Gambar 9 Hasil Pemindaian Port dan Layanan

### 3.2.2 Search Engine Discovery

*Search engine discovery* digunakan untuk mencari kebocoran informasi dari target ITSA . Ada elemen langsung dan tidak langsung untuk penemuan mesin pencari dan pengintaian. Metode langsung berhubungan dengan pencarian indeks dan konten terkait



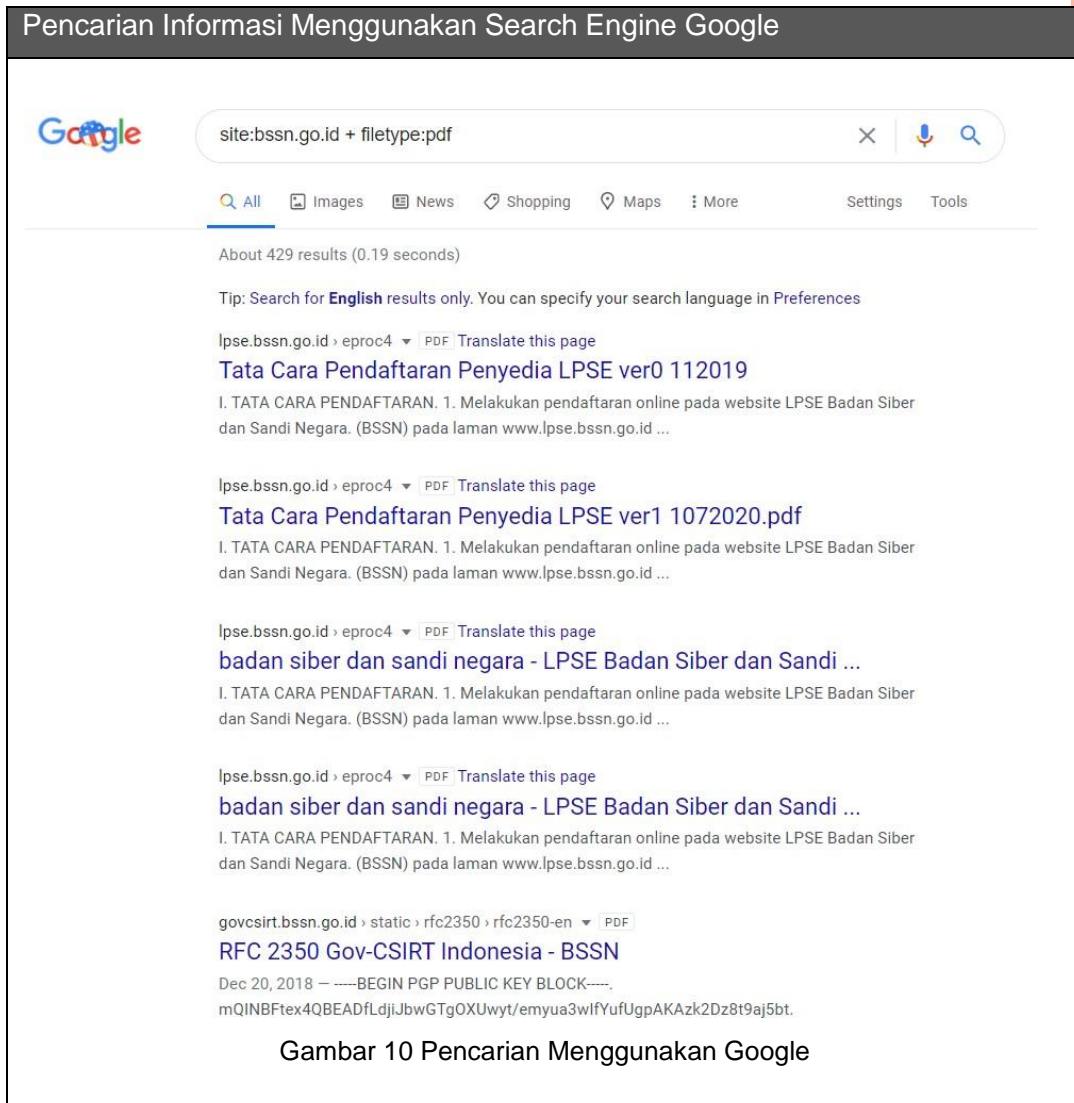
dari *cache*. Metode tidak langsung berhubungan dengan mengumpulkan informasi desain dan konfigurasi sensitif dengan menelusuri forum, newsgroup, dan situs lainnya. Beberapa situs mesin pencari yang bisa digunakan untuk melakukan *search engine discovery* adalah

- Google
- Bing
- Shodan
- Baidu
- Censy

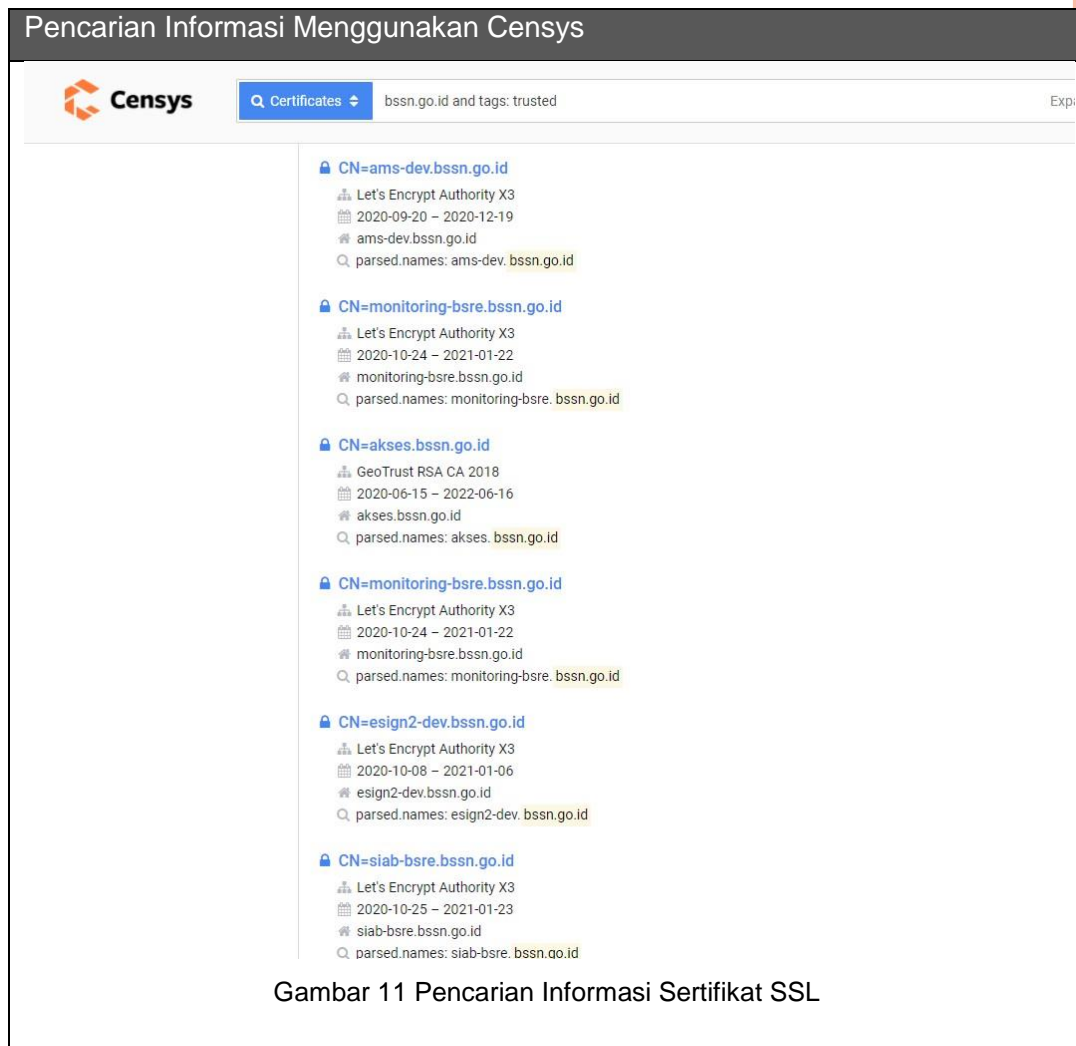
Tujuan utama dari *search engine discovery* adalah untuk mencari informasi konfigurasi aplikasi / sistem / organisasi diekspos baik secara langsung (di situs web organisasi) atau tidak langsung (di situs webpihak ketiga).

Penggunaan mesin pencari google untuk melakukan *search engine discovery* biasa dikenal dengan istilah *google search engine hacking*, google banyak menyediakan opsi-opsi untuk melakukan pencarian terhadap informasi yang dibutuhkan.

## DRAFT PANDUAN TEKNIS PELAKSANAAN IT SECURITY ASSESSMENT (ITSA)







### 3.2.3 *Fingerprint Framework Aplikasi*

*Fingerprint framework* aplikasi adalah *subtugas* penting dari proses pengumpulan informasi. Mengetahui jenis *framework* bisa otomatis memberikan keuntungan besar jika *framework* aplikasi seperti itu sudah adadiuji oleh penguji penetrasi lain. Bukan hanya kerentanan yang diketahui dalam versi yang belum ditambah tetapi kesalahan konfigurasi tertentu dalam *framework* dan struktur *file* yang diketahui yang membuat proses *fingerprint* sangat penting. Beberapa vendor dan versi *framework* berbeda secara luas. Informasi tentangnya sangat membantu dalam proses pengujian, dan juga dapat membantu mengubah jalannya tes. Informasi



seperti itu dapat diperoleh dengan analisis yang cermat terhadap lokasi umum tertentu. Setiap *framework* web memiliki beberapa penanda di lokasi tersebut yang membantu penyerang untuk menemukan mereka. Ini pada dasarnya adalah semua alat otomatis yang lakukan, mereka mencari penanda dari lokasi yang ditentukan sebelumnya dan kemudian membandingkan itu ke *database signature* yang dikenal. Untuk akurasi yang lebih baik beberapa tanda biasanya digunakan.

Ada beberapa lokasi paling umum yang harus dicari untuk mendapatkan *framework* yang digunakan saat ini:

- HTTP *headers*
- *Cookies*
- HTML *source code*
- Spesifik Folder dan *File*

Alat yang umum digunakan untuk bisa mengidentifikasi jenis *framework* yang digunakan oleh sebuah aplikasi berbasis web adalah whatweb<sup>[5]</sup>.

#### Identifikasi *Framework* Aplikasi

Perintah

```
$whatweb urltarget
```

```
root@sungai:/# whatweb https://bssn.go.id
/usr/lib/ruby/vendor_ruby/target.rb:188: warning: URI.escape is obsolete
https://bssn.go.id [200 OK] Frame, HTML5, IP[45.120.244.65], JQuery, MetaGenerat
or[Divi Child v.*,Powered by Slider Revolution 5.4.6.2 - responsive, Mobile-Frie
ndly Slider Plugin for WordPress with comfortable drag and drop interface.], Pow
eredBy[Slider], Script[text/javascript], Strict-Transport-Security[max-age=16070
400; includeSubDomains], Title[bssn.go.id | Situs Web Resmi Badan Siber dan Sand
i Negara], UncommonHeaders[link,x-content-type-options], WordPress, X-Frame-Opti
ons[SAMEORIGIN], X-UA-Compatible[IE=edge], X-XSS-Protection[1; mode=block], YouT
ube
```

Gambar 12 Hasil Identifikasi *Framework* Aplikasi

Dari hasil whatweb didapatkan bahwa *website* bssn.go.id menggunakan *framework* Wordpress. Jika menggunakan wordpress bisa mencari kerentanan menggunakan wpscan.

### 3.3. *Configuration and Deployment Management*

Konfigurasi yang tepat dari elemen tunggal yang membentuk sebuah arsitektur aplikasi penting untuk mencegah kesalahan yang mungkin terjadi membahayakan keamanan seluruh arsitektur. Tinjauan dan pengujian konfigurasi adalah tugas penting dalam membuat dan memelihara sebuah arsitektur. Ini karena banyak sistem yang berbeda biasanya dilengkapi dengan konfigurasi umum yang mungkin tidak cocok untuk tugas yang akan mereka lakukan di situs tertentu tempat mereka menginstal. Sedangkan instalasi web dan server aplikasi akan berisi banyak fungsi (seperti contoh aplikasi, dokumentasi, tes halaman) apa yang tidak penting harus dihapus sebelum masuk ke sistem produksi untuk menghindari eksploitasi pasca-pemasangan.

Pada tahapan ini beberapa pengujian antara lain :

- Pencarian *sensitive* direktori dan *file*

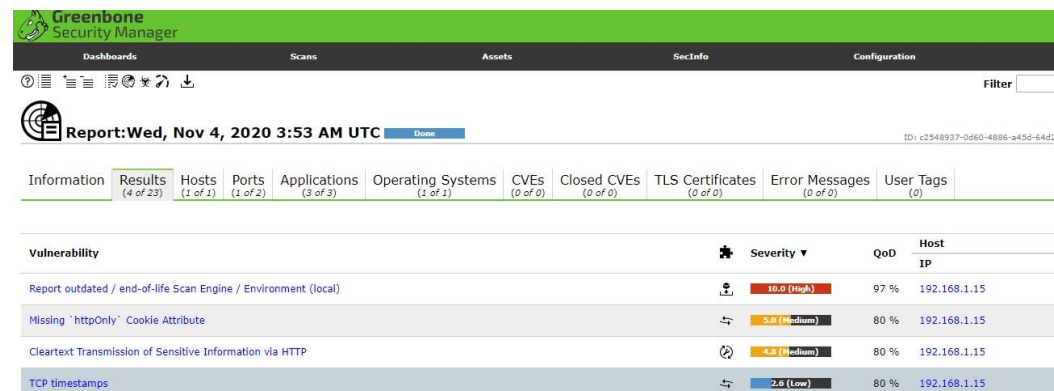
## DRAFT PANDUAN TEKNIS PELAKSANAAN IT SECURITY ASSESSMENT (ITSA)



- Pencarian kerentanan terhadap server

Alat yang bisa digunakan untuk mencari kerentanan terhadap server adalah OpenVas<sup>[6]</sup> sedangkan OWASP ZAP<sup>[7]</sup> bisa digunakan untuk mencari kerentanan pada sisi aplikasi dan pencarian sensitif direktori/file.

Pencarian kerentanan menggunakan OpenVas

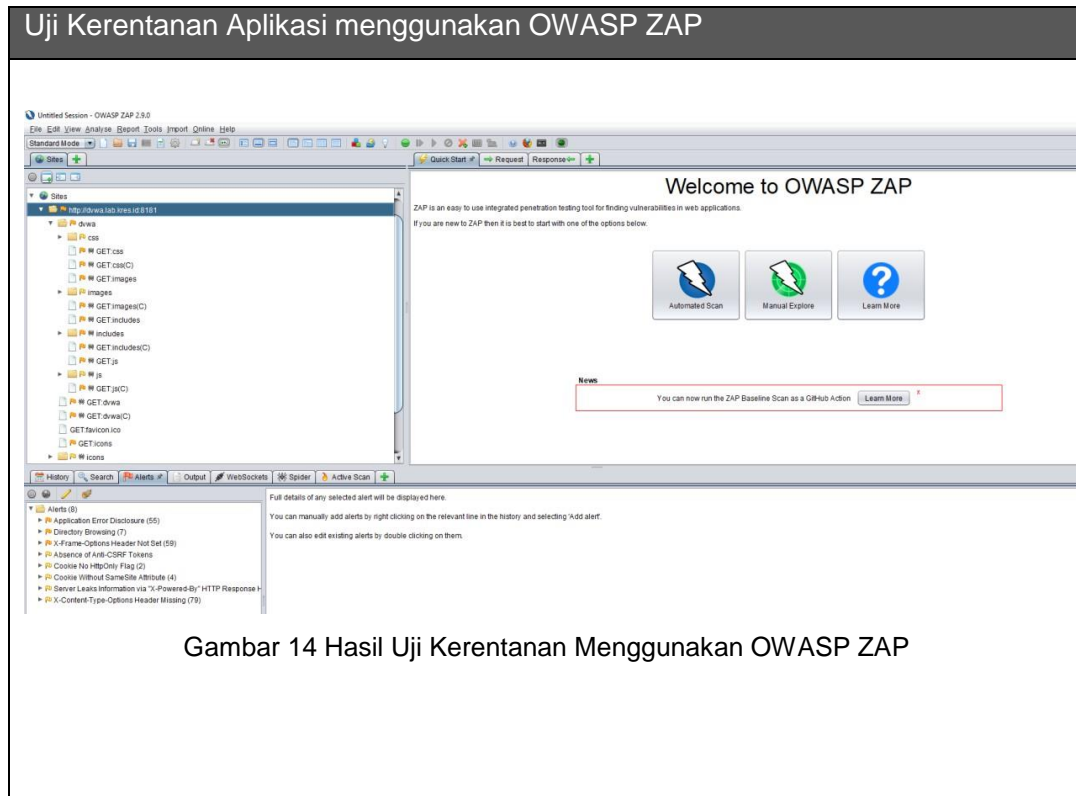


The screenshot displays the OpenVas Greenbone Security Manager interface. The top navigation bar includes links for Dashboards, Scans, Assets, SecInfo, and Configuration. The main content area shows a report for 'Wed, Nov 4, 2020 3:53 AM UTC'. The report is categorized into Information, Results (4 of 23), Hosts (1 of 1), Ports (1 of 2), Applications (3 of 3), Operating Systems (1 of 1), CVEs (0 of 0), Closed CVEs (0 of 0), TLS Certificates (0 of 0), Error Messages (0 of 0), and User Tags (0). The 'Vulnerability' section lists the following findings:

Vulnerability	Severity	QoD	Host IP
Report outdated / end-of-life Scan Engine / Environment (local)	10.0 (High)	97 %	192.168.1.15
Missing 'httpOnly' Cookie Attribute	5.0 (Medium)	80 %	192.168.1.15
Cleartext Transmission of Sensitive Information via HTTP	4.8 (Medium)	80 %	192.168.1.15
TCP timestamps	2.0 (Low)	80 %	192.168.1.15

Gambar 13 Hasil *Vulnerability Assessment* menggunakan OpenVas

Dari gambar di atas kerentanan yang terkait adalah “*Missing `httpOnly` Cookie Attribute*”



Gambar 14 Hasil Uji Kerentanan Menggunakan OWASP ZAP

### 3.4. Identify Management

Pada tahapan *identify management* pengujian uji penetrasi melihat proses *login* dan pendaftaran pengguna jika ada dalam proses dari aplikasi berbasis web.

Hanya ada dua peran dalam sistem secara umum yaitu pengguna dan administrator. Jika suatu sistem aplikasi berbasis web tidak mempunyai proses dari pendaftaran dan *login* maka tahapan ini tidak perlu dijalankan. Proses pendaftaran pengguna diuji secara manual menggunakan pengetahuan tentang cara kerjasistem.

Beberapa proses pengujian pada tahap ini antara lain :

- Pengujian registrasi pengguna
  - Pengujian terhadap karakter ilegal pada setiap *form* yang ada
- Pengujian *login* pengguna
  - Pengujian respon gagal
  - Pengujian *bruteforce*



#### Pengujian respon sukses dan gagal untuk login

Ruang lingkup pengujian ini adalah untuk memverifikasi apakah ada kemungkinan untuk mengumpulkan satu set nama pengguna yang valid dengan berinteraksi dengan mekanisme autentikasi dari aplikasi tersebut. Tes ini akan berguna untuk uji penetrasi, di mana penguji memverifikasi jika, diberi nama pengguna yang valid, itu mungkin untuk menemukan kata sandi yang sesuai dan berguna untuk melakukan *bruteforce login* pada sistem.

## Vulnerability: Brute Force

### Login

Username:

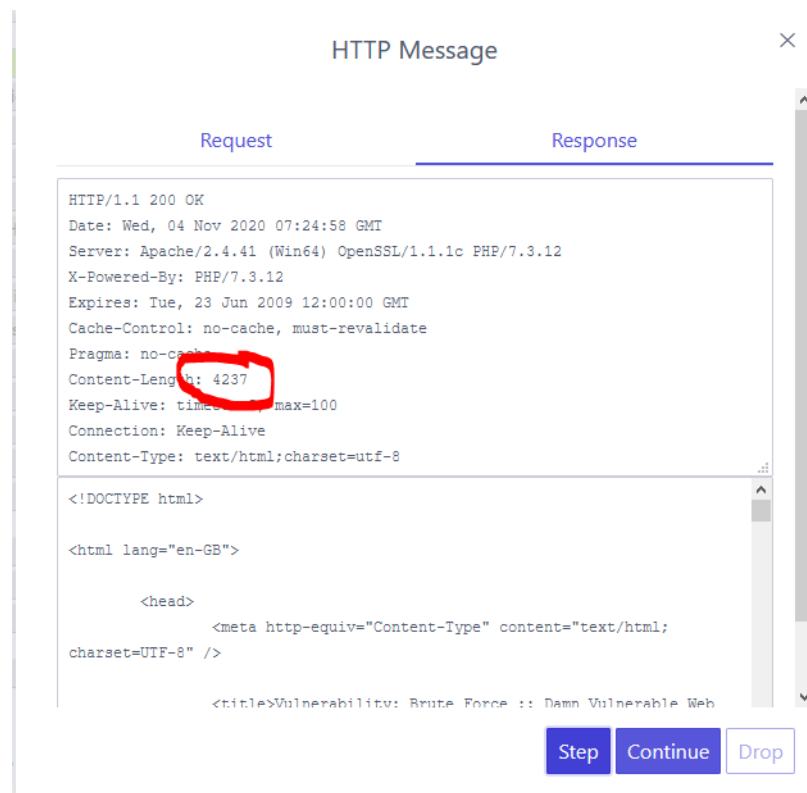
Password:

Login

Username and/or password incorrect.

Gambar 15 Respon *Login* Gagal

Jika dilihat pada respon yang lebih detail dari server bisa menggunakan OWASP ZAP



Gambar 16 Respon Detail *Login* Gagal

Pada gambar 16 hasil respon *login* gagal mempunyai ukuran *content* 4237 bytes

## Vulnerability: Brute Force

### Login

Username:

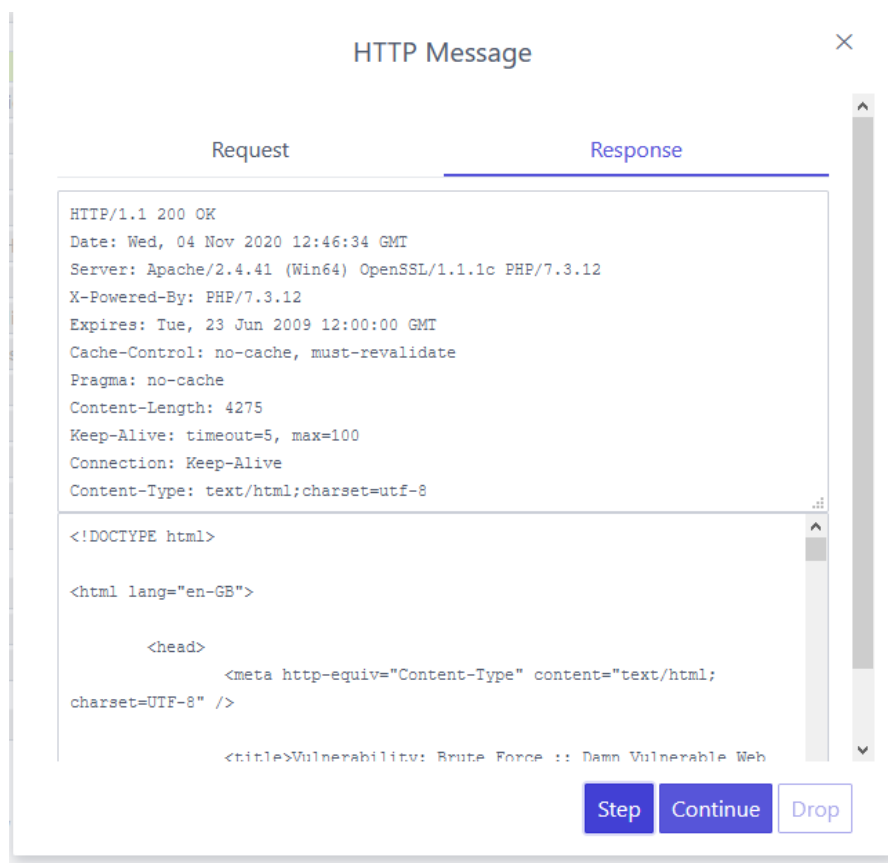
Password:

Login

Welcome to the password protected area admin



Gambar 17 Respon *Login* Sukses



Gambar 18 Respon *Login* Sukses



Pada gambar 18 jika *login* sukses maka akan mempunyai ukuran *content* sebesar 4275 bytes. Perbedaan *respon* sukses dan gagal bisa digunakan untuk melakukan perbandingan pada saat melakukan serangan *bruteforce* terhadap halaman *login*.

### 3.5. Authentication

Autentikasi memainkan peran penting dalam sebuah keamanan aplikasi sejak semua proses berikutnya pada sebuah sistem aplikasi biasanya dibuat berdasarkan identitas yang ditetapkan oleh kredensial yang diberikan. Sebuah aplikasi biasanya membutuhkan pengguna untuk memasukkan *username* dan *password* untuk membuktikan bahwa pengguna adalah yang dia yang berhak. Sebagian besar jenis autentikasi berbasis internet menggunakan nama pengguna dan kata sandi untuk mengautentikasi pengguna, tapi bentuk lain dari autentikasi aplikasi berbasis web ada untuk memberikan keamanan yang lebih kuat.

Pada bagian ini membahas protokol dan teknik autentikasi Web yang umum dan membahas serangan umum terhadap teknik tersebut.

#### 3.5.1 Mekanisme Autentikasi

Beberapa mekanisme autentikasi yang ada saat ini sesuai dengan standar HTTP antara lain :

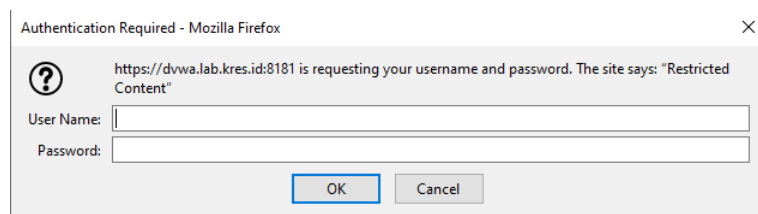
- *HTTP Authentication: Basic and Digest*
- *Integrated Windows (NTLM)*
- *Negotiate*
- *Certificate-Based*
- *Forms-Based Authentication*

#### 3.5.1.1. *HTTP Authentication : Basic*

Autentikasi dasar (*Basic authentication*), seperti namanya, adalah bentuk paling dasar dari autentikasi yang tersedia ke aplikasi berbasis Web. Ini pertama kali didefinisikan dalam spesifikasi HTTP itu sendiri. Autentikasi dasar ini memiliki bagian yang adil pada masalah keamanan dan masalahnya didokumentasikan dengan baik.

Autentikasi dasar dimulai dengan klien membuat permintaan ke server Web yang melindungi sumber daya, tanpa kredensial autentikasi apa pun. Server akan membalas dengan akses pesan ditolak yang berisi *header WWW-Authenticate* yang meminta kredensial autentikasi dasar.

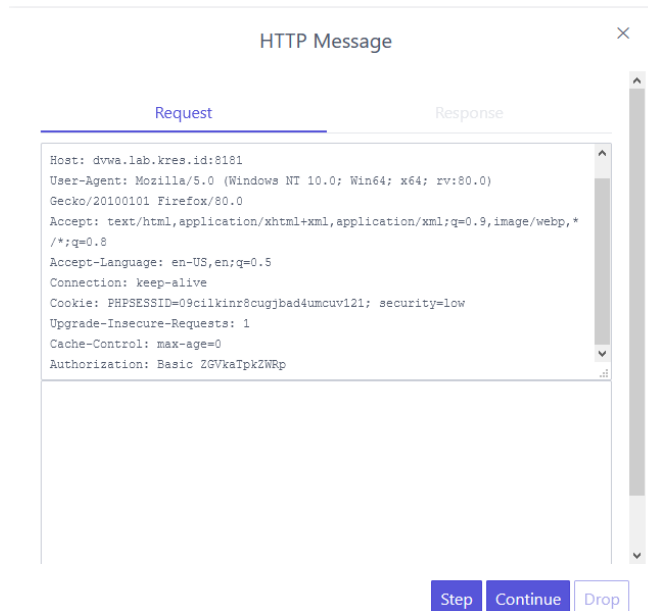
Kebanyakan browser Web berisi rutinitas untuk menangani permintaan semacam itu secara otomatis, dengan meminta pengguna untuk memasukkan nama pengguna dan kata sandi seperti yang ditunjukkan pada Gambar 19. Perhatikan bahwa ini adalah jendela sistem operasi terpisah yang dibuat oleh browser, dan bukan dalam bentuk HTML.



Gambar 19 HTTP Basic Authentication

Setelah pengguna mengetikkan kata sandinya, browser mengeluarkan kembali permintaan tersebut, untuk melakukan autentikasi. Berikut adalah pertukaran autentikasi Dasar yang khas terlihat seperti dalam HTTP mentah (diedit agar singkat). Pertama, permintaan awal untuk sumber daya diamankan menggunakan autentikasi

dasar:



Gambar 20 Request HTTP Basic Authentication

Pada gambar 20 terlihat pada *header request client* terhadap *Authorization : Basic* yang menandakan bahwa aplikasi berbasis web menggunakan jenis autentikasi HTTP Basic Authentication.

Penggunaan *HTTP Basic Authentication* rentan untuk dilakukan serangan MITM jika tidak menggunakan *protocol* yang aman untuk berkomunikasi antara *client* dan server.

#### 3.5.1.2. *HTTP Authentication : Digest*

Autentikasi *Digest* dirancang untuk memberikan tingkat keamanan yang lebih tinggi daripada autentikasi Dasar. Ini dijelaskan di RFC 2617. Autentikasi *Digest* didasarkan pada autentikasi model respons-tantangan (*challenge-response authentication*). Ini adalah teknik umum yang digunakan untuk membuktikan bahwa seseorang mengetahui suatu rahasia, tanpa diminta orang yang mengirim rahasia dalam *plaintext* akan mudah disadap.



Autentikasi *Digest* berfungsi mirip dengan autentikasi Dasar. Para pengguna membuat permintaan tanpa kredensial autentikasi, dan server Web membalas dengan WWW-Authenticate *header* yang menunjukkan kredensial diperlukan untuk mengakses sumber daya yang diminta. Tapi bukannya mengirim namapengguna dan kata sandi dalam pengkodean Base 64 seperti pada Basic, tantangan (*challenge*) server klien dengan nilai acak yang disebut nonce. Browser kemudian menggunakan satu arah fungsi kriptografi untuk membuat intisari pesan (*message digest*) dari nama pengguna, kata sandi, yang diberikan nilai nonce, metode HTTP, dan URI yang diminta. Fungsi intisari pesan (*message digest*), juga dikenal sebagai algoritma *hashing*, adalah fungsi kriptografi yang dengan mudah dihitung dalam satu arah, dan secara komputasi tidak bisa untuk dibalik. Bandingkan ini dengan autentikasi Dasar, di mana membalik pengkodean Base 64 itu hal yang mudah. Algoritme *hashing* apa pun dapat ditentukan di dalamnya tantangan (*challenge*) server; RFC 2617 menjelaskan penggunaan fungsi *hash* MD5 sebagai defaultnya.

#### 3.5.1.3. ***Integrated Windows (NTLM)***

Autentikasi *Integrated Windows (NTLM)* (sebelumnya dikenal sebagai autentikasi NTLM dan Windows Autentikasi tantangan / respons NT) menggunakan NT LAN *Manager (NTLM)* milik Microsoft melalui HTTP. Karena menggunakan NTLM bukan daripada algoritma intisari standar, ini hanya berfungsi di antara Microsoft Internet Explorer browser dan server Web IIS. Karena sebagian besar situs Internet ingin mendukung banyak browser, mereka biasanya tidak menerapkan autentikasi Windows Terpadu. Ini membuat autentikasi Windows

Terintegrasi lebih cocok untuk penggunaan intranet. Autentikasi Windows terintegrasi bekerja dengan cara yang sama seperti autentikasi *Digest*, menggunakan mekanisme tantangan-respons. Ketika klien meminta sumber daya yang dilindungi oleh Terintegrasi Autentikasi Windows, server merespons dengan HTTP 401 Access *Denied* dan sebuah WWW- Authenticate: *header* NTLM [challenge]. Nilai [tantangan] berisi intisari *nonce* NTLM dan informasi lain yang terkait dengan permintaan tersebut. Internet Explorer kemudian akan melakukannya ,kumpulkan kredensial NTLM untuk pengguna Windows yang saat ini masuk, gunakan NTLM algoritma untuk mencirikan nilai tantangan, lalu memberikan nilai *hash* dalam respons HTTP dengan *header* Authorization:NTLM [response].



Gambar 21 Login Window NTLM Authentication

#### 3.5.1.4. *Negotiate*

Autentikasi *Negotiate* adalah perpanjangan dari autentikasi NTLM; jenis ini diperkenalkan pertama kali di Windows 2000. Ini menyediakan autentikasi berbasis Kerberos melalui HTTP dan dianggap sangat aman. Sesuai dengan namanya, Autentikasi Negosiasi menggunakan proses negosiasi untuk memutuskan pada



tingkat keamanan yang akan digunakan. Secara default, Negotiasikan akan menggunakan autentikasi metode terkuat yang tersedia. Dalam kasus host Windows 2000 di domain Windows yang sama. Negosiasi akan menggunakan autentikasi berbasis Kerberos. Namun, jika server tidak sama dalam domain, Negotiasikan akan kembali ke autentikasi berbasis NTLM. Negosiasi dapat memberikan keamanan yang kuat jika semua *host* adalah Windows 2000 (atau lebih tinggi) dan berada di domain yang sama. Namun, konfigurasi ini cukup terbatas dan tidak umum kecuali di intranet perusahaan. Selain itu, karena kemampuan *fallback* alami Bernegosiasi, NTLM biasanya dapat digunakan sebagai pengganti autentikasi Kerberos. Peretas saja perlakukan Negotiasikan sebagai NTLM dan lakukan serangan seolah-olah mereka berurusan dengan autentikasi NTLM.

#### 3.5.1.5. ***Certificate-Based***

Autentikasi berdasarkan sertifikat lebih kuat daripada metode autentikasi yang telah kita diskusikan sejauh ini. Autentikasi bersertifikat menggunakan kriptografi kunci publik, dan sertifikat digital untuk mengautentikasi pengguna. Autentikasi sertifikat dapat digunakan selain yang lain skema autentikasi berbasis kata sandi untuk memberikan keamanan yang lebih kuat. Penggunaan sertifikat dianggap sebagai implementasi autentikasi dua faktor. Selain sesuatu Anda tahu (kata sandi Anda), Anda harus mengautentikasi dengan sesuatu yang Anda miliki (sertifikat). Sertifikat dapat disimpan di perangkat keras (yaitu, kartu pintar) untuk memberikan keseimbangan tingkat keamanan yang lebih tinggi — kepemilikan token fisik dan ketersediaan yang sesuai pembaca kartu pintar akan diminta untuk mengakses situs yang dilindungi



sedemikian rupa. Sertifikat klien memberikan keamanan yang lebih kuat, namun dengan biaya yang besar. Kesulitan mendapatkan sertifikat, pendistribusian sertifikat, dan pengelolaan sertifikat untuk basis klien membuat metode autentikasi ini sangat mahal untuk situs besar. Namun, situs yang memiliki data yang sangat sensitif atau basis pengguna terbatas, seperti yang umum dengan aplikasi bisnis-ke-bisnis (B2B) akan mendapatkan keuntungan besar dari penggunaan sertifikat.

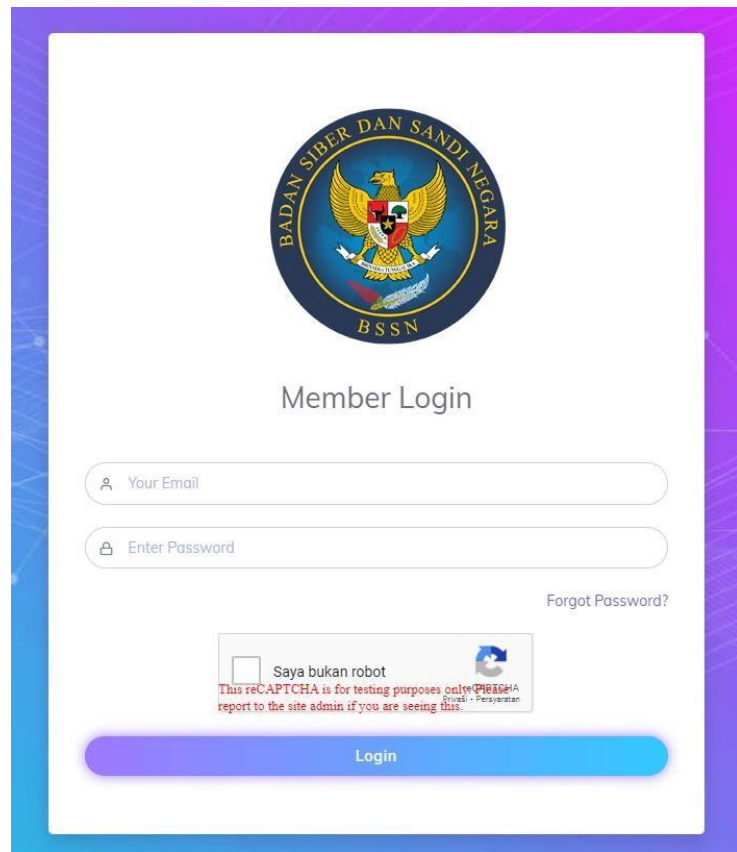
#### 3.5.1.6. **Forms-Based Authentication**

Autentikasi berbasis *form* adalah metode yang banyak digunakan untuk sebuah aplikasi berbasis web yang digunakan oleh banyak pengguna dan tidak bergantung pada fitur apa pun dari *protocol*/WEB standar dan tidak ada standar untuk melakukan autentikasi berbasis formular. Ini adalah autentikasi yang mekanismenya sangat dapat disesuaikan menggunakan formulir, biasanya terdiri dari HTML dengan <FORM> dan

<INPUT> tag yang menggambarkan bidang bagi pengguna untuk memasukkan informasi nama pengguna / sandi mereka. Setelah data dimasukkan melalui HTTP (atau SSL), itu dievaluasi oleh beberapa logika di sisi server dan, jika kredensial valid, beberapa jenis token diberikan ke browser klien untuk digunakan kembali untuk permintaan selanjutnya.

Pada autentikasi berbasis form biasanya aplikasi akan melindungi sumber daya yang ada pada sisi server dan menggunakan *cookies* pada sisi *client* dan *session* pada sisi *server* untuk melakukan validasi terhadap setiap permintaan sumber daya.

Contoh sebuah halaman autentikasi berbasis *form* seperti terlihat pada Gambar 22 di bawah ini.



Gambar 22 Autentikasi Berbasis Form

### 3.5.2 **Pengujian** Autentikasi

Dalam pengujian eksploitasi autentikasi bisa digunakan beberapa Teknik serangan sesuai dengan jenis autentikasi yang digunakan pada aplikasi berbasis WEB.

Beberapa Teknik serangan untuk autentikasi adalah :

- *Password Guessing*
- *Session ID Prediction and Brute Forcing*
- *Subverting Cookies*
- *Bypassing SQL-Backed Login Forms*

#### 3.5.2.1. **Password Guessing**

Serangan menebak kata sandi adalah teknik yang paling efektif dalam menguji keamanan autentikasi aplikasi



*bruteforce attack* akan mencoba kombinasi antara huruf, angka dan juga karakter spesial sesuai dengan aturan yang dibuat oleh penguji [8].

## Pengujian Password Guessing

Pengujian *password guessing* secara *dictionary attack* bisa menggunakan alat OWASP ZAP

New Fetch! Progress [1 HTTP://http://www.trendmicro.com/] Login Log					Current Issues 9			
Message Sent 20	Errors 0	Shore Errors						
Txn ID	Message Type	Code	Reason	RTT	Size Resp. Header	Size Req. Body	State	Periods
14	Fuzzed	200 OK	OK	64 ms	358 bytes	4,275 bytes	Reflected	admin_password
9	Original	200 OK	OK	19 ms	358 bytes	4,275 bytes	Reflected	
1	Fuzzed	200 OK	OK	41 ms	358 bytes	4,275 bytes	Medium	
2	Fuzzed	200 OK	OK	36 ms	358 bytes	4,237 bytes	Reflected	user: user
3	Fuzzed	200 OK	OK	19 ms	358 bytes	4,237 bytes	Reflected	user: admin
4	Fuzzed	200 OK	OK	19 ms	358 bytes	4,237 bytes	Reflected	user: password
5	Fuzzed	200 OK	OK	62 ms	358 bytes	4,237 bytes	Reflected	user: 123456
6	Fuzzed	200 OK	OK	50 ms	358 bytes	4,237 bytes	Reflected	guest: user
7	Fuzzed	200 OK	OK	57 ms	358 bytes	4,237 bytes	Reflected	guest: admin
8	Fuzzed	200 OK	OK	51 ms	358 bytes	4,237 bytes	Reflected	guest: password
10	Fuzzed	200 OK	OK	43 ms	358 bytes	4,237 bytes	Reflected	guest: 123456
11	Fuzzed	200 OK	OK	32 ms	358 bytes	4,237 bytes	Reflected	admin: user
12	Fuzzed	200 OK	OK	30 ms	358 bytes	4,237 bytes	Reflected	admin: guest
13	Fuzzed	200 OK	OK	32 ms	358 bytes	4,237 bytes	Reflected	admin: admin
15	Fuzzed	200 OK	OK	40 ms	358 bytes	4,237 bytes	Reflected	admin: 123456
16	Fuzzed	200 OK	OK	35 ms	358 bytes	4,237 bytes	Reflected	user

Gambar 23 Hasil *Password Guessing*

Pada gambar 23 di atas terlihat hasil serangan *password guessing* mendapatkan hasil sukses dengan *username admin* dan *password* adalah *password*



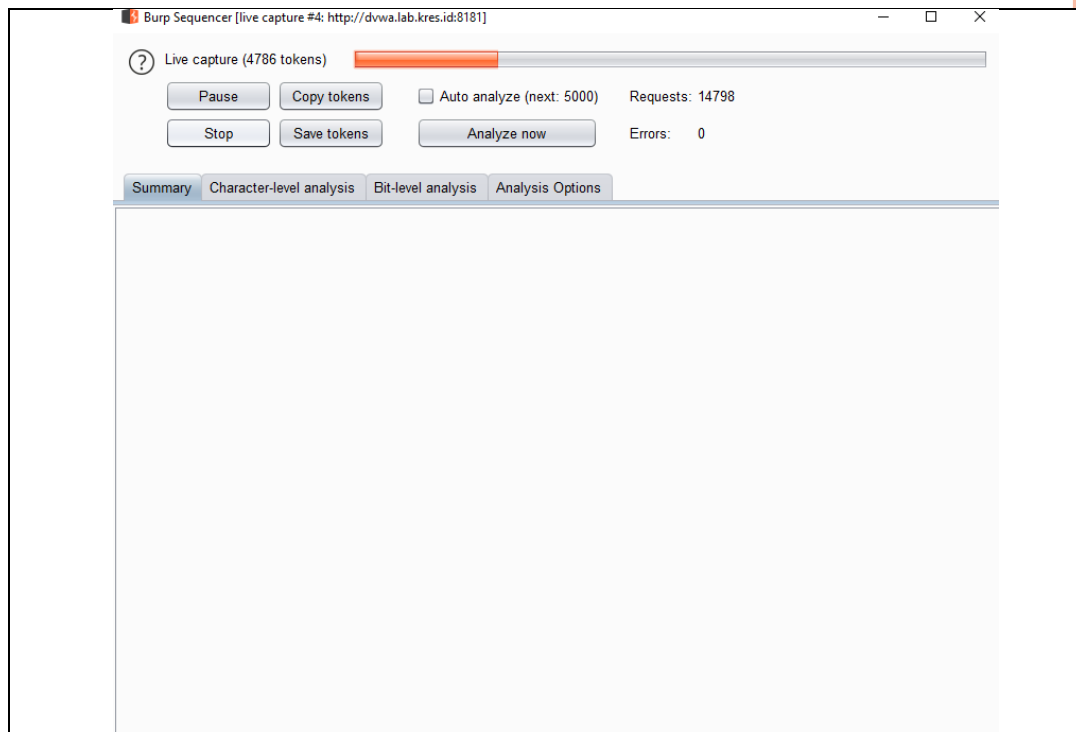
#### 3.5.2.2. *Session ID Prediction and Brute Forcing*

Banyak aplikasi web mengelola autentikasi dengan menggunakan sesi pengenalan (ID sesi). Oleh karena itu, jika pembuatan ID sesi dapat diprediksi, pengguna yang berniat jahat dapat menemukan ID sesi yang valid dan mendapatkan akses tidak sah ke aplikasi, meniru seorang pengguna yang sebelumnya diautentikasi. Alat yang bisa digunakan untuk melakukan prediksi *Session ID* dan melakukan *bruteforce* adalah Burp Suite.

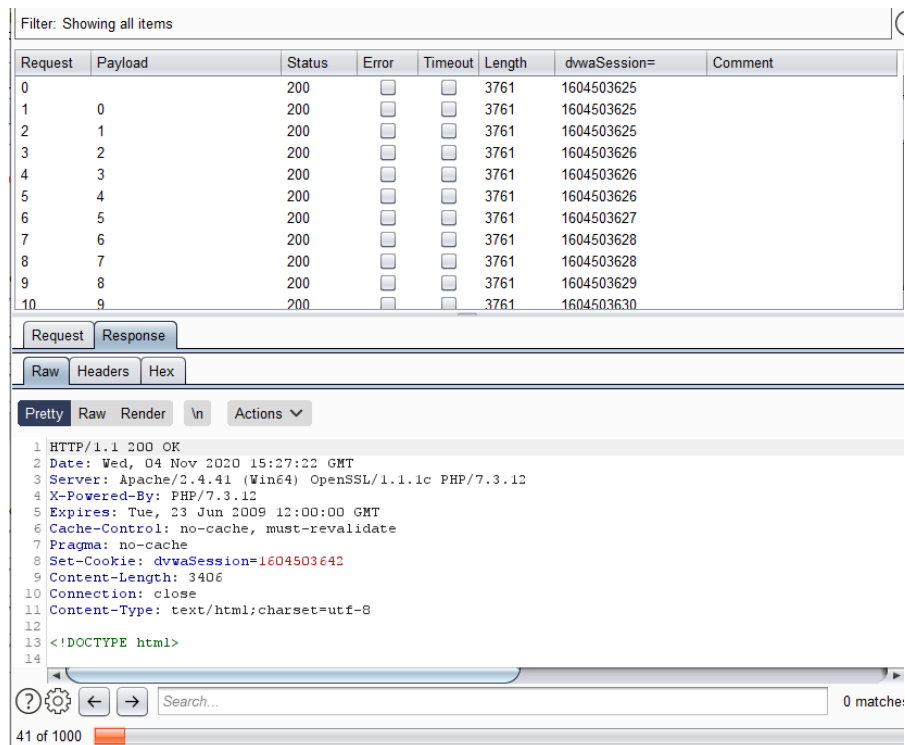
##### Melakukan Prediksi Session ID

Dalam melakukan prediksi Session ID bisa menggunakan Burp Suite<sup>[9]</sup> dengan fitur Burp Sequencer. Burp sequencer akan melakukan Analisa terhadap token yang muncul hasil *response* dari server dan akan mencari mekanisme yang digunakan untuk melakukan pembentukan sebuah *session ID*.

## DRAFT PANDUAN TEKNIS PELAKSANAAN IT SECURITY ASSESSMENT (ITSA)



Gambar 24 Burp Session Sequencer



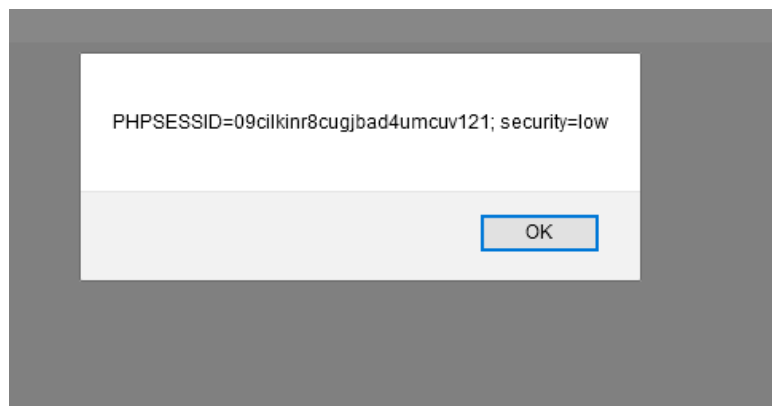
Gambar 25 Burp Session ID Intruder



### 3.5.2.3. *Subverting Cookies*

*Cookies* biasanya berisi data sensitif yang terkait dengan autentikasi. Jika *cookie* berisi kata sandi atau pengenal sesi, mencuri *cookie* bisa menjadi serangan yang sangat sukses untuk menguji keamanan situs Web. Ada beberapa teknik umum yang digunakan untuk mencuri *cookie*, dengan yang paling populer adalah *script injection* dan eavesdropping.

Injeksi skrip adalah serangan yang menyuntikkan skrip pada sisi klien ke dalam browser dan menjalankannya kode di sisi klien agar mengirimkan *cookie* ke peretas. Serangan ini cukup unik karena menggunakan kelemahan di situs Web untuk menyerang browser, bukan situs Web. Serangan tersebut bekerja dengan menyuntikkan skrip sisi klien, biasanya JavaScript, ke dalam situs Web. JavaScript berbahaya berisi kode untuk mengirim *cookie* ke peretas yang dijalankan oleh peramban, dan peretas sekarang dapat menggunakan *cookie* ini untuk "mencatat" tanpa menggunakan nama pengguna atau sandi. Biasanya Teknik ini memanfaatkan celah keamanan *Cross Site Scripting* (XSS).



Gambar 26 Contoh XSS Menampilkan *Cookies*

#### 3.5.2.4. *Bypassing SQL-Backed Login Forms*

Pada aplikasi Web yang melakukan autentikasi berbasis formulir dengan *back-end* SQL, injeksi SQL dapat digunakan untuk melewati autentikasi. Banyak aplikasi berbasis Web menggunakan *database* untuk menyimpan kata sandi dan menggunakan SQL untuk meminta *database* melakukan memvalidasi autentikasi kredensial. Pernyataan SQL umum akan terlihat seperti berikut :

```
SELECT * from AUTHENTICATIONTABLE WHERE Username = 'username input' AND Password =
```

Jika validasi *input* tidak dilakukan dengan benar, pengguna dapat memasukkan:

```
Username' --
```

Pada *field username* akan mengubah pernyataan SQL menjadi:

```
SELECT * from AUTHENTICATIONTABLE WHERE Username = 'Username' -- AND Password =
```

Tanda hubung (--) di akhir pernyataan SQL menentukan bahwa pernyataan SQL lainnya adalah komentardan harus diabaikan. Pernyataan tersebut setara dengan:

```
SELECT * from AUTHENTICATIONTABLE WHERE Username =
```

Pada pernyataan SQL *dikatas* menghilangkan pemeriksaan password. Ini adalah serangan umum yang tidak memerlukan banyak penyesuaian berdasarkan aplikasi Web, seperti halnya banyak serangan lain untuk autentikasi berbasis Formulir. Untuk melakukan serangan satu tingkat lebih tinggi, injeksi SQL (*SQL Injection*) dapat dilakukan pada *field password* juga. Dengan asumsi pernyataan SQL yang sama digunakan, menggunakan kata sandi untuk melakukan *bypass login* menggunakan *input* seperti berikut :

```
DUMMYPASSWORD' OR 1 = 1 --
```

Akan memiliki pernyataan SQL sebagai berikut :

Penambahan OR 1 = 1 di akhir pernyataan SQL akan selalu dievaluasi

```
SELECT * from AUTHENTICATIONTABLE WHERE Username = 'Username' AND Password =
```

sebagai benar, dan autentikasi sekalilagi dapat dilewati.

### 3.6 Authorization

Otorisasi adalah konsep yang mengizinkan akses ke sumber daya hanya untuk yang diizinkan untuk dapat menggunakannya. Menguji Otorisasi berarti pemahaman bagaimana proses otorisasi bekerja, dan menggunakan informasi itu untuk menghindari mekanisme otorisasi. Otorisasi adalah proses yang muncul setelah autentikasi berhasil, jadi penguji akan memverifikasi poin ini setelah dia memegang kredensial yang valid, terkait dengan serangkaian peran dan hak istimewa yang ditentukan dengan baik. Selama ini jenis penilaian, itu harus diverifikasi jika memungkinkan untuk melewati skema otorisasi, temukan jalur kerentanan traversal, atau temukan cara untuk meningkatkan hak istimewa yang diberikan kepada penguji.

Otorisasi juga dapat diserang di tingkat server Web. Dalam hal ini, Web server itu sendiri mungkin salah dalam melakukan konfigurasi dan mengizinkan akses ke *file* di luar dokumen Web *root*. *File-file* ini dapat berisi informasi konfigurasi sensitif, termasuk kata sandi. Jenis serangan otorisasi lainnya adalah melihat kode sumber halaman sebagai kebalikannya ke keluaran yang dihasilkan secara dinamis. Mendapatkan akses di luar *root* dokumen Web mungkin semudah menggunakan karakter traversal direktori (*../* *..*). Melihat kode sumber mungkin seperti sesederhana mengirim sufiks yang dikodekan dalam URL, seperti dalam kasus mesin servlet yang salah tangani *".Js% 70"*. Bagaimanapun, tujuannya adalah untuk mengakses informasi yang dibatasi.

Sekarang kita tahu apa yang ingin kita capai, bagaimana kita melakukan peretasan terhadap otorisasi? Tekniknya sebenarnya cukup sederhana, satu-satunya tangkapan adalah jika aplikasi memungkinkan itu atau tidak. Pada dasarnya Anda perlu menanyakan server Web, "Tunjukkan data untuk akun X!" Jika Aplikasi web tidak dirancang dengan benar, dengan



senang hati ia akan memberikan informasi tersebut. Ada beberapa konsep yang perlu diingat saat menguji kontrol akses aplikasi antara lain :

- **Eskalasi Hak Istimewa Horizontal**  
Mengakses informasi pengguna lain. Untuk Misalnya, aplikasi perbankan *online* mungkin mengontrol akses berdasarkan nomor rekening pengguna. Nomor rekening dapat diubah secara berurutan untuk melihat akun orang lain, tetapi mengelola aplikasi (seperti membuat, menghapus, atau mengubah akun) akan membutuhkan eksploitasi yang berbeda. Serangan ini menargetkan fungsionalitas yang tersedia untuk tingkat pengguna, tetapi terhadap data yang dibatasi.
- **Eskalasi Hak Istimewa Vertikal**  
Mengakses informasi pengguna yang lebih tinggi. Untuk Misalnya, aplikasi mungkin memiliki kerentanan dalam manajemen sesi yang memungkinkan Anda untuk masuk ke bagian administrator. Atau sandi administrator mudah untuk ditebak. Serangan ini menargetkan fungsionalitas dan data tidak tersedia untuk tingkat pengguna.
- **Akses *File* Sewenang-wenang**  
Biasanya, termasuk *file*, *file* dengan kredensial *database*, atau *file* di luar *root* dokumen Web dibatasi dari pengguna aplikasi. Serangan ini dengan cara memasukkan parameter yang berbeda dan dikombinasikan dengan kesalahan konfigurasi pada server dapat mengizinkan pengguna jahat untuk mengakses *file-file* ini. Biasanya serangan ini menargetkan server Web, tetapi aplikasi yang menggunakan metode pembuatan *template* yang tidak aman membuat kerentanan bisa ada dalam aplikasi juga.

Setiap jenis akses hak istimewa menggunakan metode pengujian yang sama. Jika otorisasi ke pengguna lain informasi profil pengguna dapat diperoleh dengan mengubah nilai *cookie* atau mungkin juga nilai itu untuk mendapatkan hak administrator dari nilai yang sama. Di kesempatan lain, kontrol hak istimewa berbasis peran aplikasi mungkin memblokir eskalasi

vertikal. Rinciannya data yang akan diubah di setiap permintaan akan berbeda dari satu aplikasi ke aplikasi lain, tetapi *file* tempat untuk melihat informasi selalusama.

### 3.6.1 **Matrik Peran (Role Matrix)**

Informasi yang berguna untuk membantu proses uji otorisasi adalah matriks peran. Sebuah matriks peran berisi daftar semua pengguna (atau tipe pengguna) dalam aplikasi dan tindakan mereka yang sesuai. Ide dari matriks ini bukan untuk memberi tanda centang untuk setiap tindakan yang diizinkan, tetapi untuk mencatat catatan tentang bagaimana tindakan dijalankan dan token sesi apa yang diperlukan tindakan tersebut. Contoh matrik peran dapat dilihat pada tabel 1.

Matriks peran mirip dengan peta fungsionalitas. Saat kami menyertakan URI itu masing-masing akses pengguna untuk fungsi tertentu, lalu pola mungkin muncul. Contoh pada Tabel 1 mungkin tampak terlalu sederhana, tetapi perhatikan bagaimana administrator memandang yang lain.

Tabel 1 Matrik Peran

Role	User	Admin
Melihat profil diri sendiri	/profile/view.php?UID=100	/profile/view.php?UID=1
Modifikasi profil diri sendiri	/profile/update.php?UID=100	/profile/update.php?UID=1
Melihat profil pengguna lain	n/a	/profile/view.php?UID=1 &EUID=100
Menghapus pengguna	n/a	/admin/deluser.php?id=100



### 3.6.2 **Pengujian Otorisasi**

Banyak hal yang perlu Anda ketahui untuk melakukan serangan terhadap otorisasi. Duplikasi situs dan analisis situs Web akan membantu dalam menentukan bagaimana mengubah permintaan HTTP untuk menguji keamanan aplikasi. Secara umum apabila Anda ingin mengubah *field input* yang berhubungan dengan id pengguna, nama pengguna, grup akses, nama *file*, pengenalan *file*, dan sebagainya. Lokasi *field* ini bergantung pada aplikasi. Tapi dalam protokol HTTP, hanya ada beberapa *field* tempat nilai-nilai ini dapat diteruskan.

*Field* tersebut adalah *cookie*, *string* kueri, data dalam permintaan POST, dan *tag* tersembunyi. Otorisasi terjadi setiap kali aplikasi menarik data dari *database* atau mengakses halaman Web. Dapatkah pengguna mengakses informasi tersebut? Bagaimana aplikasinya mengidentifikasi pengguna (apakah berdasarkan autentikasi, URL, manajemen sesi)? .

Skenario yang mungkin untuk serangan otorisasi akan tumbuh dengan jumlah fungsionalitas dalam aplikasi. Untuk berhasil meluncurkan serangan eskalasi hak istimewa, Anda perlu mengidentifikasi komponen aplikasi dan melacak identitas atau peran pengguna. Ini mungkin sederhana mencari nama pengguna Anda di salah satu lokasi berikut, atau skema otorisasi mungkin didasarkan pada nilai tertentu yang ditetapkan oleh server.

#### 3.6.1.1. **Kueri String**

*Kueri string* adalah bit data tambahan di URI setelah tanda tanya (?) Yang digunakan untuk melewati variabel. *String* kueri digunakan untuk mentransfer data antara klien dan server. Itu adalah daftar yang dipisahkan dengan *ampersand* dan dapat berisi beberapa nilai data. Contohnya sebagai berikut

```
https://dvwa.lab.kres.id/mail.php?mailbox=joe&msg=10
```



Pada contoh di atas Di dalam kueri string adalah *mailbox=joe&msg=10* . Kueri string terlihat di *address bar* pada browser, dan mudah diubah tanpa alat khusus.

Hal yang dapat dicoba adalah mengubah URI menjadi

```
https://dvwa.lab.kres.id/mail.php?mailbox=jane&msg=10
```

Jika dengan autentikasi *joe* dapat melihat mailbox milik *jane* maka ini merupakan sebuah kerentanan dari otorisasi.

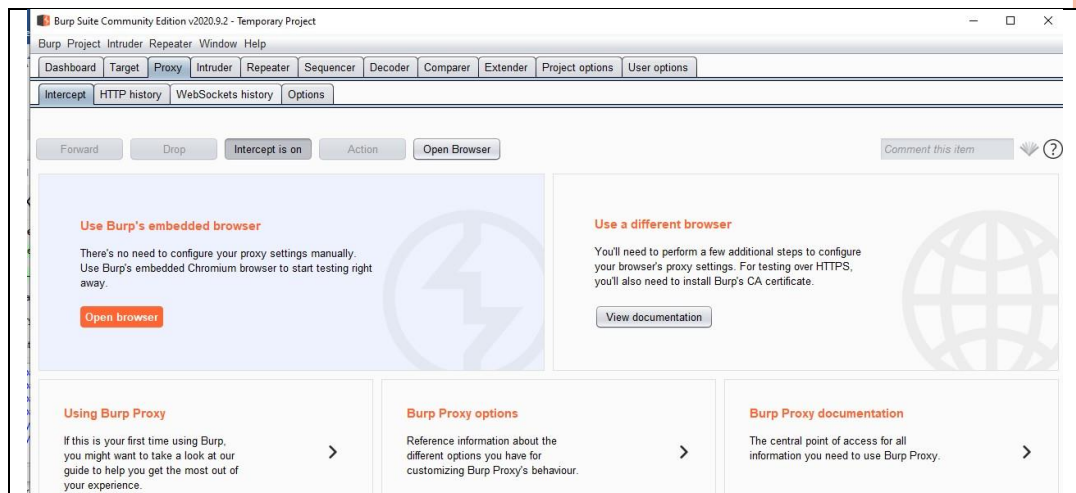
#### 3.6.1.2. **POST Data**

Karena *kueri string* pada browser sangat mudah dimodifikasi, banyak pemrogram aplikasi berbasis Web lebih suka menggunakan metode POST daripada GET dengan *kueri string* Ini biasanya melibatkan penggunaan formulir. Ada beberapa Teknik untuk mengubah nilai-nilai metode POST. Teknik paling dasar melibatkan penyimpanan halaman HTML, memodifikasi sumber HTML, dan mem-*posting* permintaan palsu. Untuk mempermudah modifikasi POST data bisa menggunakan alat berbasis *proxy* sebagai contoh Burp Suite dan OWASP ZAP yang akan memungkinkan mengubah data ini dengan cepat.

##### Melakukan intersepsi POST data

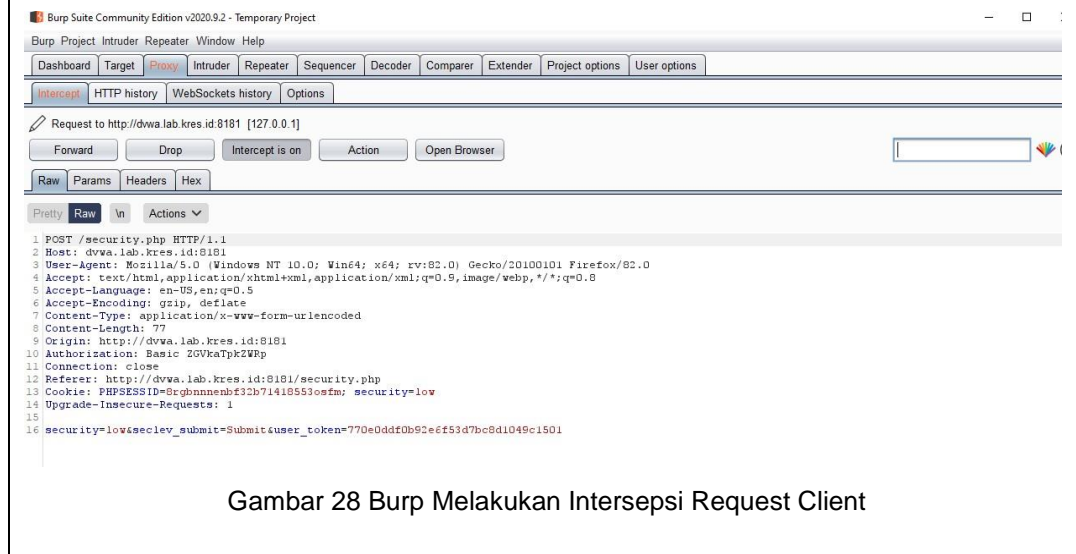
Melakukan intersepsi POST data bisa menggunakan Burp Suite dengan fitur Proxy dengan mengaktifkan fitur *intercept is on*

## DRAFT PANDUAN TEKNIS PELAKSANAAN IT SECURITY ASSESSMENT (ITSA)



Gambar 27 Burp Proxy Intercept

Jika sudah mengaktifkan fitur *intercept* pada burp , selanjut burp akan selalu melakukan *intercept* terhadap setiap *request* dari klien. Kita bisa melakukan modifikasi terhadap parameter yang ada pada POST data untuk mencoba otorisasi.



Gambar 28 Burp Melakukan Intersepsi Request Client

### 3.6.1.3. *Hidden Tags*

*Hidden tags* sering digunakan untuk melacak sesi, penyertaan yang diperlukan oleh aplikasi. Beberapa situs menggunakan *tag* tersembunyi untuk parameter yang tidak di masukan oleh pengguna. Meskipun *tag* tersembunyi disembunyikan dari pengguna yang melihat situs web melalui browser, *tag* tersembunyi masih terlihat di sumber HTML dari halaman web.

*Tag* tersembunyi adalah bagian dari formulir HTTP, jadi Anda akan melihat nilainya diteruskan dalam permintaan GET atau POST. Anda masih harus mencari *tag* yang sebenarnya, karena *field* tersebut nama atau komentar HTML dapat memberikan petunjuk tambahan untuk fungsi *tag*.

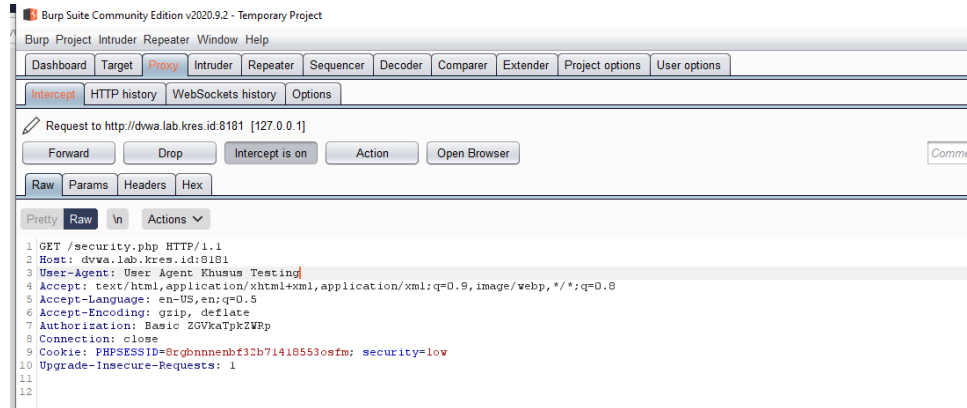
```
<form action="#" method="POST">
  <p>Security level is currently: <em>low</em>.<p>
  <p>You can set the security level to low, medium, high or impossible. The security level changes the vulnerability level of I
  <ol>
    <li> Low - This security level is completely vulnerable and <em>has no security measures at all</em>. It's use is to be a
    <li> Medium - This setting is mainly to give an example to the user of <em>bad security practices</em>, where the develop
    <li> High - This option is an extension to the medium difficulty, with a mixture of <em>harder or alternative bad practic
    <li> Impossible - This level should be <em>secure against all vulnerabilities</em>. It is used to compare the vulnerable
      Prior to DVWA v1.9, this level was known as 'high'.</li>
  </ol>
  <select name="security">
    <option value="low" selected="selected">Low</option><option value="medium">Medium</option><option value="high">High</opti
  </select>
  <input type="submit" value="Submit" name="seclav_submit">
  <input type="hidden" name="user_token" value="0fb9ffea019053bb12c33ea39f076f78" />
</form>
```

Gambar 29 Contoh Hidden Tag pada Source HTML

### 3.6.1.4. *HTTP Header*

*Header* HTTP biasanya tidak digunakan oleh aplikasi Web yang bekerja dengan browser Web. Mereka terkadang digunakan dengan aplikasi yang memiliki desktop yang menggunakan protokol HTTP. Bagian ini pun dapat dimodifikasi. *Cookies* mungkin yang paling banyak *header* yang terkenal, tetapi skema otorisasi juga dapat didasarkan pada "Lokasi:" dan *Header* "Referer." Aplikasi mungkin juga mengandalkan tajak khusus untuk melacak atribut tertentu dari pengguna. Salah satu tes otorisasi yang paling sederhana untuk diatasi adalah pemeriksaan browser. Burp bisa digunakan untuk melakukan modifikasi HTTP Header Jadi, jika aplikasi memerlukan Internet Explorer untuk alasan politik dan bukan karena alasan teknis (seperti

memerlukan komponen ActiveX tertentu), Anda dapat mengubah *header* ini untuk meniru IE.



Gambar 30 Melakukan Perubahan HTTP *Header*

### 3.6.1.5. Cookies

*Cookie* adalah bentuk manajemen sesi yang populer meskipun penggunaan *cookie* telah banyak ditemukan kerentanan keamanan. Namun, penggunaannya masih umum dan *cookie* sering digunakan untuk menyimpan *field* penting seperti nama pengguna dan nomor akun. *Cookie* dapat digunakan untuk menyimpan hampir semua data, dan semua bidang dapat dimodifikasi dengan mudah menggunakan program seperti Burp Suite.

### 3.6.1.6. URI

*Universal Resource Identifier* (URI) adalah *string* di bilah Lokasi browser. URI akan terdiri dari nama *hostserver* Web, bersama dengan *file* yang akan diambil. Dengan hanya memodifikasi nama *file* dan URI, terkadang seorang peretas dapat mengambil *file* yang biasanya tidak dapat mereka akses. Misalnya, situs mungkin memiliki tautan ke

```
http://www.reports.com/data/report12345.txt
```

setelah Anda membayar untuk akses ke laporan itu. Melihat URI dari sudut pandang peretas, Anda akan mencoba mengakses

```
http://www.reports.com/data/report12346.txt.
```

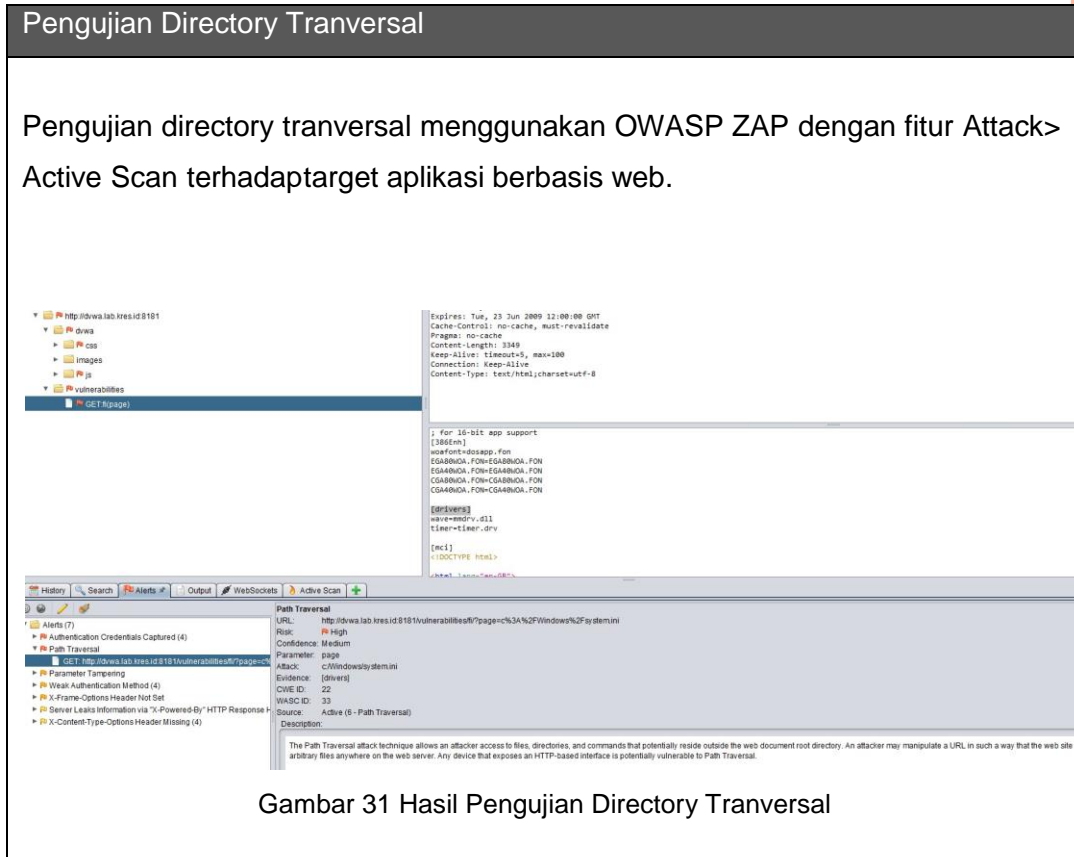
Direktori Traversal adalah contoh lain dari melewati aplikasi atau server Web yang mempunyai skema otorisasi. Serangan Direktori Traversal *Unicode* yang dipublikasikan pertama kali untuk IIS memanfaatkan kelemahan dalam mesin penguraian dan otorisasi server. Biasanya, IIS memblokir upaya untuk keluar dari *root* dokumen Web dengan URI seperti

```
"/Scripts/../../../../winnt".
```

Representasi *Unicode* untuk garis miring (/) adalah "% c0% af". IIS tidak menafsirkan representasi *Unicode* selama pemeriksaan otorisasinya, yang memungkinkan pengguna yang berniat jahat untuk keluar dari *root* dokumen dengan URI seperti

```
"/ scripts /..% c0% af ..% c0% af ..% c0% afwinnt ".
```

Hampir semua alat untuk melakukan pencarian kerentanan pada aplikasi bisa mendeteksi kerentanan ini. Salah satu contoh bisa menggunakan Burp Suite dan OWASP ZAP.



### 3.7 Session Management

Protokol HTTP tidak menentukan bagaimana sesi pengguna harus dikelola dan dilacak untuk aplikasi berbasis Web. Protokol dirancang untuk pengambilan dokumen sederhana dan bukan untuk aplikasi Web kompleks yang umum saat ini.

Sebagai contoh, jika Anda pergi ke <https://www.acme.com> dan ingin membeli ACME *Roadrunner* Trap terbaru 2000 dan klik pada tombol Beli, keranjang belanja dan pemrosesan order akan dilakukan berbeda dari toko *online* lainnya. Protokol itu sendiri tidak menentukan bagaimana melakukannya.

Alasan paling dasar untuk melacak sesi pengguna adalah untuk aplikasi yang membutuhkan pengguna untuk mengautentikasi. Setelah pengguna mengautentikasi, server harus dapat menerima berikutnya permintaan dari pengguna itu, tapi abaikan permintaan dari pengguna yang belum diautentikasi. Alasan lainnya adalah untuk aplikasi belanja *online*. Aplikasi

harus dapat menjawab pertanyaan seperti:

- Apa yang sedang dijelajahi pengguna?
- Apa yang dipilih pengguna untuk dibeli?
- Apa yang pengguna putuskan untuk tidak dibeli?
- Apakah pengguna siap untuk membeli?
- Apakah ini masih pengguna yang asli?

Apa artinya ini bagi seorang peretas? Jika Anda menyerahkannya kepada pengembang individu dan Perancang situs web untuk merancang solusi mereka sendiri untuk melakukan manajemen status sesi, mereka cenderung melakukan kesalahan yang menyebabkan masalah keamanan.

Pengembang situs web telah merancang sejumlah cara untuk melakukan Teknik untuk manajemen status sesi yang bekerja dalam kerangka protokol HTTP. Teknik-teknik ini cukup pintar; namun, tidak semuanya aman. Perbedaan utama antara teknik dari perspektif keamanan adalah di mana status sesi dikelola, di klien atau server.

### 3.7.1 *Client-Side Techniques*

Beberapa informasi Umum yang dilacak selama sesi pengguna antara lain :

Tabel 2 Tabel Hasil Pelacakan Menggunakan Teknik *Client-Side*

Session Attribute	Keterangan
Username	Field yang agak jelas, tetapi terkadang digunakan untuk melacak file pengguna untuk menyesuaikan halaman. Misalnya, memasukkan "Selamat datang kembali, Tori!" saat pengguna masuk ke aplikasi.





User Identifier	Aplikasi web yang menggunakan database sering melacak pengguna dan memiliki beberapa bentuk indeks numerik yang mengidentifikasi secara unik pengguna. Dalam banyak kasus, ini bisa saja menjadi baris nomor dalam tabel database tempat informasi pengguna tersimpan.
User Role	Jenis pengguna apa yang mengakses aplikasi? Bisamereka melihat data? Ubah data? Kelola pengguna lain akun?
Session Identifier	Aplikasi atau server Web terkadang menetapkan nilai sesi yang valid untuk jangka waktu yang singkat

#### 3.7.1.1. *Hidden Fields*

*Field* tidak menyiratkan keamanan sesi yang buruk, tetapi ini bisa menjadi indikator. Mari kita lihat bagindari FORM yang diekstrak dari halaman *login* aplikasi.

```
<FORM name=login_form action=
https://login.victim.com/config/login?4rfr0naidr6d3
method=post >

<INPUT name=Tries type=hidden> <INPUT value=us
name=I8N type=hidden>

<INPUT name=Bypass type=hidden> <INPUT
value=64mbvjoubpd06 name=U type=hidden> <INPUT
value=pVjsXMKjKD8rlggZTYDLWwNY_Wlt
name=Challenge type=hidden>

User Name:<INPUT name=Login>

Password:<INPUT type=password maxLength=32 value=""
name=Passwd>
```



Saat pengguna memasukkan nama pengguna dan kata sandinya, dia sebenarnya mengirimkan tujuh potongan informasi ke server meskipun hanya dua yang terlihat di halaman Web. Dari contoh di atas, tampak bahwa dua *field* tersembunyi melacak informasi status, “*Tries*” dan “*U*”. Pada titik ini tidak jelas apakah kerentanan itu ada. Ingat, penguji perlu mengidentifikasi semua mekanisme autentikasi pada aplikasiterlebih dahulu.

### 3.7.1.2. *The URL*

Lihat lagi contoh FORM dari bagian sebelumnya. Ada lagi “*field*” tersembunyi di elemen *action* FORM:

```
<FORM name = login_form action = https://login.victim.com/config/login?4rfr0naidr6d3 method = post>
```

Variabel sesi tidak harus disetel dalam FORM untuk dilacak oleh aplikasi mereka. Server dapat mengatur parameter atau membuat pengalihan yang disesuaikan untuk pengguna tertentu.

Contoh lain mungkin terlihat seperti ini:

/redirect.html/103-6733477-6580661?

/ ViewBasket; \$ sid \$

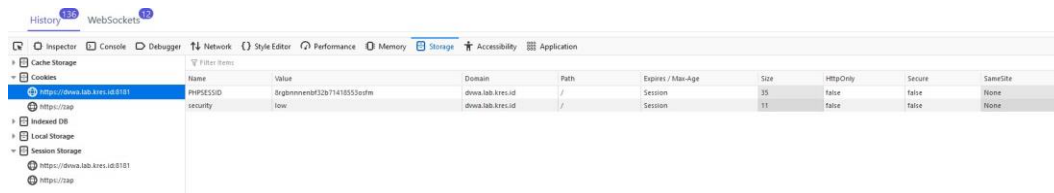
rO5J5l0EAACii6fr0eK2sQJnUEiakKFH?

/index.php?session\_id={E3E0FC4C-E5F7-48A4-8DD9-48FD08906D85}

Dalam kasus terakhir, nama “*session\_id*” memberikan ID sesi. Membawa ID sesi URL tidak secara umum tidak aman, tetapi ada beberapa hal yang perlu diingat. Pada HTTPS Jika ID sesi dapat diputar ulang dari komputer lain, maka seorang pengguna jahat dapat mengendus koneksi HTTP *cleartext* untuk menipu pengguna lain.

### 3.7.1.3. HTTP Headers and Cookies

HTTP *Header* dan *Cookie* mungkin merupakan lokasi paling umum untuk menyimpan informasi keadaan sesi pengguna. *Nonpersistent Cookie* digunakan untuk melacak status untuk satu sesi. PHP `PHPSESSID` nilai adalah contoh yang baik dari jenis *cookie* ini. Nilai *cookie* ini tidak pernah disimpan di komputer pengguna. Pada *Developer Tools* bisa digunakan melihat apakah *cookie* bersifat *persistent* apa tidak.



Gambar 32 Melihat Status Cookies

### 3.7.2 Server-Side Techniques

Server web modern memiliki kemampuan untuk menghasilkan sesi acak mereka sendiri (*random*) ID. ID yang dihasilkan oleh server ini cenderung sebesar (32 bit), angka acak ini menghalangi banyak jenis serangan, meskipun semuanya rentan terhadap tayangan ulang sesi (*session replay*) serangan. Tabel di bawah mencantumkan beberapa jenis yang umum dan pelacakan sesi yang sesuai variabel.

#### 3.7.1.4. Session Database

Aplikasi yang sangat bergantung pada *database* yang memiliki opsi untuk melacak sesi hampir sepenuhnya di sisi server. *Database* sesi adalah teknik pengelolaan yang sangat efektif sesi di beberapa server Web dengan cara yang aman. Server masih menghasilkan *file* nomor unik dan meneruskannya ke klien; namun, tidak ada informasi tambahan yang keluar dari server.



Saat pengguna pertama kali masuk ke aplikasi, aplikasi menghasilkan sesi sementara. Ini menyimpan ID dalam tabel sesi. Semua informasi keadaan disimpan di baris yang sama dengan ID sesi. Setiap kali pengguna meminta halaman baru, aplikasi mengambil sesi token dan mencari nilai dalam tabel sesinya. Selama ID sesi valid, aplikasi tersebut mengambil informasi status saat ini dari baris di tabel sesi.

Keuntungan dari *database* sesi adalah hanya satu nilai yang perlu diteruskan ke klien. Informasi keadaan bagian tidak dapat diendus, dipalsukan, atau dimodifikasi. Rutinitas lain di *database* dapat mengumpulkan tabel secara berkala dan secara otomatis mengakhiri ID sesi yang telah ada digunakan untuk waktu yang lama.

Tabel di bawah adalah *variable session* yang umum di hasilkan oleh *application server*.

Tabel 3 Variable Session

Application Server	Session ID Variable
IIS	ASPSESSIONID
Tomcat (Servlet/JSP engine)	JSESSIONID
User Role	Jenis pengguna apa yang mengakses aplikasi? Bisamereka melihat data? Ubah data? Kelola pengguna lain akun?
PHP	PHPSESSID
ColdFusion	CFID

### 3.7.3 **Pengujian Session Management**

Beberapa Teknik yang umum dapat digunakan untuk menguji *session management* antara lain :

- *Session Fixation*
- *Session Fuzzing*

#### 3.7.3.1 **Session Fixation**

Ketika aplikasi tidak memperbarui *cookie* sesinya setelah sebuah autentikasi pengguna yang sukses, dimungkinkan untuk menemukan sesi kerentanan *fixation* dan memaksa pengguna lain untuk menggunakan *cookie* yang dikenal penyerang. Dalam hal ini, penyerang dapat mencuri sesi pengguna (*session hijacking*).

Kerentanan fiksasi sesi terjadi ketika:

- Aplikasi web mengautentikasi pengguna tanpa membatalkannya terlebih dahulu ID sesi yang ada, dengan demikian akan terus menggunakan ID sesi sudah terkait dengan pengguna tersebut.
- Seorang penyerang dapat memaksa ID sesi yang diketahui pada pengguna bahwa, setelah pengguna mengautentikasi, penyerang memiliki akses ke *file* sesi terautentikasi.

Dalam eksploitasi yang umum pada kerentanan *session fixation*, penyerang membuat sesi baru di aplikasi web dan mencatat yang terkait pengenalan sesi. Penyerang kemudian menyebabkan korbannya autentikasi terhadap server menggunakan pengenalan sesi yang sama, memberikan penyerang akses ke akun pengguna yang aktif.

### 3.8 ***Input Validation***

Serangan validasi *input* mencoba mengirimkan data yang tidak diharapkan oleh aplikasi yang menerima. Biasanya, aplikasi akan melakukan beberapa jenis pemeriksaan kesesuaian pada *input* pengguna. Pemeriksaan ini mencoba memastikan bahwa data berguna. Diperlukan pemeriksaan yang lebih penting untuk mencegah data agar tidak merusak server. Pemeriksaan yang kurang ketat diperlukan jika data hanya dibatasi untuk panjang tertentu.

#### 3.8.1 **Mengharapkan yang Tidak Terduga**

Salah satu kegagalan terbesar dalam validasi *input* adalah menulis rutinitas dalam JavaScript dan menempatkannya di browser. Pada awalnya, mungkin terlihat diinginkan untuk menggunakan sisi klien apapun bahasa *scripting* untuk rutinitas validasi. Mereka mudah diterapkan dan banyak digunakan didukung antara browser Web (meskipun ada kebiasaan browser individu).

Jenis serangan validasi *input* biasanya termasuk dalam salah satu dari tiga kategori antara lain:

- **Masukan Tak Terduga**  
Ini termasuk *SQL Injection*, *XSS* dan semua *input* yang menghasilkan pesan *error* pada aplikasi.
- **Karakter Perintah Eksekusi (*Command Injection*)**  
Serangan ini khusus untuk memasukan perintah sistem, seperti memasukkan titik koma untuk menjalankan perintah arbitrer di sebuah Server Web UNIX.
- ***Buffer Overflows***  
Serangan *overflow* cenderung menjadi serangan yang paling sederhana untuk dieksekusi. Ini melibatkan melempar sebanyak mungkin terhadap satu variabel atau *field* dan perhatikan hasilnya. Hasilnya mungkin aplikasi macet atau bisa berakhir menjalankan perintah sewenang-wenang.



Efek serangan *input* validasi berkisar dari tidak berbahaya hingga membahayakan Web server. Serangan- serangan ini juga dapat dikategorikan berdasarkan tujuannya:

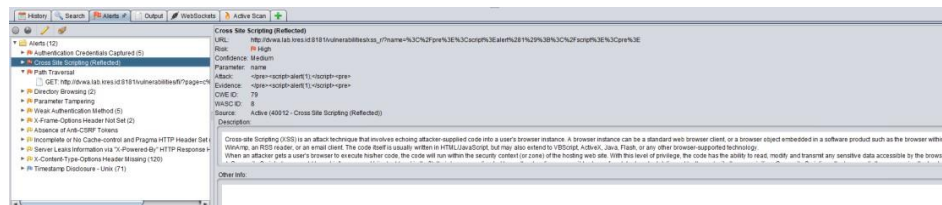
- Membangkitkan Kesalahan Informasi Aplikasi mungkin memberikan informasi tentang entri SQL(nama tabel, nama bidang).
- Kesalahan mengungkapkan direktori lengkap jalur (huruf *drive*, direktori *home*). Kesalahan dalam eksekusi halaman menyebabkan aplikasi untuk membuang kode sumber.
- Memperoleh Akses Data Sewenang-wenang Seorang pengguna mungkin dapat mengakses data untuk sesama pengguna, seperti satu pelanggan dapat melihat penagihan pelanggan lain informasi. Seorang pengguna mungkin dapat mengakses data yang memiliki hak istimewa, seperti *anonym* pengguna dapat menghitung, membuat, atau menghapus pengguna.
- Memperoleh Eksekusi Perintah Sewenang-wenang *Input* berisi perintah yang dijalankan oleh server, seperti mengambil kata sandi, mencantumkan direktori, atau menyalin *file*. Perintah lain dijalankan oleh aplikasi, seperti serangan SQL *injection*.
- *Cross-Site* atau *Embedded Scripting* Serangan ini adalah bagian dari serangan *social engineering* terhadap pengguna lain. Serangan lain menargetkan aplikasi sendiri, dengan tujuan menjalankan perintah sistem atau membaca *file* arbitrer.
- Pengujian validasi *input* adalah proses berulang. Anda memasukkan karakter yang tidak valid ke dalam *field* (atau vektor serangan lainnya) dan periksa hasilnya. Jika hasilnya *error*, lalu informasi apa apakah kesalahan terungkap? Komponen aplikasi apa yang menyebabkan kesalahan? Proses ini berlanjut sampai semua *input field* telah diperiksa.

### 3.8.2 Pengujian *Input Validation*

Hampir semua alat uji penetrasi dan pencarian kerentanan untuk aplikasi berbasis web bisa mengidentifikasi dan bisa digunakan untuk menguji *input validation*. Burp Suite dan OWASP ZAP bisa digunakan untuk melakukan serangan terhadap *input validation*.

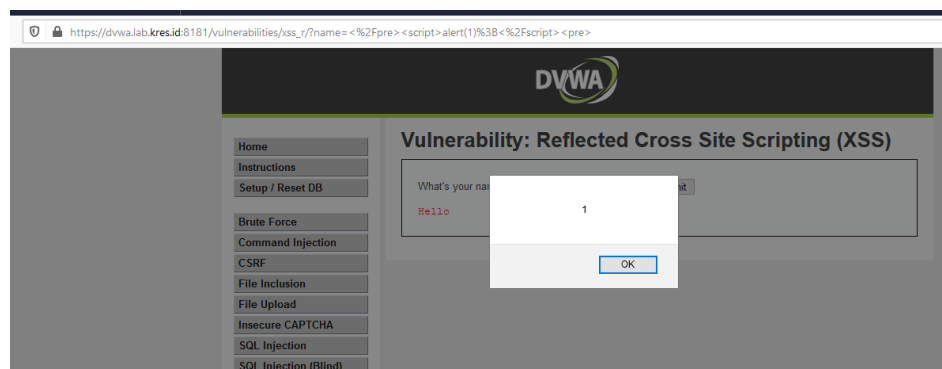
#### Pengujian Input Validasi

Pada OWASP ZAP gunakan fitur *attack > active scan* untuk melakukan pengujian terhadap *input* validasi target.



Gambar 33 OWASP ZAP Mendapatkan Kerentanan XSS

Pada gambar 33 OWASP ZAP mendapatkan kerentanan pada aplikasi berupa XSS, selanjutnya hasil dari *tool* di coba Kembali menggunakan browser untuk melihat tidak adanya *false positive* dari sebuah kerentanan yang ditemukan.

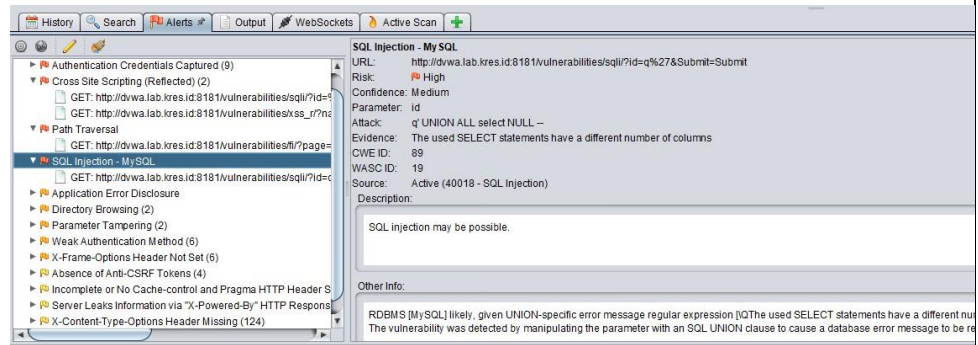


Gambar 34 Konfirmasi Kerentanan XSS



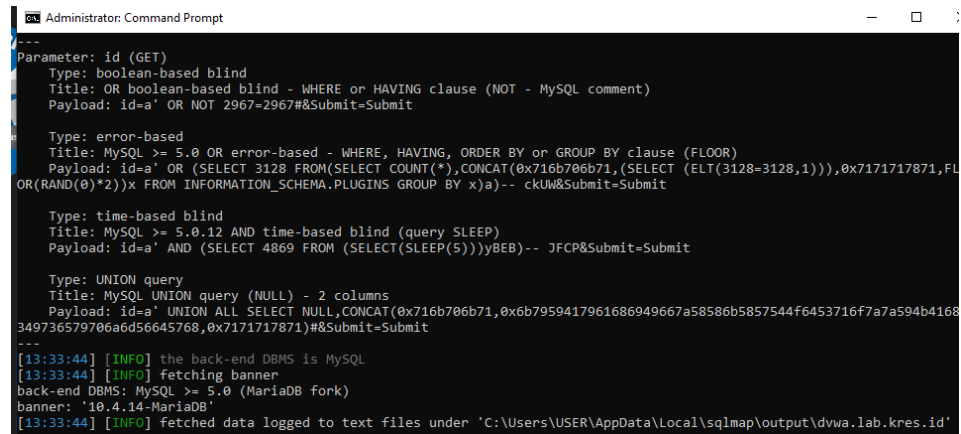
## Input Validasi – SQL Injection

Pada OWASP ZAP gunakan fitur attack > active scan untuk melakukan pengujian terhadap input validasi target.



Gambar 35 OWASP ZAP Menemukan Kerentanan SQL Injection

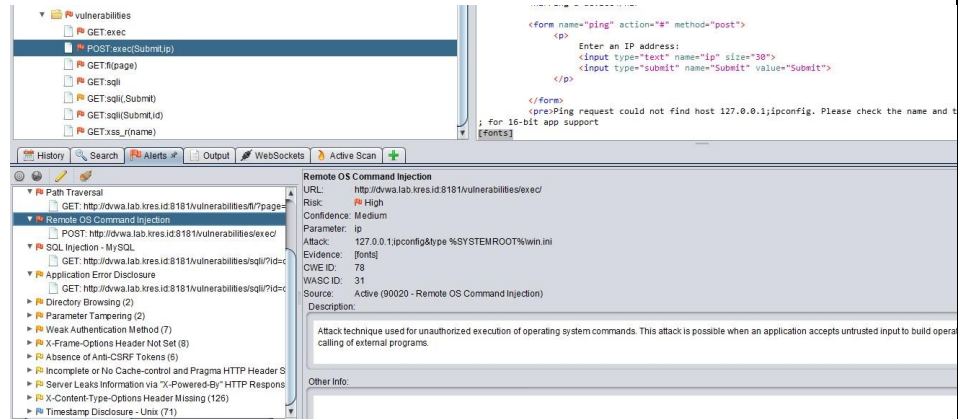
Pada gambar 35 OWASP ZAP mendapatkan kerentanan pada aplikasi berupa SQL, selanjutnya hasil dari *tool* di coba Kembali menggunakan SQLMAP<sup>[10]</sup> untuk melihat tidak adanya *false positive* dari sebuah kerentanan yang ditemukan.



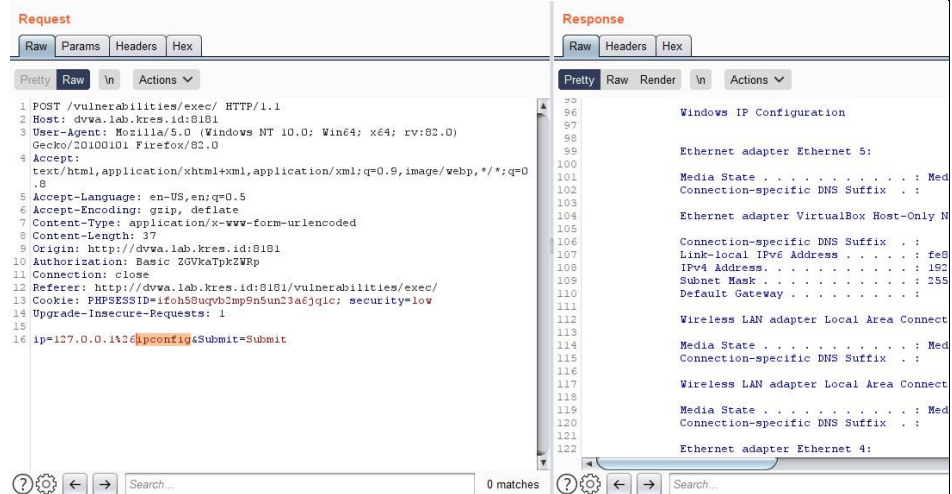
Gambar 36 Hasil Pengujian Kerentanan SQL Injection

### Pengujian Input Validasi – Command Injection

Pada OWASP ZAP gunakan fitur *attack > active scan* untuk melakukan pengujian terhadap *input* validasi target.



Gambar 37 OWASP ZAP Mendapatkan Kerentanan *Command Injection*



Gambar 38 Percobaan *Command Injection* Menggunakan Burp

### 3.9 Error Handling

Sering kali, selama uji penetrasi pada aplikasi berbasis web, banyak didapatkan kode kesalahan yang dihasilkan dari aplikasi atau server web. Kesalahan ini mungkin saja ditampilkan dengan menggunakan permintaan tertentu, baik yang dibuat khusus dengan alat bantu atau



dibuat secara manual. Kode-kode ini sangat berguna untuk selama kegiatan uji penetrasi, karena biasanya aplikasi mengungkapkan banyak informasi tentang *database*, *bug*, dan komponen teknologi lainnya terhubung langsung dengan aplikasi web.

Bagian ini menganalisis kode yang lebih umum (pesan kesalahan) dan memfokuskan relevansinya selama penilaian kerentanan. Aspek terpenting untuk kegiatan ini adalah fokus pada kesalahan yang dihasilkan oleh aplikasi, melihatnya sebagai kumpulan dari informasi yang akan membantu dalam langkah analisis pengujian selanjutnya, baik dapat memfasilitasi efisiensi penilaian dengan mengurangi keseluruhan waktu yang dibutuhkan untuk melakukan uji penetrasi.

Hampir semua alat uji penetrasi dan pencarian kerentanan untuk aplikasi berbasis web bisa mengidentifikasi dan bisa digunakan untuk menguji *error handling* dari aplikasi.

Terhadap 3 jenis sumber kesalahan (*error*) yang mungkin muncul pada saat melakukan uji penetrasi, di antaranya :

- *Web Server Errors*

Kesalahan umum yang dapat kita lihat selama pengujian adalah HTTP 404 atau sumber daya tidak ditemukan. Seringkali kode kesalahan ini memberikan detail yang berguna tentang server web yang mendasari dan komponen terkait.

- *Application Server Errors*

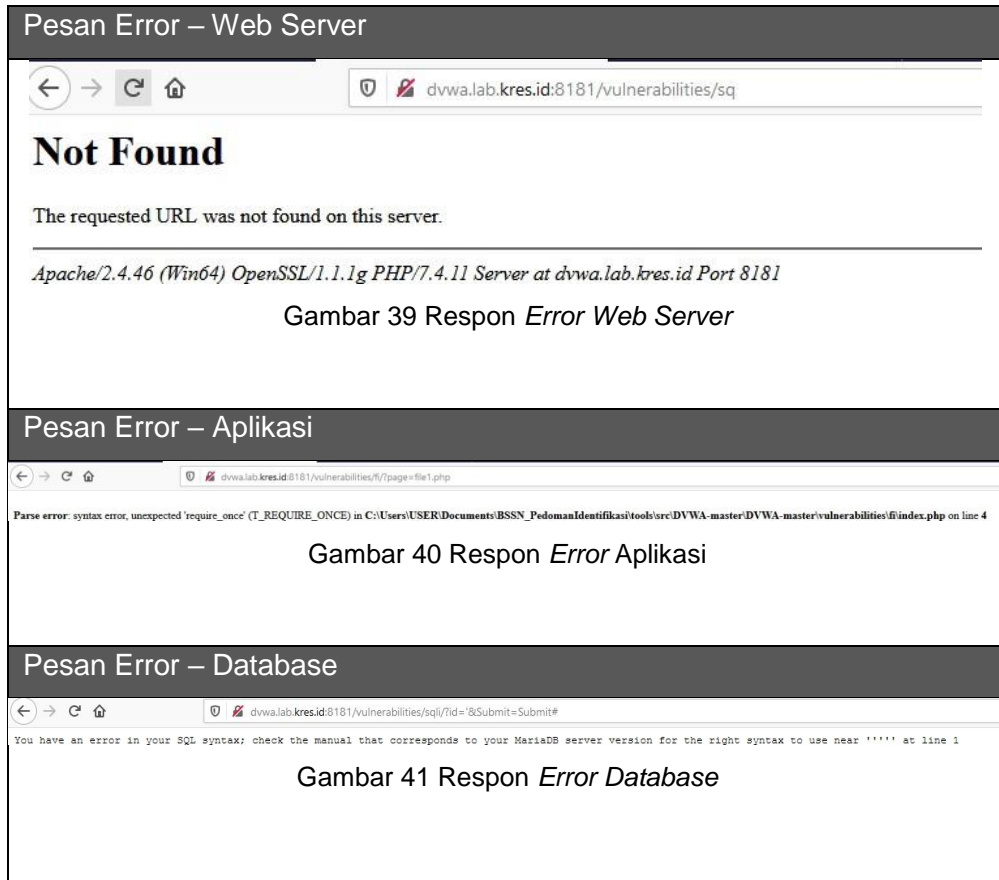
Kesalahan yang bersumber dari aplikasi dikembalikan oleh aplikasi itu sendiri melalui server web. Ini bisa jadi pesan kesalahan dari *framework* kode (PHP, JSP dll.) atau bisa jadi kesalahan spesifik yang dikembalikan dengan kode aplikasi. Kesalahan aplikasi mendetail biasanya tersedia informasi konfigurasi server, perpustakaan dan versi aplikasi yang digunakan.

- *Database Errors*

Kesalahan yang bersumber dari *database* adalah yang pesan kesalahan yang dikembalikan oleh Sistem *Database* ketika ada masalah dengan kueri atau koneksi. Setiap Sistem *database*, seperti



MySQL, Oracle atau MSSQL, memiliki serangkaian format atau pesan kesalahan sendiri. Kesalahan tersebut dapat memberikan informasi yang menarik seperti IP server *database*, tabel, kolom, dan detail *login*.



Setiap temuan pesan kesalahan (*error*) khususnya yang bersumber dari aplikasi dan *database* mempunyai risiko yang berbeda-beda sesuai dengan pesan kesalahan (*error*) yang ditampilkan.

### 3.10 *Cryptographic*

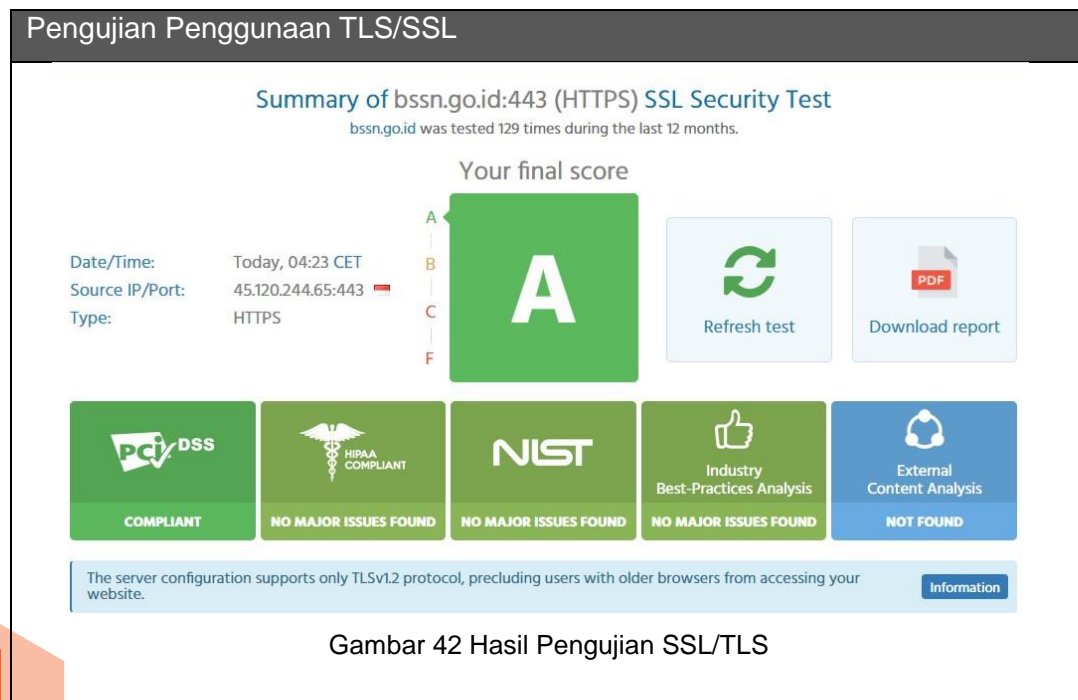
Penggunaan algoritma enkripsi yang salah dapat mengakibatkan eksposur terhadap data sensitif, kebocorankunci, autentikasi rusak, sesi tidak aman, dan serangan *spoofing*. Ada beberapa algoritma enkripsi atau *hash* yang diketahui lemah dan tidak disarankan untuk digunakan seperti MD5 dan RC4.

Selain pilihan yang tepat dari enkripsi yang aman atau algoritma *hash*, penggunaan parameter yang tepat juga penting untuk tingkat keamanan. Misalnya, mode ECB (*Electronic Code Book*) tidak disarankan untuk digunakan dalam enkripsi asimetris.

Alat pemindai kerentanan seperti Nessus<sup>[11]</sup>, NMAP (skrip), atau OpenVAS dapat memindai untuk penggunaan atau penerimaan enkripsi lemah terhadap protokol seperti SNMP, TLS, SSH, SMTP, dll.

#### 3.10.1. *Pengujian Cryptographic*

Pada pengujian *cryptographic* bisa menggunakan alat pemindai kerentanan maupun *website* yang dapat digunakan untuk pengujian sebagai contoh :immuniweb.com



Gambar 42 Hasil Pengujian SSL/TLS

### 3.11 ***Business Logic***

Aplikasi harus memastikan bahwa hanya data yang valid secara logis yang dapat dimasukkan di bagian depan serta langsung ke sisi server sebuah aplikasi sistem.

Kerentanan yang terkait dengan validasi data bisnis bersifat unik untuk masing-masing aplikasi dan memiliki kemungkinan kerentanan yang berbeda, untuk itu diperlukan Analisa dari pengujian uji penetrasi untuk bisa memahami alur dari proses bisnis pada aplikasi yang di uji. Pada tahapan pengujian pada *business logic* seorang pengujian harus menjalankan semua fungsi yang terlihat pada sisi klien dan mencoba melakukan intersepsi menggunakan Burp Suite atau OWASP ZAP terhadap semua *request* dari klien dan bahkan bisa mencoba manipulasi terhadap *response* yang diberikan oleh server.

## PENUTUP

### A. KESIMPULAN

Dokumen petunjuk teknis ITSA berbasis *web* merupakan dokumen pendukung yang digunakan untuk membantu personil maupun tim dalam melaksanakan kegiatan ITSA. Isinya meliputi penjelasan mengenai aplikasi berbasis *web*, teknik yang digunakan dalam melakukan ITSA, serta langkah-langkah pengujian yang dilakukan. Penjelasan mengenai langkah-langkah tersebut telah disertakan pula dengan dokumentasi untuk setiap pengujian. Sehingga hal ini dapat memudahkan pembaca dalam memahami petunjuk teknis yang dibuat.

### B. SARAN

Dokumen pendukung yang dibuat belum cukup dalam rangka upaya untuk membantu personil dalam melaksanakan kegiatan ITSA. Hal ini perlu diimbangi dengan meningkatkan kompetensi baik secara otodidak maupun pelatihan-pelatihan yang diikuti. Selain itu petunjuk teknis yang telah dibuat belum tentu relevan hingga tahun-tahun mendatang. Maka perlu adanya *update* dokumen menyesuaikan dengan *Website Security Checklist Guide* versi terbaru.

## LEMBAR PENGESAHAN

PETUNJUK TEKNIS  
PELAKSANAAN IIT SECURITY ASSESSMENT (ITSA)  
APLIKASI BERBASIS WEB

Telah disahkan oleh :

<p>Direktur Operasi Keamanan Siber</p> <div><p>Ditandatangani Secara Elektronik oleh : DIREKTUR OPERASI KEAMANAN SIBER</p><p>Ferdinand Mahulette, S.E. Brigadir Jenderal TNI</p></div> <p>Ferdinand Mahulette, S.E</p>	<p>Koordinator Kelompok</p> <p>Satryo Suryantoro, S.Sos</p>
---	---

KELOMPOK FUNGSI OPERASI IDENTIFIKASI DANN PROTEKSI  
DIREKTORAT OPERASI KEAMANAN SIBER  
BADAN SIBER DAN SANDI NEGARA



## REFERENSI

- [1] OWASP Testing Guide, <https://owasp.org/www-project-web-security-testing-guide>, 2009.
- [2] Hacking Exposed – Web Applications, Joel Scambray, 2002.
- [3] Mastering Modern Web Penetration Testing – Prakhar Prasad ,2016
- [4] Scooping Security Assessments – A Project Management Approach – Ahmed Abdel Aziz, 2021

### Lampiran I : Alat Pengujian ITSA Aplikasi Berbasis Web

Berikut adalah penggunaan alat bantu yang digunakan dalam panduan ITSA untuk aplikasi berbasis WEB.

- [1] <https://developers.google.com/web/tools/chrome-devtools>
- [2] <https://docs.microsoft.com/en-us/microsoft-edge/devtools-guide>
- [3] <https://developer.mozilla.org/en-US/docs/Tools>
- [4] Nmap - <https://nmap.org/>
- [5] WhatWeb - <https://github.com/urbanadventurer/WhatWeb>
- [6] OpenVas - <https://www.openvas.org/>
- [7] OWASP Zend Attack Proxy (ZAP) - <https://www.zaproxy.org/>
- [8] Daftar Kata Sandi lemah – <https://vyra.bssn.go.id/tools/password.lst>
- [9] Burp Suite - <https://portswigger.net/burp>
- [10] SQLMap - <http://sqlmap.org/>
- [11] Nessus - <https://www.tenable.com/products/nessus>

## Lampiran II : Pendeteksian dan Menghindari WAF

*Web Application Firewall* (WAF) adalah sebuah sistem *firewall* untuk level aplikasi HTTP. *Web Application Firewall* (WAF) bekerja dengan melakukan deteksi dan pencegahan terhadap serang-seranganyang ada pada sisi aplikasi berbasis web. *Web Application Firewall* (WAF) ini menerapkan seperangkat aturan untuk percakapan pada protokol HTTP. Umumnya, aturan ini mencakup serangan umum seperti *Cross-site Scripting* (XSS) dan *SQL Injection*.

*Web Application Firewall* (WAF) dapat berbentuk alat (*hardware*), *plugin* pada server dan juga layanan dari penyedia jasa, WAF dapat disesuaikan dengan aplikasi.

Beberapa contoh *Web Application Firewall* (WAF) dalam bentuk alat (*hardware*) adalah :

- F5 *Web Application Firewall* (WAF)
- Barracuda *Web Application Firewall* (WAF)
- Fortinet *Web Application Firewall* (WAF)

Beberapa contoh *Web Application Firewall* (WAF) dalam bentuk *plugin* pada server adalah :

- ModSecurity (<https://www.modsecurity.org>)
- NAXSI (<https://github.com/nbs-system/naxsi>)
- WebKnight (<https://www.aqtronix.com/?PageID=99>)
- Shadow Daemon (<https://shadowd.zecure.org/overview/introduction/>)

Sedangkan beberapa contoh penyedia jasa layanan *Web Application Firewall* (WAF) adalah :

- Incapsula
- Cloudflare
- SUCURI

Beberapa produk *Web Application Firewall* (WAF) mengidentifikasi diri mereka sendiri di dalambagian respon.

Gambar 43 Hasil Identifikasi Jenis WAF

Jika sebuah aplikasi berbasis web menggunakan *Web Application Firewall* (WAF) dari penyedia jasa , *WebApplication Firewall* (WAF) akan menyembunyikan alamat asli dari aplikasi web di belakangnya. Maka itu dibutuhkan teknik untuk bisa mendapatkan alamat asli dari aplikasi web yang akan dilakukan uji penetrasi.

### Pencarian Alamat Asli Web - Cloudflare

Untuk mendapatkan alamat asli dari sebuah website yang dilindungi oleh WAF bisa menggunakan [crimeflare.org](https://crimeflare.org)



The screenshot shows a web browser window with the address bar displaying "Not secure | crimeflare.org:82/cgi-bin/cfsearch.cgi". The main content area has a yellow background and displays the following information:

- Domain: [elle.ns.cloudflare.com](#) and [seth.ns.cloudflare.com](#)
- SSL certificate info for [bnh\[REDACTED\]](#) nameserver search: [elle seth](#)
- A direct-connect IP address was found: [bnh\[REDACTED\]](#) 162.209.1.100 UNITED STATES
- An attempt to fetch a page from this IP was unsuccessful.
- Previous lookups for this domain:
  - 2020-10-26: [bnh\[REDACTED\]](#) 162.209.1.100 UNITED STATES
- A note at the bottom states: "The direct-connect IP addresses listed above may point to a server that hosts more than one domain. Click below to see other domains from our database with the same direct-connect IP. Th" followed by a link to [162.209.1.100](#).
- A link at the bottom right says "Back to search box".

Gambar 44 Hasil Pencarian Alamat Asli

**Lampiran III : Websites Security Checklist Guide**

<b>Information Gathering</b>	<b>Nama Pengujian</b>	<b>Tujuan</b>	<b>Status (*)</b>	<b>Tools</b>
WSTG-INFO-01	<i>Conduct Search Engine Discovery Reconnaissance for Information Leakage</i>	- Mengidentifikasi informasi konfigurasi dan desain sensitif aplikasi, sistem, maupun organisasi yang terdampak baik melalui situs web dan layanan pihak ketiga		<ul style="list-style-type: none"> <li>- search engine</li> <li>- common crawl</li> <li>- Internet Archive</li> <li>- Wayback Machine</li> <li>- Startpage</li> <li>- Shodan</li> </ul>
WSTG-INFO-02	<i>Fingerprint Web Server</i>	-Menentukan versi dan jenis server web yang digunakan untuk mengetahui celah kerentanan lanjutan		<ul style="list-style-type: none"> <li>- Netcraft</li> <li>- Nikto</li> <li>- Nmap</li> </ul>
WSTG-INFO-03	<i>Review Webserver Metafiles for Information Leakage</i>	<ul style="list-style-type: none"> <li>-Mengidentifikasi <i>path</i> tersembunyi melalui analisis <i>file metadata</i></li> <li>-Mengekstrak dan memetakan informasi lain</li> </ul>		<ul style="list-style-type: none"> <li>- Browser (view source atau dev tools functionality)</li> <li>- Curl</li> <li>- Wget</li> <li>- Burp Suite</li> <li>- ZAP</li> </ul>
WSTG-INFO-04	<i>Enumerate Applications on Webserver</i>	-Mengenumerasi aplikasi yang ada di server web		<ul style="list-style-type: none"> <li>- DNS lookup tools (nslookup, dig, dll)</li> <li>- search engines (google, bing, dll)</li> <li>- DNS khusus yang berhubungan dengan layanan pencarian web based (see text)</li> <li>- Nmap</li> <li>- Nessus</li> <li>- nikto</li> </ul>

DRAFT PANDUAN TEKNIS  
PELAKSANAAN IT SECURITY ASSESSMENT (ITSA)



Information Gathering	Nama Pengujian	Tujuan	Status (*)	Tools
WSTG-INFO-05	<i>Review Webpage Content for Information Leakage</i>	<ul style="list-style-type: none"> <li>- Meninjau <i>metadata</i> halaman web untuk menemukan berbagai kebocoran informasi</li> <li>- Mengumpulkan <i>file</i> JavaScript dan meninjau kode JS untuk memahami aplikasi dan menemukan kebocoran informasi</li> <li>- Mengidentifikasi apakah ada <i>file source map</i> atau <i>file debug front-end</i> lainnya</li> </ul>		<ul style="list-style-type: none"> <li>- wget</li> <li>- browser "view source" function</li> <li>- eyeballs</li> <li>- curl</li> <li>- burp suite</li> <li>- waybackurls</li> <li>- google maps API scanner</li> </ul>
WSTG-INFO-06	<i>Identify application entry points</i>	<ul style="list-style-type: none"> <li>- Mengidentifikasi kemungkinan titik masuk dan injeksi melalui analisis permintaan dan respons</li> </ul>		<ul style="list-style-type: none"> <li>- OWASP ZAP</li> <li>- Burp suite</li> <li>- fiddler</li> </ul>
WSTG-INFO-07	<i>Map execution paths through application</i>	<ul style="list-style-type: none"> <li>- Memetakan aplikasi target dan memahami alur kerja</li> </ul>		<ul style="list-style-type: none"> <li>- ZAP</li> <li>- list of spreadsheet software</li> <li>- diagramming software</li> </ul>
WSTG-INFO-08	<i>Fingerprint Web Application Framework</i>	<ul style="list-style-type: none"> <li>- Mengetahui komponen-komponen yang digunakan oleh aplikasi web</li> </ul>		<ul style="list-style-type: none"> <li>- whatweb</li> <li>- wappalyzer</li> </ul>
WSTG-INFO-09	<i>Fingerprint Web Application</i>	<ul style="list-style-type: none"> <li>- Mengetahui komponen-komponen yang digunakan oleh aplikasi web</li> </ul>		<ul style="list-style-type: none"> <li>- whatweb</li> <li>- wappalyzer</li> </ul>
WSTG-INFO-10	<i>Map Application Architecture</i>	<ul style="list-style-type: none"> <li>- <i>Generate</i> peta aplikasi yang ada berdasarkan penelitian yang dilakukan</li> </ul>		

DRAFT PANDUAN TEKNIS  
PELAKSANAAN IT SECURITY ASSESSMENT (ITSA)



Configuration and Deploy Management Testing	Nama Pengujian	Tujuan	Status (*)	Tools
WSTG-CONF-01	<i>Test Network Infrastructure Configuration</i>	<ul style="list-style-type: none"> <li>- Meninjau konfigurasi aplikasi yang diatur di seluruh jaringan</li> <li>- Melakukan validasi <i>framework</i> dan sistem yang digunakan aman dan tidak rentan terhadap kerentanan yang disebabkan karena <i>software</i> tidak dirawat atau pengaturan dan kredensial <i>default</i></li> </ul>		
WSTG-CONF-02	<i>Test Application Platform Configuration</i>	<ul style="list-style-type: none"> <li>- Memastikan <i>file default</i> dan <i>file</i> yang diketahui telah dihapus- Memvalidasi bahwa tidak ada kode debug atau ekstensi di <i>environments</i></li> </ul>		<ul style="list-style-type: none"> <li>- CIS-CAT Lite-</li> <li>- Microsoft's Attack Surface Analyzer</li> <li>- NIST's National checklist Program</li> </ul>
WSTG-CONF-03	<i>Test File Extensions Handling for Sensitive Information</i>	<ul style="list-style-type: none"> <li>- Melakukan ekstensi <i>file</i> sensitif atau ekstensi yang mungkin berisi data mentah (contoh : <i>scripts</i>, <i>raw data</i>, kredensial, dll)</li> <li>- Melakukan validasi bahwa tidak ada <i>framework bypass</i> pada aturan yang ditetapkan</li> </ul>		<ul style="list-style-type: none"> <li>- Wget</li> <li>- Curl</li> <li>- google for "web mirroring tools"</li> </ul>
WSTG-CONF-04	<i>Review Old Backup and Unreferenced Files for Sensitive Information</i>	<ul style="list-style-type: none"> <li>- menemukan dan menganalisis <i>file</i> yang tidak direferensikan yang mungkin berisi mengenai informasi sensitif</li> </ul>		<ul style="list-style-type: none"> <li>- nessus</li> <li>- nikto2</li> </ul>
WSTG-CONF-05	<i>Enumerate Infrastructure and Application Admin Interfaces</i>	<ul style="list-style-type: none"> <li>- mengidentifikasi <i>interfaces</i> dan fungsionalitas administrator yang tersembunyi</li> </ul>		<ul style="list-style-type: none"> <li>- OWASP ZAP- Forced Browse</li> <li>- THC-HYDRA</li> <li>- netsparker</li> </ul>



DRAFT PANDUAN TEKNIS  
PELAKSANAAN IT SECURITY ASSESSMENT (ITSA)



Configuration and Deploy Management Testing	Nama Pengujian	Tujuan	Status (*)	Tools
WSTG-CONF-06	<i>Test HTTP Methods</i>	<ul style="list-style-type: none"> <li>- Menghitung metode HTTP yang didukung</li> <li>- Menguji <i>access control bypass</i></li> <li>- Menguji kerentanan XST</li> <li>- Menguji teknik <i>override</i> metode HTTP</li> </ul>		<ul style="list-style-type: none"> <li>- Ncat</li> <li>- Curl</li> <li>- nmap http-methods NSE script</li> <li>- w3af plugin htaccess_methods</li> </ul>
WSTG-CONF-07	<i>Test HTTP Strict Transport Security</i>	- Menguji <i>header</i> dan validitas HSTS		
WSTG-CONF-08	<i>Test RIA cross domain policy</i>	- Meninjau dan memvalidasi <i>policy file</i>		<ul style="list-style-type: none"> <li>- nikto</li> <li>- OWASP ZAP</li> <li>- W3af</li> </ul>
WSTG-CONF-09	<i>Test File Permission</i>	- Meninjau dan mengidentifikasi izin <i>file rogue</i> apapun		<ul style="list-style-type: none"> <li>- Windows AccessEnum</li> <li>- Windows AccessChk</li> <li>- Linux namei</li> </ul>
WSTG-CONF-10	<i>Test for Subdomain Takeover</i>	<ul style="list-style-type: none"> <li>- Menghitung semua domain yang mungkin</li> <li>- Mengidentifikasi domain yang terlupakan ataupun salah konfigurasi</li> </ul>		<ul style="list-style-type: none"> <li>- dig-man page</li> <li>- recon-ng-Web Reconnaissance Framework</li> <li>- theHarvester - OSINT subdomain enumeration tool</li> <li>- dnsrecon - DNS Enumeration Script</li> <li>- OWASP Amass DNS enumeration</li> </ul>

DRAFT PANDUAN TEKNIS  
PELAKSANAAN IT SECURITY ASSESSMENT (ITSA)



<b>Configuration and Deploy Management Testing</b>	<b>Nama Pengujian</b>	<b>Tujuan</b>	<b>Status (*)</b>	<b>Tools</b>
WSTG-CONF-11	<i>Test Cloud Storage</i>	- Memastikan konfigurasi kontrol akses untuk penyimpanan sudah terpasang dengan benar		- AWS CLI

<b>Identity Management Testing</b>	<b>Nama Pengujian</b>	<b>Tujuan</b>	<b>Status (*)</b>	<b>Tools</b>
WSTG-IDNT-01	<i>Test Role Definitions</i>	- Mengidentifikasi dan mendokumentasikan <i>roles</i> yang digunakan oleh aplikasi - Mencoba untuk beralih, mengubah, atau mengakses <i>role</i> lain - Meninjau perincian roles dan kebutuhan		- Burp's Authorize Extension - ZAP's Access Control Testing add-on
WSTG-IDNT-02	<i>Test User Registration Process</i>	- Memverifikasi antara persyaratan identitas untuk pendaftaran selaras dengan persyaratan bisnis dan keamanan - Memvalidasi proses pendaftaran		- HTTP Proxy
WSTG-IDNT-03	<i>Test Account Provisioning Process</i>	- Memverifikasi akun yang dapat menyediakan akun lain dan jenisnya		- HTTP Proxy
WSTG-IDNT-04	<i>Testing for Account Enumeration and Guessable User Account</i>	- Meninjau proses yang berkaitan dengan identifikasi pengguna (contoh : pendaftaran, <i>login</i> , dll) - Menghitung pengguna jika memungkinkan melalui analisis <i>respon</i>		- OWASP ZAP - curl - PERL
WSTG-IDNT-05	<i>Testing for Weak or unenforced username policy</i>	- Menentukan apakah struktur nama akun yang konsisten membuat aplikasi rentan terhadap penghitungan akun atau tidak - Menentukan apakah pesan <i>application's error</i> mengizinkan penghitungan akun atau tidak		

DRAFT PANDUAN TEKNIS  
PELAKSANAAN IT SECURITY ASSESSMENT (ITSA)



Authentication Testing	Nama Pengujian	Tujuan	Status (*)	Tools
WSTG-ATHN-01	<i>Testing for Credentials Transported over an Encrypted Channel</i>	- Menilai apakah ada kasus penggunaan situs web atau aplikasi yang menyebabkan server atau klien bertukar kredensi tanpa enkripsi		
WSTG-ATHN-02	<i>Testing for Default Credentials</i>	- Menghitung aplikasi untuk kredensial <i>default</i> dan memvalidasinya - Meninjau dan menilai akun pengguna baru dan apakah akun tersebut dibuat dengan <i>default</i> atau pola yang dapat diidentifikasi		- Burp Intruder - THC Hydra - Nikto 2
WSTG-ATHN-03	<i>Testing for Weak Lock Out Mechanism</i>	- Mengevaluasi kemampuan mekanisme penguncian akun untuk memitigasi <i>brute force password</i> - Mengevaluasi resistensi mekanisme buka kunci terhadap pembukaan kunci akun yang tidak sah		
WSTG-ATHN-04	<i>Testing for Bypassing Authentication Schema</i>	- Memastikan bahwa autentikasi diterapkan di seluruh layanan yang memerlukannya		- WebGoat - OWASP ZAP
WSTG-ATHN-05	<i>Testing for Vulnerable Remember Password</i>	- Memvalidasi bahwa sesi yang dihasilkan dikelola dengan aman dan tidak membahayakan kredensial pengguna		
WSTG-ATHN-06	<i>Testing for Browser Cache Weaknesses</i>	- Meninjau apakah aplikasi menyimpan informasi sensitif di sisi klien - Meninjau akses dapat terjadi tanpa otorisasi		- OWASP ZAP
WSTG-ATHN-07	<i>Testing for Weak Password Policy</i>	- Menentukan ketahanan aplikasi terhadap tebakan kata sandi <i>brute force</i> menggunakan <i>password dictionaries</i> yang tersedia dengan mengevaluasi persyaratan <i>length</i> , <i>complexity</i> , <i>reuse</i> , dan <i>aging requirements</i> dari <i>password</i>		
WSTG-ATHN-08	<i>Testing for Weak Security Question Answer</i>	- Menentukan <i>complexity</i> - Menilai kemungkinan jawaban pengguna dan kemampuan <i>brute force</i>		

DRAFT PANDUAN TEKNIS  
PELAKSANAAN IT SECURITY ASSESSMENT (ITSA)



<b>Authentication Testing</b>	<b>Nama Pengujian</b>	<b>Tujuan</b>	<b>Status (*)</b>	<b>Tools</b>
WSTG-ATHN-09	<i>Testing for Weak Password Change or Reset Functionalities</i>	<ul style="list-style-type: none"> <li>- Menentukan resistensi aplikasi terhadap subversi dari proses perubahan akun yang memungkinkan seseorang untuk mengubah kata sandi akun</li> <li>- Menentukan ketahanan fungsi reset kata sandi terhadap <i>guessing</i> atau <i>bypassing</i></li> </ul>		
WSTG-ATHN-10	<i>Testing for Weaker Authentication in Alternative Channel</i>	<ul style="list-style-type: none"> <li>- Mengidentifikasi saluran autentikasi alternatif</li> <li>- Menilai langkah-langkah keamanan yang digunakan</li> </ul>		

<b>Authorization Testing</b>	<b>Nama Pengujian</b>	<b>Tujuan</b>	<b>Status (*)</b>	<b>Tools</b>
WSTG-ATHZ-01	<i>Testing Directory Traversal File Include</i>	<ul style="list-style-type: none"> <li>- Mengidentifikasi titik injeksi yang berkaitan dengan <i>path traversal</i></li> <li>- Menilai teknik <i>bypassing</i> dan mengidentifikasi sejauh mana <i>path traversal</i></li> </ul>		<ul style="list-style-type: none"> <li>- DotDotPwn-The Directory Traversal Fuzzer</li> <li>- Path Traversal Fuzz String (from Wfuzz Tool)</li> <li>- OWASP ZAP</li> <li>- Burp Suite</li> <li>- Encoding/Decoding Tools</li> <li>- String searcher "grep"</li> <li>- DirBuster</li> </ul>

DRAFT PANDUAN TEKNIS  
PELAKSANAAN IT SECURITY ASSESSMENT (ITSA)



<b>Authorization Testing</b>	<b>Nama Pengujian</b>	<b>Tujuan</b>	<b>Status (*)</b>	<b>Tools</b>
WSTG-ATHZ-02	<i>Testing for Bypassing Authorization Schema</i>	- Mengkaji apakah memungkinkan untuk akses horizontal atau vertikal		- OWASP ZAP (ZAP add-on: Access Control Testing) - Port Swigger Burp Suite (Burp extension : AuthMatrix & Burp extension : Authorize)
WSTG-ATHZ-03	<i>Testing for Privilege Escalation</i>	- Mengidentifikasi titik injeksi yang terkait dengan manipulasi <i>privilege</i> - Fuzz / mencoba untuk melewati langkah-langkah keamanan		- OWASP ZAP
WSTG-ATHZ-04	<i>Testing for Insecure Direct Object References</i>	- Mengidentifikasi titik-titik dimana referensi objek dapat terjadi - Menilai langkah-langkah kontrol akses dan kerentanan terhadap IDOR		

<b>Session Management Testing</b>	<b>Nama Pengujian</b>	<b>Tujuan</b>	<b>Status (*)</b>	<b>Tools</b>
WSTG-SESS-01	<i>Testing for Session Management Schema</i>	- Mengumpulkan token sesi untuk pengguna yang sama dan berbeda jika memungkinkan - Menganalisis dan memastikan bahwa ada cukup keacakan untuk menghentikan serangan <i>session forging</i> - Memodifikasi <i>cookie</i> yang tidak ditandatangani dan berisi informasi yang dapat dimanipulasi		- OWASP ZAP - Burp Sequencer - YEHG's Jhijack

DRAFT PANDUAN TEKNIS  
PELAKSANAAN IT SECURITY ASSESSMENT (ITSA)



Session Management Testing	Nama Pengujian	Tujuan	Status (*)	Tools
WSTG-SESS-02	<i>Testing for Cookies Attributes</i>	- Memastikan konfigurasi keamanan yang tepat diatur untuk <i>cookie</i>		- Intercepting Proxy (OWASP ZAP, Web Proxy Burp Suite) - Browser Plug-in (Tamper Data for FF Quantum, "FireSheep" for FireFox, "EditThisCookie" for chrome, dan "Cookiebro-cookie manager" for FireFox)
WSTG-SESS-03	<i>Testing for Session Fixation</i>	- Menganalisis mekanisme autentikasi dan alurnya - Memodifikasi <i>cookie</i> dan nilai dampaknya		- OWASP ZAP
WSTG-SESS-04	<i>Testing for Exposed Session Variables</i>	- Memastikan enkripsi yang tepat diterapkan - Meninjau konfigurasi <i>caching</i> - Menilai keamanan saluran dan metode		
WSTG-SESS-05	<i>Testing for Cross Site Request Forgery</i>	- Menentukan apakah mungkin untuk memulai permintaan atas nama pengguna yang tidak diprakarsai oleh pengguna		- OWASP ZAP - CSRF Tester - Pinata-csrf-tool
WSTG-SESS-06	<i>Testing for Logout Functionality</i>	- Menilai UI <i>logout</i> - Menganalisis batas waktu sesi dan apakah sesi dihentikan dengan benar setelah <i>logout</i>		- Burp Suite-Repeater
WSTG-SESS-07	<i>Testing Session Timeout</i>	- Memvalidasi bahwa ada batas waktu sesi yang sulit		
WSTG-SESS-08	<i>Testing for Session Puzzling</i>	- Mengidentifikasi semua variabel sesi - Merusak <i>logical flow</i> dari pembuatan sesi		
WSTG-SESS-09	<i>Testing for Session Hijacking</i>	- Mengidentifikasi <i>cookie</i> sesi yang rentan - Membajak <i>cookie</i> yang rentan dan menilai tingkat risikonya		- OWASP ZAP - Jhijack

DRAFT PANDUAN TEKNIS  
PELAKSANAAN IT SECURITY ASSESSMENT (ITSA)



<b>Data Validation Testing</b>	<b>Nama Pengujian</b>	<b>Tujuan</b>	<b>Status (*)</b>	<b>Tools</b>
WSTG-INPV-01	<i>Testing for Reflected Cross Site Scripting</i>	<ul style="list-style-type: none"> <li>- Mengidentifikasi variabel</li> <li>- Menilai <i>input</i> yang diterima dan pengkodean yang diterapkan</li> </ul>		<ul style="list-style-type: none"> <li>- PHP Charset Encoder (PCE)</li> <li>- Hackvertor</li> <li>- XSS-Proxy</li> <li>- ratproxy</li> <li>- Burp Proxy</li> <li>- OWASP ZAP</li> </ul>
WSTG-INPV-02	<i>Testing for Stored Cross Site Scripting</i>	<ul style="list-style-type: none"> <li>- Mengidentifikasi masukan tersimpan yang tercermin di sisi klien</li> <li>- Menilai <i>input</i> telah diterima dan menerapkan <i>encoding</i> saat <i>return</i></li> </ul>		<ul style="list-style-type: none"> <li>- PHP Charset Encoder (PCE)</li> <li>- Hackvertor</li> <li>- BeEF</li> <li>- XSS-Proxy</li> <li>- Burp Proxy</li> <li>- XSS Assistant</li> <li>- OWASP ZAP</li> <li>- XSS Hunter Portable</li> </ul>
WSTG-INPV-03	<i>Testing for HTTP Verb Tampering</i>	<ul style="list-style-type: none"> <li>- Menghitung metode HTTP yang didukung</li> <li>- Menguji <i>access control bypass</i></li> <li>- Menguji kerentanan XST</li> <li>- Menguji teknik <i>override</i> metode HTTP</li> </ul>		<ul style="list-style-type: none"> <li>- Ncat</li> <li>- Curl</li> <li>- nmap http-methods</li> <li>- NSE script</li> <li>- w3af plugin htaccess_methods</li> </ul>
WSTG-INPV-04	<i>Testing for HTTP Parameter Pollution</i>	<ul style="list-style-type: none"> <li>- Mengidentifikasi <i>backend</i> dan metode <i>parsing</i> yang digunakan</li> <li>- Menilai titik injeksi dan melakukan percobaan <i>bypassing filter input</i> menggunakan HPP</li> </ul>		<ul style="list-style-type: none"> <li>- OWASP ZAP</li> </ul>

DRAFT PANDUAN TEKNIS  
PELAKSANAAN IT SECURITY ASSESSMENT (ITSA)



Data Validation Testing	Nama Pengujian	Tujuan	Status (*)	Tools
WSTG-INPV-05	<i>Testing for SQL Injection</i>	<ul style="list-style-type: none"> <li>- Mengidentifikasi SQL</li> <li>- Mengkaji <i>severity</i> SQL dan tingkat akses yang bisa dicapai</li> </ul>		<ul style="list-style-type: none"> <li>- SQL injection fuzz string (from wfuzz tool)-fuzzdb</li> <li>- sqlbftools</li> <li>- Bernardo Damele A. G.: sqlmap, automatic SQL injection tool</li> <li>- Muhaimin Dzulfakar: MySqliot, MySql Injection takeover tool</li> </ul>
WSTG-INPV-06	<i>Testing for LDAP Injection</i>	<ul style="list-style-type: none"> <li>- Mengidentifikasi LDAP</li> <li>- Mengkaji <i>severity</i> LDAP</li> </ul>		<ul style="list-style-type: none"> <li>- Softerra LDAP Browser</li> </ul>
WSTG-INPV-07	<i>Testing for XML Injection</i>	<ul style="list-style-type: none"> <li>- Mengidentifikasi XML</li> <li>- Menilai jenis eksploitasi yang dapat dilakukan dan <i>severity</i>-nya</li> </ul>		<ul style="list-style-type: none"> <li>- XML injection fuzz string (from wfuzz tool)</li> </ul>
WSTG-INPV-08	<i>Testing for SSI Injection</i>	<ul style="list-style-type: none"> <li>- Mengidentifikasi SSI</li> <li>- Mengkaji <i>severity</i> SSI</li> </ul>		<ul style="list-style-type: none"> <li>- Web Proxy Burp Suite</li> <li>- OWASP ZAP</li> <li>- String searcher: grep</li> </ul>
WSTG-INPV-09	<i>Testing for XPath Injection</i>	<ul style="list-style-type: none"> <li>- Mengidentifikasi XPATH</li> </ul>		
WSTG-INPV-10	<i>Testing for IMAP SMTP Injection</i>	<ul style="list-style-type: none"> <li>- Mengidentifikasi IMAP/SMTP</li> <li>- Memahami data <i>flow</i> dan struktur <i>deployment</i> sistem</li> </ul>		
WSTG-INPV-11	<i>Testing for Code Injection</i>	<ul style="list-style-type: none"> <li>- Mengidentifikasi injeksi ketika mencoba memasukkan kode ke dalam aplikasi</li> <li>- Mengkaji <i>severity</i> injeksi tersebut</li> </ul>		



DRAFT PANDUAN TEKNIS  
PELAKSANAAN IT SECURITY ASSESSMENT (ITSA)



Data Validation Testing	Nama Pengujian	Tujuan	Status (*)	Tools
WSTG-INPV-12	<i>Testing for Command Injection</i>	- Mengidentifikasi dan menilai injeksi <i>command</i>		- OWASP WebGoat - Commix
WSTG-INPV-13	<i>Testing for Format String Injection</i>	- Menilai apakah menginjeksi konversi format <i>string</i> dapat menyebabkan perilaku abnormal yang tidak diinginkan dari aplikasi atau tidak		
WSTG-INPV-14	<i>Testing for Incubated Vulnerability</i>	- Mengidentifikasi injeksi yang disimpan - Memahami bagaimana <i>recall</i> bisa terjadi - Mengatur <i>listener</i> atau mengaktifkan <i>recall</i> jika memungkinkan		- XSS-Proxy - OWASP ZAP - Burp Suite - Metasploit
WSTG-INPV-15	<i>Testing for HTTP Splitting Smuggling</i>	- Menilai apakah aplikasi rentan terhadap <i>splitting</i> , dan mengidentifikasi kemungkinan serangan yang dapat terjadi - Menilai apakah rantai komunikasi rentan terhadap <i>smuggling</i> , dan mengidentifikasi kemungkinan serangan yang dapat terjadi		
WSTG-INPV-16	<i>Testing for HTTP Incoming Requests</i>	- Memantau semua permintaan HTTP yang masuk dan keluar ke server web untuk memeriksa setiap permintaan yang mencurigakan - Memantau <i>traffic</i> HTTP tanpa perubahan <i>proxy</i> browser pengguna akhir atau aplikasi sisi klien		- Fiddler - TCPProxy - Charles Web Debugging Proxy - WireShark - PowerEdit-Pcap - pcpatteller - replayproxy - Ostinato
WSTG-INPV-17	<i>Testing for Host Header Injection</i>	- Menilai apakah <i>header-host</i> sedang melakukan <i>parsing</i> secara dinamis dalam aplikasi - Melewati kontrol keamanan yang mengandalkan <i>header</i>		

DRAFT PANDUAN TEKNIS  
PELAKSANAAN IT SECURITY ASSESSMENT (ITSA)



<b>Data Validation Testing</b>	<b>Nama Pengujian</b>	<b>Tujuan</b>	<b>Status (*)</b>	<b>Tools</b>
WSTG-INPV-18	<i>Testing for Server-side Template Injection</i>	<ul style="list-style-type: none"> <li>- Mendeteksi titik kerentanan injeksi <i>template</i></li> <li>- Mengidentifikasi mesin <i>templating</i></li> <li>- Membangun eksploitasi</li> </ul>		<ul style="list-style-type: none"> <li>- Tplmap</li> <li>- Backslash Powered Scanner Burp Suite extension</li> <li>- Template expression test strings/payloads list</li> </ul>
WSTG-INPV-19	<i>Testing for Server-Side Request Forgery</i>	<ul style="list-style-type: none"> <li>- Mengidentifikasi SSRF</li> <li>- Menguji apakah titik injeksi dapat dieksploitasi</li> <li>- Menilai tingkat <i>severity</i> kerentanan</li> </ul>		

<b>Error Handling</b>	<b>Nama Pengujian</b>	<b>Tujuan</b>	<b>Status (*)</b>	<b>Tools</b>
WSTG-ERRH-01	<i>Testing for Improper Error Handling</i>	<ul style="list-style-type: none"> <li>- Mengidentifikasi <i>error</i> pada <i>output existing</i></li> <li>- Menganalisis <i>output</i> berbeda yang kembali</li> </ul>		
WSTG-ERRH-02	<i>Testing for Stack Traces</i>			

<b>Cryptography</b>	<b>Nama Pengujian</b>	<b>Tujuan</b>	<b>Status (*)</b>	<b>Notes</b>
WSTG-CRYP-01	<i>Testing for Weak Transport Layer Security</i>	<ul style="list-style-type: none"> <li>- Memvalidasi konfigurasi layanan</li> <li>- Me-review dan memvalidasi kekuatan kriptografi sertifikat digital</li> <li>- Memastikan bahwa keamanan TLS tidak <i>bypass</i> dan diimplementasikan di seluruh aplikasi</li> </ul>		

DRAFT PANDUAN TEKNIS  
PELAKSANAAN IT SECURITY ASSESSMENT (ITSA)



<b>Cryptography</b>	Nama Pengujian	Tujuan	Status (*)	Notes
WSTG-CRYP-02	<i>Testing for Padding Oracle</i>	<ul style="list-style-type: none"> <li>- Mengidentifikasi pesan terenkripsi yang bergantung pada <i>padding</i></li> <li>- Mencoba untuk melakukan <i>break padding</i> pada pesan terenkripsi dan menganalisis kembali pesan <i>error</i></li> </ul>		<ul style="list-style-type: none"> <li>- Bletchley</li> <li>- PadBuster</li> <li>- Padding Oracle Exploitation Tool (POET)</li> <li>- Poracle</li> <li>- python-paddingoracle</li> </ul>
WSTG-CRYP-03	<i>Testing for Sensitive Information Sent via Unencrypted Channels</i>	<ul style="list-style-type: none"> <li>- Mengidentifikasi informasi sensitif yang ditransmisikan melalui beragam <i>channel</i></li> <li>- Menilai <i>privacy</i> dan keamanan pada <i>channel</i> yang digunakan</li> </ul>		<ul style="list-style-type: none"> <li>- curl</li> <li>- grep</li> <li>- Wireshark</li> <li>- TCPDUMP</li> </ul>
WSTG-CRYP-04	<i>Testing for Weak Encryption</i>	<ul style="list-style-type: none"> <li>- Memberikan petunjuk untuk mengidentifikasi kelemahan enkripsi atau <i>hashing</i> yang digunakan dan diimplementasikan</li> </ul>		<ul style="list-style-type: none"> <li>- Vulnerability Scanner (Nessus, NMAP, OpenVas)</li> <li>- Static code analysis tool (klocwork, fortify, coverity, checkmark)</li> </ul>

<b>Business logic Testing</b>	Nama Pengujian	Tujuan	Status (*)	Notes
WSTG-BUSL-01	<i>Test Business Logic Data Validation</i>	<ul style="list-style-type: none"> <li>- Mengidentifikasi data injeksi</li> <li>- Memvalidasi semua pemeriksaan yang terjadi di <i>back end</i> dan tidak bisa dilewati</li> <li>- Mencoba untuk melakukan <i>break</i> format data yang diharapkan dan menganalisis bagaimana aplikasi dapat ditangani</li> </ul>		<ul style="list-style-type: none"> <li>- OWASP ZAP</li> <li>- Burp Suite</li> </ul>

DRAFT PANDUAN TEKNIS  
PELAKSANAAN IT SECURITY ASSESSMENT (ITSA)



<b>Business logic Testing</b>	<b>Nama Pengujian</b>	<b>Tujuan</b>	<b>Status (*)</b>	<b>Notes</b>
WSTG-BUSL-02	<i>Test Ability to Forge Requests</i>	<ul style="list-style-type: none"> <li>- Me-review dokumentasi <i>project</i> untuk mencari perkiraan, prediksi, atau fungsi tersembunyi dari <i>fields</i></li> <li>- Memasukkan data <i>logic</i> valid di untuk melakukan <i>bypass</i> terhadap alur kerja bisnis <i>logic</i> yang normal.</li> </ul>		<ul style="list-style-type: none"> <li>- OWASP ZAP</li> <li>- Burp Suite</li> </ul>
WSTG-BUSL-03	<i>Test Integrity Checks</i>	<ul style="list-style-type: none"> <li>- Me-review dokumentasi <i>project</i> terkait komponen sistem yang pindah, tersimpan, dan data tertangan</li> <li>- Menentukan jenis data apa yang diterima secara logis oleh komponen dan jenis sistem yang harus diwaspadai</li> <li>- Menentukan siapa yang harus diizinkan untuk memodifikasi atau membaca data pada tiap komponen</li> <li>- Mencoba untuk menyisipkan, memperbarui, atau menghapus nilai data yang digunakan oleh setiap komponen yang seharusnya tidak diperbolehkan</li> </ul>		<ul style="list-style-type: none"> <li>- Various system/application tools (editors, file manipulation tools)</li> <li>- OWASP ZAP</li> <li>- Burp Suite</li> </ul>
WSTG-BUSL-04	<i>Test for Process Timing</i>	<ul style="list-style-type: none"> <li>- Me-review dokumentasi <i>project</i> terkait fungsionalitas waktu</li> <li>- Mengembangkan dan mengeksekusi <i>misuse case</i></li> </ul>		
WSTG-BUSL-05	<i>Test Number of Times a Function Can be Used Limits</i>	<ul style="list-style-type: none"> <li>- Mengidentifikasi fungsi yang harus menetapkan batas waktu yang dapat dipanggil</li> <li>- Menilai dan memvalidasi apakah ada batas logis yang ditetapkan pada fungsi</li> </ul>		
WSTG-BUSL-06	<i>Testing for the Circumvention of Work Flows</i>	<ul style="list-style-type: none"> <li>- Me-review dokumentasi <i>project</i> untuk metode yang dilewati atau melalui langkah-langkah dalam urutan yang berbeda dari alur <i>business logic</i></li> <li>- Mengembangkan <i>misue case</i> dan mencoba untuk menghindari tiap <i>logic flow</i> yang teridentifikasi</li> </ul>		

DRAFT PANDUAN TEKNIS  
PELAKSANAAN IT SECURITY ASSESSMENT (ITSA)

• • •

<b>Business logic Testing</b>	<b>Nama Pengujian</b>	<b>Tujuan</b>	<b>Status (*)</b>	<b>Notes</b>
WSTG-BUSL-07	<i>Test Defenses Against Application Mis-use</i>	<ul style="list-style-type: none"> <li>- Menghasilkan catatan dari semua pengujian yang dilakukan terhadap sistem</li> <li>- Me-review tes mana yang memiliki fungsi berbeda berdasarkan <i>aggressive input</i></li> <li>- Memahami pertahanan yang ada dan verifikasi apakah hal tersebut cukup untuk melindungi sistem dari teknik <i>bypassing</i> atau tidak</li> </ul>		
WSTG-BUSL-08	<i>Test Upload of Unexpected File Types</i>	<ul style="list-style-type: none"> <li>- Me-review dokumentasi <i>project</i> untuk jenis <i>file</i> yang ditolak oleh sistem</li> <li>- Memverifikasi bahwa jenis <i>file</i> yang tidak diinginkan ditolak dan ditangani dengan aman</li> <li>- Memverifikasi bahwa unggahan <i>batch file</i> aman dan tidak mengizinkan <i>bypas</i> apapun terhadap langkah keamanan yang ditetapkan</li> </ul>		
WSTG-BUSL-09	<i>Test Upload of Malicious Files</i>	<ul style="list-style-type: none"> <li>- Mengidentifikasi fungsi unggah <i>file</i></li> <li>- Meninjau dokumentasi <i>project</i> untuk mengidentifikasi jenis <i>file</i> yang dapat diterima dan berbahaya</li> <li>- Menentukan bagaimana <i>file</i> yang diunggah diproses</li> <li>- Membuat satu set <i>file</i> berbahaya untuk pengujian</li> <li>- Mencoba mengunggah <i>file</i> berbahaya ke aplikasi dan menentukan apakah hal tersebut dapat diterima dan diproses</li> </ul>		<ul style="list-style-type: none"> <li>- Metasploit's payload generation functionality</li> <li>- Intercepting proxy</li> </ul>

<b>Client Side Testing</b>	<b>Nama Pengujian</b>	<b>Tujuan</b>	<b>Status (*)</b>	<b>Notes</b>
WSTG-CLNT-01	<i>Testing for DOM-Based Cross Site Scripting</i>	<ul style="list-style-type: none"> <li>- Identifikasi <i>sink</i> DOM</li> <li>- Buat <i>payloads</i> yang berkaitan dengan setiap jenis <i>sink</i></li> </ul>		
WSTG-CLNT-02	<i>Testing for JavaScript Execution</i>	<ul style="list-style-type: none"> <li>- Identifikasi <i>sink</i> dan kemungkinan titik injeksi JavaScript</li> </ul>		

DRAFT PANDUAN TEKNIS  
PELAKSANAAN IT SECURITY ASSESSMENT (ITSA)



<i>Client Side Testing</i>	Nama Pengujian	Tujuan	Status (*)	Notes
WSTG-CLNT-03	Testing for HTML Injection	"- Identifikasi titik injeksi HTML dan nilai tingkat keparahan konten yang diinjeksi		
WSTG-CLNT-04	Testing for Client Side URL Redirect	"- Identifikasi titik injeksi yang menangani URL atau path - Nilai lokasi yang dapat dialihkan oleh sistem		
WSTG-CLNT-05	Testing for CSS Injection	"- Mengidentifikasi titik injeksi CSS - Menilai dampak dari injeksi		
WSTG-CLNT-06	Testing for Client Side Resource Manipulation	"- Mengidentifikasi sink dengan validasi inputan yang lemah - Menilai dampak dari sumber daya yang dimanipulasi		
WSTG-CLNT-07	Test Cross Origin Resource Sharing	"- Mengidentifikasi endpoint yang mengimplementasikan CORS - Memastikan konfigurasi CORS aman atau tidak berbahaya		
WSTG-CLNT-08	Testing for Cross Site Flashing	"- Dekompilasi dan analisis kode aplikasi - Menilai input sink dan penggunaan metoda yang tidak aman		
WSTG-CLNT-09	Testing for Clickjacking	"- Pahami langkah-langkah keamanan yang ada - Lakukan penilaian seberapa ketat langkah-langkah keamanan dan apakah dapat dilewati (bypass) atau tidak		
WSTG-CLNT-10	Testing WebSockets	"- Mengidentifikasi penggunaan WebSockets - Menilai implementasi dengan menggunakan tes yang sama pada channel HTTP yang normal		- OWASP ZAP - WebSocket Client - Google Chrome Simple WebSocket Client
WSTG-CLNT-11	Test Web Messaging	"- Menilai keamanan dari pengirim (asal) pesan - Memvalidasi penggunaan metode yang aman dan memvalidasi inputannya		

DRAFT PANDUAN TEKNIS  
PELAKSANAAN IT SECURITY ASSESSMENT (ITSA)



<i>Client Side Testing</i>	Nama Pengujian	Tujuan	Status (*)	Notes
WSTG-CLNT-12	Testing Browser Storage	" - Menentukan apakah situs web menyimpan data sensitif di penyimpanan sisi klien - Memeriksa penanganan kode objek penyimpanan terhadap kemungkinan serangan injeksi, seperti pemanfaatan input yang tidak divalidasi atau library yang rentan		
WSTG-CLNT-13	Testing for Cross Site Script Inclusion	"- Mencari data sensitif di seluruh sistem - Menilai kebocoran data sensitif melalui berbagai teknik		

API Testing	Nama Pengujian	Tujuan	Status (*)	Notes
WSTG-APIT-01	Testing GraphQL	"-Menilai apakah konfigurasi yang aman dan siap produksi telah diterapkan - Memvalidasi semua kolom inputan terhadap serangan umum - Memastikan bahwa kontrol access control yang tepat telah diterapkan		<ul style="list-style-type: none"> <li>- GraphQL Playground</li> <li>- GraphQL Voyager</li> <li>- sqlmap</li> <li>- InQL (Burp Extension)</li> <li>- GraphQL Raider (Burp Extension)</li> <li>- GraphQL (Add-on for OWASP ZAP)</li> </ul>

(\*) : keterangan status terdiri dari tiga kategori yakni

1. Belum Mulai

Awal tahap pengisian *Web Security Checklist Guide* (WSTG) ditulis dengan keterangan belum mulai. Artinya belum dilaksanakan pengujian keamanan pada aplikasi web tersebut.

2. Aman

Status dikatakan aman apabila ketika dilakukan pengujian tidak menemukan kerentanan yang bersangkutan.

DRAFT PANDUAN TEKNIS  
PELAKSANAAN IT SECURITY ASSESSMENT (ITSA)



3. Ada Isu

Status dikatakan ada isu apabila ditemukan kerentanan sesuai dengan WSTG tersebut.

4. N/A

Status dikatakan N/A apabila terdapat keraguan dalam melaksanakan pengujian atau dalam artian tidak yakin aman maupun ada isu.