ANALISIS PERBANDINGAN *PENETRATION TESTING TOOL* UNTUK APLIKASI *WEB*

SKRIPSI

Untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun oleh :
Bhaskara Vito Tarigan
NIM : 12515010211003

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS BRAWIJAYA
MALANG
2017

PENGESAHAN

ANALISIS PERBANDINGAN *PENETRATION TESTING TOOL*
UNTUK APLIKASI *WEB*

SKRIPSI

Diajukan untuk memenuhi sebagian persyaratan
memperoleh gelar Sarjana Komputer

Disusun Oleh :
Bhaskara Vito Tarigan
NIM: 125150102111003

Telah diperiksa dan disetujui oleh:

Dosen Pembimbing I	Dosen Pembimbing II
Ari Kusyanti, S.T, M.Sc	Widhi Yahya, S.Kom, M.Sc
NIP: 201102 8312282 001	NIK: 201607891121101

Ketua Jurusan Teknik Informatika

Tri Astoto Kurniawan, S.T.,M.T. Ph.D.
NIP: 19710518 200312 1 001

PERNYATAAN ORISINALITAS

Saya menyatakan dengan sebenar-benarnya bahwa sepanjang pengetahuan saya, di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan oleh orang lain untuk memperoleh gelar akademik di suatu perguruan tinggi, dan tidak terdapat karya atau pendapat yang pernah ditulis atau diterbitkan oleh orang lain, kecuali yang secara tertulis disitasi dalam naskah ini dan disebutkan dalam daftar pustaka.

Apabila ternyata didalam naskah skripsi ini dapat dibuktikan terdapat unsur-unsur plagiasi, saya bersedia skripsi ini digugurkan dan gelar akademik yang telah saya peroleh (sarjana) dibatalkan, serta diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2003, Pasal 25 ayat 2 dan Pasal 70).

Malang, 15 Maret 2017

Bhaskara Vito Tarigan

NIM: 12515010211103

KATA PENGANTAR

Puji syukur penulis panjatkan hanya kepada Tuhan Yesus Kristus karena atas anugerah dan kasih karunia-Nya penulis mampu menyelesaikan skripsi ini dengan judul "Analisa Perbandingan *Penetration Testing Tool* Untuk Aplikasi *Web*." dengan baik. Skripsi ini merupakan salah satu syarat untuk memenuhi persyaratan akademis guna menyelesaikan studi di Fakultas Ilmu Komputer Universitas Brawijaya.

Skripsi ini merupakan bagian dari tugas akhir penulis selama mengikuti perkuliahan dan sebagai salah satu syarat untuk mencapai gelar Sarjana Komputer di Fakultas Ilmu Komputer, Universitas Brawijaya. Melalui kesempatan ini, Penulis ingin menyampaikan rasa hormat dan terima kasih yang sebesar-besarnya kepada semua pihak yang telah memberikan bantuan dan dukungan selama pengerjaan skripsi, diantaranya:

1. Tuhan Yesus Kristus atas rahmat dan anugerah-Nya sehingga skripsi ini dapat terselesaikan dengan baik.
2. Kedua orangtua , adik , dan kakak penulis yang turut mendoakan dan memberi semangat sehingga skripsi selesai sampai kelulusan kuliah.
3. Ibu Ari Kusyanti, S.T, M.Sc. selaku Dosen Pembimbing 1 dan Bapak Widhi Yahya, S.Kom., M.Sc. selaku Dosen Pembimbing 2 atas kesediaan waktu memberikasn arahan dan kesabarana dalam membimbing penulis selama proses penyusunan skripsi ini.
4. Agus Wahyu Widodo, S.T. M,Cs selaku Kepala Prodi Informatika / Ilmu Komputer yang membantu memimpin prodi ini menjadi lebih baik ke depannya.
5. Teman yang selalu ada setiap harinya Savionita Devi Indriyana, menemani saya dan memberi semangat kepada saya setiap proses pengerjaan skripsi saya.
6. Teman-teman laskar anak soleh 2012 yang sesama pejuang skripsi yang memberikan semangat dan doa.
7. Keluarga PMK Daniel, GP GPIB Immanuel Malang, tempat dimana penulis dapat belajar untuk semakin dewasa dan bertumbuh dalam Tuhan, memberikan warna baru, sukacita, dan keceriaan. Bahkan disaat penulis ada dalam masa kesukaran. Bangga bisa jadi bagian dari setiap komunitas ini.
8. Sahabat-sahabatku, Maria Tifanny Podung , Syela Meirose Podung , dan Victor Bela Sendoro Hia yang mau membantu dalam pengerjaan skripsi , memberikan semangat, dan doa sampai akhirnya bisa menyelesaikan skripsi ini.
9. Sahabat dari Universitas Brawijaya angkatan 2012. Terima kasih atas dukungan dan bantuan selama menempuh studi.
10. Semua pihak yang telah membantu penulis dalam pengerjaan skripsi yang tidak bisa disebutkan satu persatu.

Penulis menyadari bahwa tugas akhir ini tidak lepas dari kekurangan dan kesalahan. Penulis mengharapkan kritik dan saran dari berbagai pihak demi penyempurnaan skripsi ini. Akhir kata, semoga skripsi ini dapat bermanfaat dan berguna bagi pembaca terutama mahasiswa FILKOM Universitas Brawijaya. Terima kasih.

Malang, 28 April 2017

Penulis
vitobhaskara.cisco@gmail.com

DAFTAR ISI

LEMBAR PENGESAHAN ..... **ii**

PERNYATAAN ORISINALITAS ..... **iii**

KATA PENGANTAR ..... **iv**

DAFTAR ISI ..... **vi**

DAFTAR GAMBAR ..... **x**

DAFTAR TABEL ..... **xiv**

ABSTRAK ..... **xv**

BAB 1 PENDAHULUAN **1**

1.1 Latar Belakang 1

1.2 Rumusan Masalah 2

1.3 Manfaat 2

1.4 Batasan Masalah 2

1.5 Sistematikas Pembahasan 2

BAB 2 LANDASAN KEPUSTAKAAN **4**

2.1 *Penetration Testing* 4

2.1.1 *Penetration Testing Tool*. 4

2.2 Tipe *Penetration Testing* 5

2.3 Legalitas *Penetration Testing* 5

2.3.1 Perangkat *Penetration Testing* 6

2.4 Penilaian Kerentanan 6

2.4.1 Penilaian Kerentanan & Teknik Uji Penetrasi 6

2.4.1.1 Teknik Penilaian Kerentanan 6

2.4.1.2 *Static Analysis* 6

2.4.1.3 *Manual Testing* 7

2.4.1.4 *Automated Testing* 7

2.4.2 *Penetration Testing Techniques* 7

2.4.2.1 *Black Box Testing* 7

2.4.2.2 *Grey Box Testing* 7

2.4.2.3 *White Box Testing* 7

2.4.3 Proses *Penetration Testing* 8
2.4.4 Kerentanan... 8
2.4.4.1 *Cross-Site Scripting (XSS)* 8
2.4.4.2 *SQL Injection* 9
2.4.4.3 *Directory Traversal* 9
2.4.4.4 *Authentication* 9
2.4.4.5 *Captcha* 9
2.4.4.6 *Authorization* 9
2.4.4.7 *Mongo DB Injection* 10
2.4.4.8 *File Include* 10
2.4.4.9 *Code Injection* 10
2.4.4.10 *Command Injection* 11
2.5 *Penetration Testing Tool* 11
2.5.1 *W3af* 11
2.5.1.1 Kelebihan dan Kekurangan *W3af* 11
2.5.2 *Wapiti* 11
2.5.2.1 Kelebihan dan Kekurangan *Wapiti* 11
2.5.3 *Arachni* 12
2.5.3.1 Kelebihan dan Kekurangan *Arachni* 12
2.6 *Ubuntu* 12
2.7 *Independent Samples One Way ANOVA* 13
BAB 3 METODOLOGI PENELITIAN 14
3.1 Jenis Penelitian 14
3.2 Studi Literatur 15
3.3 Lingkungan Penelitian 15
3.4 Metodologi Penelitian 15
3.4.1 Identifikasi *Web* 15
3.4.2 Analisis *Penetration Testing Tool* 15
3.5 Tabel Pengujian Otomatis 16
3.6 Pengujian.... 16

3.7 Pengujian Manual 17
3.8 Pengujian Otomatis 17
3.9 Skenario Pengujian 18
3.9.1 Skenario Pengujian Otomatis *W3af* 19
3.9.2 Skenario Pengujian Otomatis *Wapiti* 20
3.9.3 Skenario Pengujian Otomatis *Arachni* 21
BAB 4 HASIL 23
4.1 Identifikasi 23
4.2 Hasil Pengujian 24
4.2.1 Hasil Pengujian Manual 24
4.2.1.1 *XSS* 24
4.2.1.2 *SQL Injection* 25
4.2.1.3 *Directory Traversal* 26
4.2.1.4 *Command Injection* 26
4.2.1.5 *File Include* 27
4.2.1.6 *Code Injection* 27
4.2.1.7 *Authentication* 28
4.2.1.8 *Authorization* 28
4.2.1.9 *Captcha*..... 28
4.2.1.10 *Mongo DB Injection* 29
4.3 Hasil Pengujian Otomatis 29
4.3.1 Hasil Pengujian Otomatis *W3af* 29
4.3.1.1 Hasil Waktu Identifikasi *Tool W3af* 32
4.3.2 Hasil Pengujian Otomatis *Wapiti* 33
4.3.2.1 Hasil Waktu Identifikasi *Tool Wapiti* 37
4.3.3 Hasil Pengujian Otomatis *Arachni* 38
4.3.3.1 Hasil Waktu Identifikasi *Tool Arachni* 42
4.3.4 Hasil Tabel Pengujian Otomatis 43
BAB 5 PEMBAHASAN 47
5.1 Pembahasan Pengujian 47

5.1.1 *XSS* 47
5.1.1.1 Pengujian Manual 47
5.1.1.2 Pengujian Otomatis 48
5.1.2 *SQL Injection* 49
5.1.2.1 Pengujian Manual 49
5.1.2.2 Pengujian Otomatis 49
5.1.3 *Directory Traversal* 50
5.1.3.1 Pengujian Manual 50
5.1.3.2 Pengujian Otomatis 51
5.14 *Command Injection* 52
5.1.4.1 Pengujian Manual 52
5.1.4.2 Pengujian Otomatis 52
5.1.5 *Code Injection* 53
5.1.5.1 Pengujian Manual 53
5.1.5.2 Pengujian Otomatis 53
5.1.6 *File Include* 55
5.1.6.1 Pengujian Manual 55
5.1.6.2 Pengujian Otomatis 55
5.1.7 *Authentication* 57
5.1.7.1 Pengujian Manual 57
5.1.7.2 Pengujian Otomatis 57
5.1.8 *Authorization...* 58
5.1.8.1 Pengujian Manual 58
5.1.8.2 Pengujian Otomatis 59
5.1.9 *Captcha* 60
5.1.9.1 Pengujian Manual 60
5.1.9.2 Pengujian Otomatis 61
5.1.10 *Mongo DB Injection* 62
5.1.10.1 Pengujian Manual 62
5.1.10.2 Pengujian Otomatis 62

BAB 6 PENUTUP..... 64
6.1 Kesimpulan. 64
6.2 Saran 64
DAFTAR PUSTAKA 65

DAFTAR GAMBAR

Gambar 2.1 Proses Uji Penetrasi 9
Gambar 3.1 Flowchart Utama Langkah Penelitian 15
Gambar 3.2 Proses Pengujian *W3af* 19
Gambar 3.3 Proses Melakukan *Scanning* Pada *W3af* 20
Gambar 3.4 Proses Pengujian Otomatis *Wapiti* 20
Gambar 3.5 Proses *Scanning* Pada *Wapiti* 21
Gambar 3.6 Proses Pengujian Otomatis *Arachni* 21
Gambar 3.7 Proses *Scanning* Pada *Arachni* 22
Gambar 3.8 Proses Memulai *Scanning* Pada *Arachni* 22
Gambar 4.1 *Web for pentester*... 23
Gambar 4.2 *Web for pentester 2* 24
Gambar 4.3 Hasil Pengujian Manual *XSS* 25
Gambar 4.4 Hasil Pengujian Manual *SQL Injection* 25
Gambar 4.5 Hasil Pengujian Manual *Directory Traversal* 26
Gambar 4.6 Hasil Pengujian Manual Commad Injection 26
Gambar 4.7 Hasil Pengujian Manual *File Include* 27
Gambar 4.8 Hasil Pengujian Manual *Code Injection* 27
Gambar 4.9 Hasil Pengujian Manual *Authentication* 28
Gambar 4.10 Hasil Pengujian Manual *Authorization* 28
Gambar 4.11 Hasil Pengujian Manual *Captcha* 28
Gambar 4.12 Hasil Pengujian Manual *Mongo DB* 29
Gambar 4.13 Hasil Pengujian Otomatis *W3af Web for pentester* 29
Gambar 4.14 Hasil Kerentanan *XSS* Dengan *Tool W3af* 30
Gambar 4.15 Hasil Kerentanan *Command injection* Dengan *Tool W3af* 30
Gambar 4.16 Hasil Kerentanan *File Include* Dengan *Tool W3af* 30
Gambar 4.17 Hasil Kerentanan *Directory traversal* Dengan *Tool W3af* 30
Gambar 4.18 Hasil Kerentanan *Code injection* Dengan *Tool W3af* 30
Gambar 4.19 Hasil Pengujian Otomatis *Arachni Web for pentester 2* 31
Gambar 4.20 Hasil Kerentanan *SQL Injection* Dengan *Tool W3af* 31
Gambar 4.21 Hasil Kerentanan *Authentication* Dengan *Tool W3af* 31
Gambar 4.22 Hasil Kerentanan *Authorization* Dengan *Tool W3af* 31

Gambar 4.23 Hasil Kerentanan *Captcha* Dengan *Tool W3af* 32
Gambar 4.24 Hasil Kerentanan *Mongo DB Injection* Dengan *Tool W3af* 32
Gambar 4.25 Hasil Identifikasi Waktu Web For Pentester 32
Gambar 4.26 Hasil Identifikasi Waktu Web For Pentester 2 32
Gambar 4.27 Hasil Pengujian Otomatis *Wapiti Web For Pentester* 33
Gambar 4.28 Hasil Kerentanan *XSS* Dengan *Tool Wapiti* 33
Gambar 4.29 Hasil Kerentanan *SQL Injection* Dengan *Tool Wapiti* 33
Gambar 4.30 Hasil Kerentanan *File Include* Dengan *Tool Wapiti* 34
Gambar 4.31 Hasil Kerentanan *Command injection* Dengan *Tool Wapiti* 34
Gambar 4.32 Hasil Pengujian Otomatis *Wapiti Web for pentester 2* 34
Gambar 4.33 Hasil Kerentanan *Authentication* Dengan *Tool Wapiti* 35
Gambar 4.34 Hasil Kerentanan *Authorization* Dengan *Tool Wapiti* 35
Gambar 4.35 Hasil Kerentanan *Captcha* Dengan *Tool Wapiti* 35
Gambar 4.36 Hasil Kerentanan *Directory Traversal* Dengan *Tool Wapiti* 36
Gambar 4.37 Hasil Kerentanan *Code Injection* Dengan *Tool Wapiti* 36
Gambar 4.38 Hasil Kerentanan *Mongo DB Injection* Dengan *Tool Wapiti* 36
Gambar 4.39 Hasil Identifikasi Waktu Web For Pentester 37
Gambar 4.40 Hasil Identifikasi Waktu Web For Pentester 2 37
Gambar 4.41 Hasil Pengujian Otomatis *Arachni Web for pentester* 38
Gambar 4.42 Hasil Kerentanan *XSS* Dengan *Tool Arachni* 38
Gambar 4.43 Hasil Kerentanan *SQL Injection* Dengan *Tool Arachni* 39
Gambar 4.44 Hasil Kerentanan *Code injection* Dengan *Tool Arachni* 39
Gambar 4.45 Hasil Kerentanan *Directory traversal* Dengan *Tool Arachni* 39
Gambar 4.46 Hasil Kerentanan *Command injection* Dengan *Tool Arachni* 40
Gambar 4.47 Hasil Pengujian Otomatis *Arachni Web for pentester 2* 40
Gambar 4.48 Hasil Kerentanan *Captcha* Dengan *Tool Arachni* 40
Gambar 4.49 Hasil Kerentanan *Authentication* Dengan *Tool Arachni* 41
Gambar 4.50 Hasil Kerentanan *Authorization* Dengan *Tool Arachni* 41
Gambar 4.51 Hasil Kerentanan *Mongo DB* Dengan *Tool Arachni* 41
Gambar 4.52 Hasil Kerentanan *File Include* Dengan *Tool Arachni* 41
Gambar 4.53 Hasil Identifikasi Waktu Web For Pentester 42
Gambar 4.54 Hasil Identifikasi Waktu Web For Pentester 2 42

Gambar 5.1 Hasil Pengujian Manual *XSS* 47
Gambar 5.2 Hasil Pengujian Otomatis *XSS W3af* 48
Gambar 5.3 Hasil Pengujian Otomatis *XSS Wapiti* 48
Gambar 5.4 Hasil Pengujian Otomatis *XSS Arachni* 48
Gambar 5.5 Hasil Pengujian Manual *SQL Injection* 49
Gambar 5.6 Hasil Pengujian Otomatis *SQL Injection W3af* 49
Gambar 5.7 Hasil Pengujian Otomatis *SQL Injection Wapiti* 49
Gambar 5.8 Hasil Pengujian Otomatis *SQL Injection Arachni* 50
Gambar 5.9 Hasil Pengujian Manual *Directory Traversal* 50
Gambar 5.10 Hasil Pengujian Otomatis *Directory Traversal W3af* 51
Gambar 5.11 Hasil Pengujian Otomatis *Directory Traversal Wapiti* 51
Gambar 5.12 Hasil Pengujian Otomatis *Directory Traversal Arachni* 51
Gambar 5.13 Hasil Pengujian Manual *Command Injection* 52
Gambar 5.14 Hasil Pengujian Otomatis *Command Injection W3af* 52
Gambar 5.15 Hasil Pengujian Otomatis *Command Injection Wapiti* 52
Gambar 5.16 Hasil Pengujian Otomatis *Command Injection Arachni* 53
Gambar 5.17 Hasil Pengujian Manual *Code Injection* 53
Gambar 5.18 Hasil Pengujian Otomatis *Code Injection W3af* 53
Gambar 5.19 Hasil Pengujian Otomatis *Code Injection Wapiti* 54
Gambar 5.20 Hasil Pengujian Otomatis *Code Injection Arachni* 54
Gambar 5.21 Hasil Pengujian Manual *File Include* 55
Gambar 5.22 Hasil Pengujian Otomatis *File Include W3af* 55
Gambar 5.23 Hasil Pengujian Otomatis *File Include Wapiti* 56
Gambar 5.24 Hasil Pengujian Otomatis *File Include Arachni* 56
Gambar 5.25 Hasil Pengujian Manual *Authentication* 57
Gambar 5.26 Hasil Pengujian Otomatis *Authentication W3af* 57
Gambar 5.27 Hasil Pengujian Otomatis *Authentication Wapiti* 58
Gambar 5.28 Hasil Pengujian Otomatis *Authentication Arachni* 58
Gambar 5.29 Hasil Pengujian Manual *Authorization* 58
Gambar 5.30 Hasil Pengujian Otomatis *Authorization W3af* 59
Gambar 5.31 Hasil Pengujian Otomatis *Authorization Wapiti* 59
Gambar 5.32 Hasil Pengujian Otomatis *Authorization Arachni* 60

Gambar 5.33 Hasil Pengujian Manual *Captcha* 60
Gambar 5.34 Hasil Pengujian Otomatis *Captcha W3af* 61
Gambar 5.35 Hasil Pengujian Otomatis *Captcha Wapiti* 61
Gambar 5.36 Hasil Pengujian Otomatis *Captcha Arachni* 61
Gambar 5.37 Hasil Pengujian Manual *Mongo DB Injection* 62
Gambar 5.38 Hasil Pengujian Otomatis *Mongo DB Injection W3af* 62
Gambar 5.39 Hasil Pengujian Otomatis *Mongo DB Injection Wapiti* 62
Gambar 5.40 Hasil Pengujian Otomatis *Mongo DB Injection Arachni* 63

DAFTAR TABEL

Tabel 2.1 Persyaratan Sistem Pada *Ubuntu* 14
Tabel 3.1 Tabel Pengujian Otomatis 17
Tabel 4.1 Hasil Waktu Identifikasi *Tool W3af* 32
Tabel 4.2 Hasil Waktu Identifikasi *Tool Wapiti* 37
Tabel 4.3 Hasil Waktu Identifikasi *Tool Arachni* 42
Tabel 4.4 *One-Way ANOVA Web For Pentester dan Web For Pentester 2* 43
Tabel 4.5 Hasil Percobaan 1 pada Pengujian Otomatis 44
Tabel 4.6 Hasil Percobaan 2 pada Pengujian Otomatis 44
Tabel 4.7 Hasil Percobaan 3 pada Pengujian Otomatis 45
Tabel 4.8 Hasil Percobaan 4 pada Pengujian Otomatis 45
Tabel 4.9 Hasil Percobaan 5 pada Pengujian Otomatis 45

ABSTRAK

BHASKARA VITO TARIGAN, Jurusan Teknik Informatika, Fakultas Ilmu Komputer Universitas Brawijaya, Maret 2016, *Analisa Perbanding Penetration Testing Tool Untuk* Aplikasi *Web*, Dosen Pembimbing: Ari Kusyanti , S.T, M.Sc, Widhi Yahya, S.Kom, M.Sc.

Abstrak - Uji penetrasi adalah serangkaian kegiatan yang dilakukan untuk mengidentifikasi dan mengeksploitasi kerentanan keamanan. Untuk membantu mengkonfirmasi efektivitas atau ketidakefektifan langkah-langkah keamanan yang telah dilaksanakan. Paper ini membahas dan metodologi dalam melakukan uji penetrasi dengan ketiga *penetration testing tool* yaitu *W3af*, *Wapiti*, dan *Arachni*. Metodologi uji penetrasi mencakup tiga tahap: persiapan pengujian, tes dan analisa tes. Tahap uji coba melibatkan langkah-langkah berikut: analisa kerentanan, pengumpulan informasi , dan analisa *tool*. Hasil dari pengujian perbandingan ini nantinya dibandingkan dengan tool yang digunakan untuk mengetahui tool mana yang lebih banyak mengidentifikasi kerentanan. dengan Paper ini mengambarkan secara lebih lanjut untuk menganalisa perbandingan *penetration testing tool* yang menentukan dan mengetahui serangan-serangan dimana saja yang dapat bisa terdeteksi dari ketiga *tool* yang telah diuji dari kerentanan yang ada pada aplikasi *web*.

Kata kunci: *Uji Penetrasi, W3af, Wapiti, Arachni, Kerentanan*

ABSTRACT

BHASKARA VITO TARIGAN, Department of Informatics, Brawijaya University, March 2016, Comparison Analysis *Penetration Testing Tool* for *Web Applications,* Advisor: Ari Kusyanti, S.T, M.Sc, Widhi Yahya, Kom, M.Sc.

Abstract - Penetration testing is a series of activities undertaken to identify and exploit security vulnerabilities. This helps confirm the effectiveness or ineffectiveness of security measures that have been implemented. This paper discusses the methodology in conducting penetration tests with a third tool penetration test is W3af, Wapiti, and Arachni. Penetration testing methodology includes three phases: test preparation, test and analysis test. Pilot phase involves the following steps: vulnerability analysis, information collection and analysis tool. This paper portrait is more to comparison tool that determines the penetration test and find out attacks anywhere that can be detected from the three tool that have been tested on existing vulnerabilities in web applications.

Keywords: Penetration Testing, W3af, Wapiti, Arachni, Vulnerability

BAB 1
PENDAHALUAN

1.1 Latar Belakang

Keamanan adalah salah satu dari masalah utama sistem informasi. pertumbuhan konektivitas dari komputer melalui Internet, meningkatnya ekstensibilitas sistem, dan pertumbuhan ukuran yang tidak terkendali dari ukuran dan kompleksitas sistem telah membuat keamanan perangkat lunak sebuah masalah yang lebih besar sekarang daripada di masa lalu. Selain itu, satu kepentingan bisis untuk secara cukup melindungi aset informasi organisasi dengan mengikuti pendekatan yang komprehensif dan terstruktur untuk memberikan perlindungan dari risiko yang mugkin dihadapi sebuah organisasi (Aileen G., 2011). Dalam upaya untuk memecahkan masalah keamanan dan mematuhi peraturan keamanan yang berlaku, pakar keamanan telah mengembangkan berbagai metode jaminan keamanan yang mencakup bukti kebenaran desain berlapis, lingkungan rekayasa perangkat lunak dan uji penetrasi.

Uji penetrasi adalah metode komprehensif untuk menguji dasar komputasi yang lengkap, terpadu,operasional, dan terpercaya yang terdiri dari perangkat keras, perangkat lunak dan manusia.Proses ini melibatkan analisis aktif sistem untuk setiap potensi kerentanan, termasuk konfigurasi sistem yang buruk atau tidak tepat, kelemahan *hardware* dan *software*, dan kelemahan operasional dalam proses atau penanggulangan teknis (Aileen G., 2011)

Beberapa *tool* yang digunakan dalam uji penetrasi ini adalah *W3af , Wapiti,* dan *Arachni* dari setiap *tool* tersebut dapat melakukan perbandingan dengan uji penetrasi yang memungkinkan *tool* tersebut apakah dapat mendeteksi semua kerentanan pada aplikasi *web*.Uji penetrasi berbeda dari pengujian fungsional keamanan. Yang belakangan ini menunjukkan perilaku yang benar dari kontrol keamanan sistem, sedangkan uji penetrasi menentukan kesulitan bagi seseorang untuk menembus kontrol keamanan organisasi terhadap akses yang tidak berwenang pasa sistem informasi dan informasinya. Hal ini dilakukan dengan mensimulasikan pengguna yang tidak berwenang untuk menyerang sistem, baik menggunakan alat otomatis atau metode manual atau kombinasi keduanya.

Tujuan dari skripsi ini adalah untuk menganalisis perbandingan *penetration testing tool* yang menentukan dan mengetahui serangan-serangan dimana saja yang dapat terdeteksi dari ketiga *tool* yang telah diuji dari kerentanan yang ada pada sistem aplikasi *web www.pentesterlab.com*.

1.2 Rumusan Masalah

Beberapa masalah yang tercakup dalam pembuatan Tugas Akhir ini antara lain adalah :

1. Bagaimana membandingkan *penetration testing tool W3af , Wapiti , dan Arachni* pada aplikasi web *www.pentesterlab.com*
2. Bagaimana hasil analisis perbandingan tools *W3af*, *Arachni*, dan *Wapiti* pada kerentanan aplikasi web.

Tujuan dilakukannya penelitian ini adalah :

a) Mendapatkan hasil identifikasi perbandingan dari *tool W3af , Wapiti ,* dan *Arachni* pada aplikasi web yang diuji .
b) Mengetahui perbandingan sistem dari ketiga tool uji penetrasi.

1.3 Manfaat

Manfaat yang dapat diperoleh dari uji penetrasi antara lain :

- Bagi penulis, bisa memberikan tambahan ilmu dan tentang *tool* penetrasi pada penelitian lanjutan untuk pengembangan keamanan jaringan di tempat lain.
- Memberikan suatu alternatif source dengan biaya rendah dan hasil yang maksimal

1.4 Batasan Masalah

Dalam pengerjaan skripsi ini terdapa beberapa batasan masalah antara lain:

1. Pengujian akan dilakukan dalam ruang lingkup jaringan dan sistem *web* aplikasi berdasarakan informasi yang didapat dari internet.
2. Penelitian akan memfokuskan pada pengujian *tool* penetrasi dari aspek-aspek teknis seperti aplikasi *web*, *software* dan sistem operasi.
3. *Server* target adalah aplikasi *web* yang di uji kerentanannya.

1.5 Sistematika pembahasan

Sistematika pemabahasan dalam penelitian ini tersusun atas enam bab yang terdiri dari atas pendahuluan,landasan pustaka,metode peneletian yang dilkakukan,hasil,pembahasan,serta penutup yang terdiri atas pengambilan kesimpulan dan pemberian saran. Sistematika penulisan yang digunakan dalam penyusunan laporan penelitian ini adalah sebagai berikut.

BAB I PENDAHULUAN
Memuat latar belakang, rumusan maslah, ruang lingkup, tujuan , dan sistematika penulisan

BAB II LANDASAN KEPUSTAKAAN

Mengkaji teori-teori yang menunjang skrispsi ini , diantaranya tentang *Penetration Testing Tool W3AF , Arachni ,* dan *Wapiti* Vulnerabilty serta hasil nilai yang dapat dijadikan hasil dari perbandingan tersebut.

BAB III METODOLOGI

Bab ini menjelaskan secara garis besar langkah-langkah yang dilakukan dalama penelitian ini. Bab ini terdiri dari Bahan Penelitian dan alat penelitian

BAB IV HASIL

Hasil dari uji coba dalam uji *tool* penetrasi dalam aplikasi *web* (*Penetration Testing*) dapat tercapai dan dapat dikembangkan lagi dalam setiap uji coba berikutnya.

BAB V PEMBAHASAN

Penjelasan mengenai hasil dari pengujian manual dan pengujian automatis dengan menggunakan *tool W3af,Arachni,*dan *Wapiti* pada aplikasi *web www.pentesterlab.com*

BAB VI PENUTUP

Berisi kesimpulan dan saran yang diperoleh dari analisis untuk menentukan hasil dari perbandingan yang telah di uji dari uji coba pada *penetration testing tool.*

BAB 2
LANDASAN KEPUSTAKAAN

2.1 Penetration Testing

Tujuan utama dari penilaian kerentanan adalah mengidentifikasi kerentanan keamanan di bawah keadaan yang dikendalikan sehingaa mereka dapat dihilangkan sebelum pengguna yang tidak berwenang mengeksploitasi mereka. Ahli sistem komputasi menggunakan uji penetrasi untuk mengatasi masalah yang melekat dalam penilaian kerentanan, dengan fokus pada kerentanan dengan tingkat keparahan yang tinggi. Uji penetrasi adalah alat penilaian jaminan bernilai yang menguntungkan baik bisnis dan operasinya. Dari segi operasional, pengujian penetrasi membantu membentuk strategi keamanan informasi melalui indentifikasi kerentanan yang cepat dan akurat; penghapusan proaktif dari risiko yang teridentifikasi; pelaksanaan tindakan korektif; dan peningkatan pengetahuan TI (Bacudio, Aileen G., 2011).

Uji penetrasi memberikan informasi rinci tentang ancaman keamanan aktual, yang dapat dieksploitasi jika tercakup dalam doktrin dan proses keamanan organisasi. Hal ini akan membantu organisasi untuk mengidentifikasi dengan cepat dan secara akurat, pontesi kerentanan dan kerentanan yang nyata.Dengan memberikan informasi yang diperlukan untuk secara efektif dan efisien mengisolasi dan memprioritaskan kerentanan, uji penetrasi dapat membantu penyempurnaan dan perubahan konfigurasi pengujian atau patch organisasi untuk secara proaktif menghilangkan risiko yang diidentifikasi.Uji penetrasi juga dapat membantu organisasi mengukur dampak dan kemungkinan kerentanan. Hal ini akan memungkinkan organisasi untuk memprioritaskan dan menerapkan langkah-langkah korektif untuk kerentanan yang telah diketahui dan dilaporkan (Bacudio, Aileen G., 2011). Proses melaksanakan uji memerlukan banyak waktu, tenaga dan pengetahuan dalam menangani kompleksitas ruang pengujian. Uji penetrasi karena itu akan meningkatkan tingkat pengetahuan dan keterampilan siapapun yang terlibat dalam prosesnya.

2.1.1 Penetration Testing Tool

Berikut adalah beberapa *tools* yang digunakan dalam melakukan *penetration testing* :

1. *W3af*
2. *Wapiti*
3. *Arachni*
4. Metasploit
5. Acunetix
6. Fasttrack

7. OpenVas
8. Nikto

2.2 Tipe *Penetration Testing*

Terdapat tiga daerah untuk menguji pengujian penetrasi: struktur fisik dari sistem, struktur logis dari sistem, dan respon atau alur kerja dari sistem (Bacudio, Aileen G., 2011). Ketiga daerah yang menentukan ruang lingkup dan jenis uji penetrasi yaitu jaringan, aplikasi , dan rekayasa sosial.

Uji penetrasi jaringan adalah cara yang etis dan aman untuk mengidentifikasi kesenjangan atau kelemahan keamanan dalam desain, implementasi atau operasi jaringan organisasi. Penguji melakukan analsis dan eksploitasi untuk menilai apakah modem, perangkat akses remote dan koneksi pemeliharaan dapat digunakan untuk menembus target pengujian.Uji penetrasi aplikasi adalah simulasi serangan yang dimaksudkan untuk mengekspos efektivitas kontrol keamanan aplikasi dengan melihat risiko yang ditimbulkan oleh kerentanan nyata yang dapat dieksploitasi. Meskipun organisasi menggunakan *firewall* dan *monitoring system* untuk melindungi informasi, keamanan masih berbahaya karena lalu lintas dapat diizinkan untuk melewati *firewall*.

Rekayasa sosial memangsa interaksI manusia untuk mendapatkan atau membahayakan informasi tentang organisasi dan sistem komputer (Bacudio, Aileen G., 2011). Hal ini digunakan untuk menentukan tingkat kesaaran keamanan antara karyawan dalam organisasi yang memiliki sistem target. Hal ini berguna untuk menguji kemampuan organisasi untuk mencegah akses yang tidak berwenang pada informasi dan sistem informasinya.Bahwa ini adalah tes yang di fokuskan pada alur kerja organisasi.

2.3 Legalitas *Penetration Testing*

Tindakan yang termasuk dalam kegiatan penetration testing, aspek hukumnya telah diatur pada bab VII Perbuatan yang Dilarang, dalam pasal 30 Undang-undang Republik Indonesia Nomor 11 Tahun 2008 Tentang Informasi dan Transaksi Elektronik (UU ITE).

(1) Setiap Orang dengan sengaja dan tanpa hak atau melawan hukum mengakses Komputer dan/atau Sistem Elektronik milik Orang lain dengan cara apa pun.

(2) Setiap Orang dengan sengaja dan tanpa hak atau melawan hukum mengakses Komputer dan/atau Sistem Elektronik milik Orang lain dengan cara apa pun dengan tujuan untuk memperoleh Informasi Elektronik dan/atau Dokumen Elektronik.

(3) Setiap Orang dengan sengaja dan tanpa hak atau melawan hukum mengakses Komputer dan/atau Sistem Elektronik milik Orang lain dengan cara apa pun dengan melanggar, menerobos, melampaui, atau menjebol sistem pengamanan.

Kegiatan ini sah dan tidak melawan hukum karena penulis dan pengelola sistem membuat kesepakatan secara tertulis tentang poin-poin kegiatan *penetration testing* yang akan dilakukan.

2.4 Penilaian Kerentanan

Vulnerability adalah kelemahan dalam aplikasi yang dapat menjadi penghambat implementasi atau kekurangan desain yang memungkinkan penyerang untuk membahayakan pengguna dari aplikasi dan mendapatkan hak istimewa tambahan Kerentanan adalah potensi risiko untuk sistem. Penyerang menggunakan kerentanan tersebut untuk mengeksploitasi sistem dan mendapatkan akses dan informasi yang bukan kewenangannya (Goel, Jai Narayan. 2015). Sebuah sistem yang bebas kerentanan dapat menyediakan lebih banyak jaminan informasi dan sistem keamanan. Meskipun hampir mustahil untuk memiliki 100% sistem yang bebas kerentanan, tapi dengan menghilangkan kerentanannya sebanyak mungkin, dan dapat meningkatkan sistem keamanan.

Penilaian kerentanan adalah proses pemindaian sistem atau perangkat lunak atau jaringan untuk mengetahui kelemahan dan celah dalam hal tersebut. Celah ini dapat menjadi '*back door*' bagi penyerang untuk menyerang korban. Sebuah sistem mungkin memiliki kerentanan kontrol akses, kerentanan kondisi perbatasan, kerentanan validasi input, kerentananan otentikasi, kerentanan kelemahan konfigurasi, dan kerentanan penanganan eksepsi.

Uji penetrasi adalah langkah berikutnya setelah penilaian kerentanan. Uji penetrasi adalah mencoba untuk mengeksploitasi sistem dengan cara yang disahkan untuk mengetahui kemungkinan eksploitasi dalam sistem. Dalam uji penetrasi, penguji memiliki kewenangan untuk melakukan uji penetrasi dan secara seksama mengeksploitas sistem dan mencari tahu kemungkinan eksploitasi. (Goel, Jai Narayan. 2015).

2.4.1 Penilaian Kerentanan & Teknik Uji Penetrasi

2.4.1.1 Teknik Penilaian Kerentanan

Pada bagian menjelaskan beberapa teknik VAPT populer (Goel, Jai Narayan. 2015).

2.4.1.2 *Static Analysis*

Dalam teknik *Static Analysis* tidak melakukan kasus pengujian atau mengeksploitasi apapun. Menganalisis struktur kode dan isi dari sistem ,dengan teknik ini dapat mengetahui tentang semua jenis kerentanan. Dalam teknik ini tidak mengeksploitasi sistem, sehingga tidak akan ada pengaruh buruk dari pengujian pada sistem. Salah satu kelemahan utama dari teknik ini adalh bahwa hal itu cukup lambat dan membutuhkan banyak waktu untuk dilaksanakan.

2.4.1.3 *Manual Testing*

Teknik *Manual Testing* tidak memerlukan alat atau perangkat lunak apapun untuk mengetahui kerentanan. Di sini pengujian menggunakan pengetahuan dan pengalaman untuk mengetahui kerentanan dalam sistem. Pengujian ini dapat dilakukan dengan rencana pengujian yang dipersiapkan (pengujian manual sistematis) atau tanpa rencana pengujian apapun (pengujian manual eksplorasi). Teknik ini biayanya lebih murah dibandingkan dengan teknik lain, karena tidak perlu membeli alat penilaian kerentanan untuk teknik ini.

2.4.1.4 *Automated Testing*

Automated Testing menggunakan alat pengujian kerentanan otomatis untuk mengetahui kerentanan dalam sistem. Alat-alat ini melaksanakan semua kasus pengujian untuk mengetahui kerentanan. Ini mengurangi waktu yang dibutuhkan untuk melakukan pengujian. Karena alat ini, pengujian yang berulang juga dapat dilakukan dengan sangat mudah. Pengujian otomatis memberikan akurasi yang lebih baik daripada apa yang disediakan teknik lainnya. Hal tersebut hanya membutuhkan waktu yang singkat dan kasus pengujian yang sama dapat digunakan untuk operasi masa depan. Tapi alat tersebut meningkatkan biaya pengujian. Sebuah alat tunggal tidak mampu untuk mengetahui semua jenis kerentanan. Sehingga hal ini meningkatkan total biaya untuk melakukan penilaian kerentanan.

2.4.2 Teknik Uji Penetrasi

2.4.2.1 Black Box Testing

Dalam teknik ini dijelaskan memiliki pengetahuan sebelumya dari arsitektur jaringan atau sistem jaringan pengujian. Biasanya pengujian kotak hitam (*black box testing*) dilakukan dari jaringan eksternal ke jaringan internal. Penguji harus menggunakan keahlian dan keterampilannya untuk melakukan pengujian ini.

2.4.2.2 Grey Box Testing

Dalam teknik ini dijelaskan memiliki beberapa pengetahuan parsial jaringan pengujian.Penguji tidak memiliki pengetahuan tentang arsitektu jaringan yang lengkap, tapi beberapa informasi dasar dari jaringan pengujian dan konfigurasi sistem.Sebenarnya, kotak pengujian abu-abu adalah kombinasi dari kedua teknik lainnya. Hal ini dapat dilakukan dari jaringan internal atau eksternal.

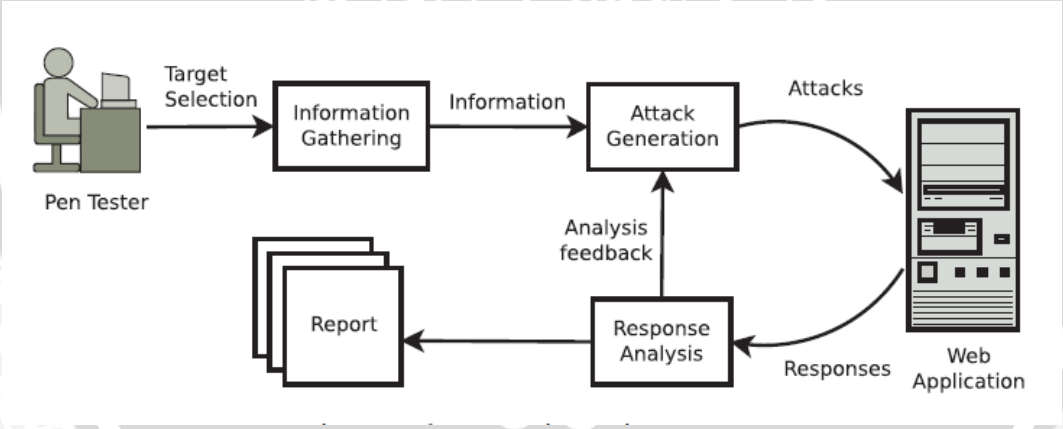
2.4.2.3 White Box Testing

Dengan memiliki sumber yang lengkap tentang konfigurasi jaringan dari jaringan pengujian dan konfigurasi sistem dari jaringan atau sistem

pengujian. Biasanya pengujian ini dilakukan dari jaringan internal. Pengujian kotak putih memerlukukan pemahaman mendalam tentang jaringan atau sistem pengujian dan memberikan hasil yang baik. (Goel, Jai Narayan. 2015).

2.4.3 Proses *Penetration Testing*

Proses uji penetrasi secara luas dapat dibagi menjadi tiga tahap: pengumpulan informasi, generasi serangan, dan analisis respon. Gambar 2.1 menunjukkan gambaran tingkat tinggi dari proses pengujian penetrasi generik. Pada tahap pengumpulan informasi, penguju menggunakan berbagai teknik, seperti pemindaian otomatis, *web crawler*, dan rekayasa sosial, untuk memperoleh informasi tentang aplikasi target. Informasi ini digunakan untuk menggerakan fase generasi serangan, dimana penguji menggunakan informasi yang diidentifikasi, bersama-sama dengan pengetahuan domain tentang kemungkinan kerentanan, untuk menghasilkan serangan. Penguji penetrasi biasanya menggunakan berbagai alat komersial dan open-source untuk mengotomatisasi generasi serangan. Terakhir, fase analisis respon memeriksa apakah serangan telah berhasil dan, jika demikian, informasi tentang serangan itu. Hasil akhir dari proses pengujian penetrasi adalah laporan yang merinci kerentanan yang ditemukan dan serangan yang sesuai. Developer dapat menggunakan informasi ini untuk menghilangkan kerentanan dan meningkatkan keamanan perangkat lunak mereka (Mukhopadhyay,Indraneel., 2014) .



Gambar 2.1 Proses Uji Penetrasi
(Sumber : William G. J. Halfond, 2011)

2.4.4 Kerentanan

2.4.4.1 Cross-Site Scripting (XSS)

Cross-site scripting sering disingkat sebagai *XSS*. Singkatnya hal itu terjadi ketika seorang penyerang dapat memasukkan kode HTML (seperti *javascript*), yang kemudian akan dilaksanakan untuk pengunjung situs. Sebuah contoh adalah buku tamu yang menunjukkan teks yang dimasukkan dalam buku tamu di *web*site. Jika penyerang memasuki *string script> alert ('XSS');</ script>* pop-up dengan

dapat dieksploitasi dengan cara yang lebih serius. Sebuah serangan mungkin menggunakan *XSS* untuk mencuri cookie pengguna, yang kemudian dapat digunakan untuk meniru pengguna pada situs *web*(Loo,Frank van der. 2011).

2.4.4.2 SQL Injections

SQL Injection dapat terjadi ketika masukan pengguna yang tidak divalidasi digunakan untuk membangun sebuah query SQL yang kemudian dieksekusi oleh *web server*. Sebuah contoh yang sangat terkenal adalah quer yang digunakan oleh user login. Query ini biasanya seperti *"SELECT*FROM users WHERE username = '* masukkan *username'*dan *password = 'x 'OR' 1 '=' 1 ' ".* Karena *'1'* selalu sama dengan *'1'*, query ini berlaku untuk semua catatan dalam database. Penjelasan lebih rinci dapat ditemukan di *web*site OWASP. *SQL Injection server* menunjukakan pesan kesalahan saat sintaks query SQL salah, untuk *SQL Injection* pesan kesalahan ini tidak ditampilkan. Sebaliknya penyerang aka melihat pesan kesalahan umum atau halaman (Loo,Frank van der. 2011)

2.4.4.3 Directory Traversal

Directory Traversal bertujuan untuk mengakses file dan direktori yang disimpan di luar *web* root folder. Dengan memanipulasi variabel yang mereferensi file dengan urutan " titik-titik garis miring (../)" dan variasinya atau dengan menggunakan jalur *absolute file*, mungkin untuk mengakses file dan direktori arbitary yang tersimpan pada sistem file termasukm kode atau konfigurasi sumber aplikasi dan file sistem kritis. Perlu dicatat bahwa akses ke file dibatasi oleh kontrol akses operasional sistem .

2.4.4.4 Authentication

Authentication adalah proses dimana sebuah entitas membuktikan identitas entitas lain, biasanya melalui pemberitahuan, seperti nama pengguna dan password. Tergantung pada kebutuhan pengguna, terdapat beberapa mekanisme otentikasi yang tersedia untuk dipilih. Jika tidak di pilih dan dilaksanakan secara benar, mekanisme otentikasi dapat mengekspos kerentanan yang penyerang dapat manfaatkan untuk mendapatkan akses ke sistem pengguna.

2.4.4.5 Captcha

Captcha adalah sebuah program yang dapat menghasilkan dan menilai pengujian dimana: bisa dilewati oleh kebanyakan orang, tapi program komputer saat ini tidak bisa melewatinya. Program tersebut dapat digunakan untuk membedakan manusia dari komputer dan memiliki banyak aplikasi untuk keamanan praktis(Ahn, Luis von.2000)

2.4.4.6 Authorization

Authorization adalah layanan kemanan utama yang menyangkut banyaknya perangkat lunak, dengan sebagian besar layanan keamanan lain yang menyangkut banyaknya perangkat lunak, dengan sebagian besar layanan keamanan lain yang

mendukungnya. Misalnya, keputusan kontrol akses umumnya ditegakkan atas dasar kebijakan pengguna tertentu, dan otentikasi adalah cara membangun pengguna yang bersangkutan. Demikian pula, kerahasiaan benar-benar sebuah manifestasi dari kontrol akses, khususnya kemampuan untuk membaca data. karena, dalam keamanan komputer, kerahasiaan sering identik dengan enkripsi, itu menjadi untuk menegakkan sebuah kebijakan akses-kontrol. Kebijakan yang akan diberlakukan oleh mekanisme akses kontrol umumnya beroperasi pada set sumber daya tersebut (kemampuan). Misalnya, kemampuan umum untuk file pada sistem file adalah: membaca, menulisn mengeksekusi, membuat, dan menghapus. Namun, ada operasi lain yang dipertimbangkan "meta-operasi" yang sering diabaikan – terutama membaca dan menulis atribut file, pengaturan kepemilikan file, dan menetapkan kebijakan kontrol akses ke salah satu operasi ini.

2.4.4.7 Mongo DB Injection

MongoDB merupakan *open-source NoSQL-Database* yang dikembangkan oleh 10gen di C ++. NoSQL adalah sebuah tren baru dalam pengembangan database dan mengacu secara umum pada database tanpa skema tetap. Database tersebut biasanya memiliki keamanan transaksi yang lebih rendah tetapi lebih cepat dalam mengakses data dan skalanya lebih baik dari database relasional. Untuk informasi lebih lanjut tentang NoSQL database disarankan membaca buku *NoSQL* yang ditulis oleh S. Edlich et. Al (Rapperswill, 2012).

2.4.4.8 File Include

File Include jarak jauh sama dengan inklusi file lokal, kecuali bahwa file yang disertakan adalah file dari *server* yang berbeda dari yang dijalankan aplikasi *web*. Contoh dari kerentanan ini adalah seperti untuk inklusi file lokal. Namun, selain mengubah parameter nama file ke file lokal, penyerang harus memasukkan *path* ke sebuah file jarak jauh.

2.4.4.9 Code Injection

Code Injection adalah istilah umum untuk jenis serangan yang terdiri dari kode injeksi yang kemudian diinterpretasikan/dieksekusi oleh applikasi. Jenis serangan ini mengeksploitasi buruknya penanganan data yang tidak dipercaya. Jenis-jenis serangan biasanya dimungkinkan karena kurang tepatnya validasi data input / output , misalnya : karakter yang diizinkan (kelompok ekspresi reguler standar atau kode khusus), format data, jumlah data yang diharapkan.

Code Injection berbeda dari *Command Injection* bahwa seorang penyerang hanya dibatasi oleh fungsi bahasa yang diinjeksi itu sendiri. Jika seorang penyerang mampu menginjeksi kode PHP ke dalam aplikasi dan melaksanakannya, ia hanya dibatasi oleh apa yang mampu dilakukan PHP. Perintah injeksi terdiri dari memanfaatkan kode yang ada untuk menjalankan perintah, biasanya dalam konteks *shell*.

2.4.4.10 Command Injection

Command Injection adalah peyerang yang dapat mengeksekusi perintah di *server*. Sebuah contohnya adalah aplikasi *web* yang memungkinkan pengguna memasukkan alamat IP dimana *server* akan mengirimkan ping. Jika penyerang akan memasuki string 1.2.3.4;ls *server* akan mengirim ping ke alamat IP 1.2.3.4 dan menjalankan perintah "ls". Penjelasan lebih rinci dapat ditemukan di situs *web* OWASP . Kerentanan ini dapat di uji dengan alat uji penetrasi dengan memasukkan koma diikuti oleh perintah (misalnya "ls") menjadi masukan halaman yang mungkin rentan dan memeriksa jika respon dari aplikasi *web* berisi output dari perintah yang di masukkan (Loo, Frank Van Der.2011).

2.5 Penjelasan *Penetration Testing Tool*

2.5.1 W3af

W3af adalah singkatan dari *Web Application Attack and Audit Framework* (Loo, Frank Van Der.2011).Ini adalah sebuah program *open-source*, ditulis dengan *pyhton*. Dengan menggunakan plugin untuk melakukan serangan pada aplikasi *web*. Keterangan tentang kerentanan plugin ini diklaim untuk medeteksi yang dapat ditemukan di *web*site alat ini. *W3af* juga menggunakan struktur berbasis teks yang digerakan oleh menu, tetapi juga memiliki GUI. Hasil dikeluarkan pada konsol atau ke XML-, text-, atau HTML-file.

2.5.1.1 Kelebihan dan Kekurangan *W3af*

Kelebihan dari *W3af* sendiri adalah :

- Mudah digunakan.
- Merupakan identifikasi aplikasi *web* dengan menggunakan 130 *plug-in*.
- Menampilkan hasil scan yang lebih teknis.
- Pengguna memiliki pilihan diantara *user interface* dan sebuah *command-line interface*.

  Kekurangan dari *W3af* sendiri adalah :
- ada Bug pada *W3af*

2.5.2 Wapiti

Wapiti adalah *open-source* lain yang ditulis dalam *pyhton* (Loo, Frank Van Der.2011).Kerentanan yang diklaimnya dapat mendeteksi hal-hal yang dapat ditemukan di *web*site alat ini. Ini bekerja dari baris perintah sepenuhnya secara otomatis, namun opsi baris perintah dapat digunakan untuk menyesuaikan pemindaian. Output ditulis ke konsol atau XML-, text-, atau HTML-file.

2.5.2.1 Kelebihan dan Kekurangan *Wapiti*

Kelebihan dari *Wapiti* sendiri adalah :

- *Wapiti* mendukung GET dan POST *HTTP* dengan metode untuk serangan.

- Menampilkan peringatan saat anomali ditemukan (misalnya 500 kesalahan dan timeout).
- Cepat dan mudah untuk mengaktifkan / menonaktifkan modul serangan.
- Waktu *scanning* dapat sekalian langsung di *attack*/serang.

Kekurangan dari *Wapiti* sendiri adalah :

- Memakan waktu yang lama ketika *scanning*

2.5.3 Arachni

Arachni adalah kerangka ruby dengan fitur yang lengkap, modular dan dengan kinerja yang tinggi, ditujukan untuk membantu penguji penetrasi dan administrator mengevaluasi keamanan aplikasi *web*. *Arachni* cerdas, karena hal tersbut melatih diri dengan belajar dari tanggapan *HTTP*yang diterima selama proses audit. Tidak seperti pemindai lainnya, *Arachni* memperhitungkan sifat dinamis dari aplikasi *web* dan dapat mendeteksi perubahan yang disebabkan saat bepergian melaluli jalur kompleksitas *cyclomatic* aplikasi *web*. Dengan cara ini vektor serangan / input yang tidak akan terdeteksu oleh non-manusia secara mulus ditangani oleh *Arachni*. Terakhir, *Arachni* menghasilkasn performa yang hebat karena model *HTTP*asinkron (*courtsey of Typhoeus*). Dengan demikian, hanya akan dibatasi oleh respon dari *server* di bawah audit dan *bandwith* anda yang tesedia.

2.5.3.1 Kelebihan dan Kekurangan *Arachni*

Kelebihan dari *Arachni* sendiri adalah :

- Menampilkan hasil *scanning* dengan banyak pilihan.
- Permintaan *HTTP asyncronous* untuk komunikasi yang cepat.
- Mendukung *scan multi-instance* untuk audit yang super cepat.
- Melakukan analisis meta data pada respon *HTTP*yang diterima.

Kekurangan dari *Arachni* sendiri adalah :

- Memakan waktu yang lama ketika *scanning*

2.6 Ubuntu

Ubuntu adalah sebuah kata Afrika kuno yang berarti 'kemanusiaan kepada orang lain'. Ini juga berarti 'saya adalah saya karena siapa kita semua'. Sistem operasi *Ubuntu* membawa semangat *Ubuntu* ke dunia komputer [https://www.*ubuntu*.com/about/about-*ubuntu*].

Ubuntu terdiri dari banyak paket perangkat lunak, yang sebagian besar didistribusikan di bawah lisensi perangkat lunak bebas (juga dikenal sebagai *open source*). Lisensi utama yang digunakan adalah GNU General Public License (GNU GPL) yang, bersama dengan GNU *Lesser General Public License* (GNU LGPL), secara eksplisit menyatakan bahwa pengguna bebas untuk menjalankan, menyalin, mendistribusikan, mempelajari, mengubah, mengembangkan dan meningkatkan

perangkat lunak. *Ubuntu* disponsori oleh perusahaan yang berbasis di Inggris Canonical Ltd, yang dimiliki oleh pengusaha Afrika Selatan Mark Shuttleworth.

Ubuntu berfokus pada kegunaan.Ubiquity installer mengizinkan *Ubuntu* untuk diinstal pada hard disk dari dalam lingkungan Live CD, tanpa perlu restart komputer sebelum instalasi. *Ubuntu* juga menekankan aksesibilitas dan internasionalisasi untuk menjangkau orang sebanyak mungkin. Dimulai dengan 5,04, UTF-8 yang menjadi standar penyandian karakter,yang memungkinkan untuk mendukung berbagai *script* non-Romawi. Sebagai fitur keamanan, alat sudo digunakan untuk menetapkan hak sementara untuk melakukan tugas administratif, sehingga akun root tetap terkunci, dan mencegah pengguna yang tidak berpengalaman dari secara tidak sengaja melakukan perubahan sistem katastropik atau pembukaan lubang keamanan.*PolicyKit* juga secara luas diimplementasikan ke dalam desktop untuk lebih mengeraskan sistem melalui prinsip yang paling sedikit keistimewaannya.

Versi Desktop *Ubuntu* sekaran mendukung x86 32 bit dan 64 bit. GPU yang didukung diperlukan untuk menjalankan efek visual seperti Unity Shell. Dalam kasus GPU yang tidak memadai, Unity dapat direduksi menjadi Unity 2D yang membutuhkan *hardware* yang lebih rendah.

Kebutuhan Minimal	*Server*	*Desktop*
Processor	300 Mhz	700 Mhz
Memory (RAM)	128 Mib	384 Mib
Hard Drive	1 GB	5 GB
Monitor Resolution	640x480	640x480

Tabel 2.1 Persyaratan Sistem pada *Ubuntu* Untuk Melakukan Uji Penetrasi

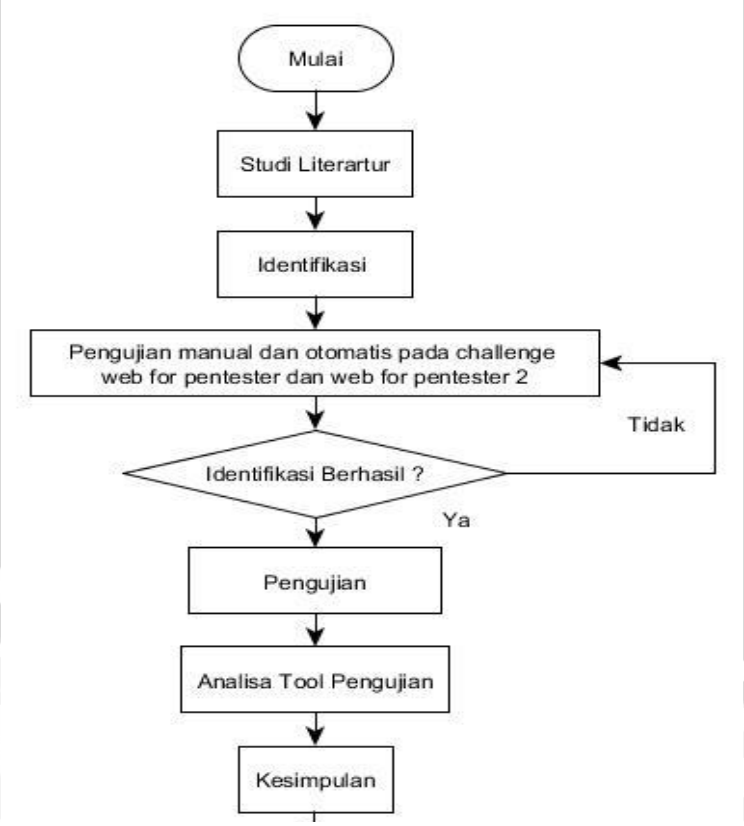
2.7 Independent Samples One Way ANOVA

Independent Samples One Way ANOVA merupakan pengujian yang digunakan untuk membandingkan means atau rata-rata dari dua kelompok independen untuk menentukan apakah ada bukti statistik yang menunjukkan rata-rata dari populasi tersebut memiliki perbedaan yang signifikan. Pengujian ini hanya dapat dilakukan lebih dari dua kelompok.

BAB 3
METODOLOGI

3.1 Jenis Penelitian

Dalam bab 3 ini , penelitian yang direncanakan merupakan jenis penelitian non-implementatif untuk uji *tool* penetrasi suatu sistem keamanan *web* apliikasi. Sesuai dengan tujuan penulisan ini, yaitu analisa perbandingan *penetration testing tool* untuk aplikasi *web* yang menguji *vulnerability* pada *web for pentester* dan *web for pentester 2* di *www.pentesterlab.com* dengan menggunakan *penetration testing tool W3af*,*Wapiti*, dan *Arachni*. Langkah-langkah yang digunakan dalam penelitian ini ditunjukkan pada Gambar 3.1



Gambar 3.1 *Flowchart* Utama Langkah Penelitian

3.2 Studi Literatur

Metodologi yang digunakan dalam analisis uji *tool* penetrasi untuk aplikasi *web* yang melalui beberapa langkah seperti identifikasi , pengujian manual dan pengujian otomatis pada *challenge* aplikasi *web www.pentesterlab.com*, dan hasil pengujian.

3.3 Lingkungan Penelitian

Lingkungan penelitian yang digunakan pada penelitian ini adalah aplikasi *web www.pentesterlab.com* yang terdapat pada layanan *web for pentester* dan *web for pentester 2*.

3.4 Metodologi Penelitian

Pada studi literatur ini dilakukannya hal ini bertujuan untuk mendapatkan hasil dari uji penetrasi dengan menggunakan *tool* yang digunakan untuk menguji kerentanan (*Vulnerability*) pada *web* yang akan di uji dengan *framework tool* yang digunakan. Mempelajari berbagai macam jurnal ilmiah di literatur studi ini , tentang literatur ini berupa memahami tentang konsep kinerja *penetration testing tool* terhadap aplikasi *web*.

3.4.1 Identifikasi *Web*

Tahap pertama dalam melakukan *vulnerability assessment* atau yang disebut juga dengan pengukuran kerentanan atau celah keamanan adalah melakukan identifikasi pada sistem yang digunakan aplikasi *web www.pentesterlab.com* yang akan diuji. Identifikasi dilakukan dengan cara mengidentifikasi *vulnerability* dari target aplikasi *web* yang akan diuji, dan kemudian dari identifikasi *vulnerability* yang digunakan dilakukan identifikasi kelemahan yang mungkin terdapat pada *vulnerability* yang digunakan oleh aplikasi *web* tersebut.

3.4.2 Analisis *Penetration Testing Tool*

Analisis pengujian *tool* uji penetrasi dilakukan dengan penentuan parameter-parameter pengujian yang akan diuji, penentuan parameter pengujian dilakukan berdasarkan *vulnerability* atau celah keamanan yang merupakan sebuah panduan atau acuan dalam melakukan pengujian keamanan terhadap aplikasi untuk menguji aspek keamanan yang menentukan kerentanan suatu aplikasi *web*. Setelah parameter-parameter pengujian telah selesai ditentukan hal selanjutnya yang dilakukan adalah memilih *penetration testing tool* yang sesuai dengan parameter yang akan diuji untuk agar membantu melakukan pengujian dalam aplikasi *web*, dalam penelitian ini pengujian *tool* penetrasi akan menggunakan *W3af, Arachni, Wapiti*.

3.5 Tabel Pengujian Otomatis

Tahap selanjutnya dalam pengujian penetrasi di gunakan table yang akan menggunakan table yang berguna untuk mengetahui bagaimana penilaian terhadap kerentanan yang diserang dan mengujinya dengan 3 aplikasi *web scanner*.

VULNERABILITY/APLLICATION WEB SCANNER	*W3AF*	*ARACHNI*	*WAPITI*
XSS			
SQL INJECTION			
DIRECTORY TRAVERSAL			
CODE INJECTION			
COMMAND INJECTION			
AUTHENTICATION			
AUTHORIZATION			
CAPTCHA			
MONGO DB			
FILE INCLUDE			

Tabel 3.1 Tabel Pengujian Otomatis

Tabel pengujian ini nanti digunakan untuk membandingkan *Tool* Uji Pentrasi mana yang paling banyak dan bisa mendeteksi kerentanan pada aplikasi *web www.pentesterlab.com* dan menjadi lebih mudah dalam mengetahui kerentanan tersebut.

3.6 Pengujian

Pada bagian pengujian ini menggunakan beberapa *Penetration testing tool* yang di jelaskan dari bab sebelumnya , pengujian dengan metode manual dan metode otomatis diharapkan celah keamana yang mungkin tidak terdeteksi pada metode manual dapat terdeteksi pada metode otomatis, dan celah keamanan yang tidak terdeteksi pada metode otomatis dapat terdeteksi pada metode manual. Perbandingan dari penilaian kerentanan yang didapat dan menjadi lebih efisien dalam melakukan *scanning* terhadap aplikasi *web www.pentesterlab.com*. Ada 3 *Penetration testing tool* yang diuji yaitu :

- *Wapiti*
- *W3af*
- *Arachni*

Dari masing-masing *penetration testing tool* tersebut dapat melakukan proses scanning terhadap aplikasi web yang dituju dengan menggunakan host dari

aplikasi web yang sudah disediakan. Pengujian tersebut melakukan mengidentifikasi dari kerentanan-kerentanan yang ada pada setiap challenge yang dipilih oleh penguji. Hasil pengujian dari ketiga tool ini nantinya akan di bandingkan dari segi banyaknya kerentanan yang di dapat dari hasil proses scanning.

3.7 Pengujian Manual

Pengujian secara manual merupakan pengujian yang dilakukan dengan penentuan parameter celah keamanan yang akan diuji terlebih dahulu. Penentuan parameter-parameter pengujian celah keamanan yang akan diuji dalam pengujian manual dilakukan berdasarkan *OWASP Testing Guide*, pengujian parameter, setelah itu akan dilakukan analisis. OWASP merupakan sebuah organisasi terbuka yang difokuskan pada peningkatan keamanan, mengembangkan, membeli, dan memelihara perangkat lunak. Semua perangkat atau *tool*, dokumen, forum, materi pada OWASP bebas dan terbuka untuk siapa saja yang tertarik dalam meningkatkan keamanan aplikasi. Tujuan dari OWASP adalah untuk membuat kemanan perangkat lunak terdeteksi sehingga individu atau suatu organisasi dapat membuat keputusan tentang resiko keamanan perangkat lunak yang benar. Berikut merupakan parameter-parameter yang akan diuji pada layanan aplikasi *web www.pentesterlab .com* :

- *SQL Injection*
- *XSS*
- *Directory Traversal*
- *Authentication*
- *Captcha*
- *Authorization*
- *Mongo DB* Injection
- *File Include*
- *Code Injection*
- *Command Injection*

Setelah Penentuan parameter telah selesai dilakukan hal yang selanjutnya dilakukan adalah mempersiapkan *software W3af,Wapiti,* dan *Arachni* yang digunakan untuk melakukan pengujian pada kerentanan-kerentanan pada aplikasi *web* yang di uji oleh softaware *web scanner*.

3.8 Pengujian Otomatis

Pengujian secara otomatis dilakukan dengan bantuan *software W3af , Wapiti , dan Arachni* . *Software W3af, Wapiti, dan arachi* merupakan *software* penguji keamanan aplikasi *web* secara otomatis yang melakukan audit pada aplikasi *web* dengan menguji kerentanan yang ada seperti *SQL Injection*, *Cross-Site Scripting*,

dan kerentanan lain yang dapat dieksploitasi secara umum. *W3af* , *Wapiti* , dan *Arachni* melakukan scan pada setiap halaman dan situs yang terdapa pada aplikasi *web* yang dapat diakses melalui *web browser* menggunakan protokol *HTTP*ataupun HTTPS.

Pengujian secara otomatis dilakukan setelah melakukan pengujian secara manual. Tujuan dari pengujian otomatis ini adalah untuk memeriksa ulang celah keamanan yang terdapat pada aplikasi *web* dalam hal ini www.pentesrlab.com yang terlewat saat pengujian secara manual yang telah dilakukan, setelah hasil *scanning* muncul maka akan dilakukan pengujian terhadap hasil *scanning* untuk memastikan hasil pengujian *vulnerability* dari hasil tersebut. Berikut ini merupakan langkah dalam proses pengujian secara otomatis pada *website www.pentesterlab.com* :

1. Hal pertama yang dilakukan terlebih dahulu adalah melakukan pesiapan *software* yang digunakan dengan melakukan install ke 3 *software W3af* , *Wapiti* , dan *Arachni*
2. Pengujian otomatis ini dengan menguji 2 aplikasi *web* dari *www.pentesterlab.com* yaitu *web for pentester* dan *web for pentester 2*. Dari 2 aplikasi *web* ini akan diuji lewat ke 3 aplikasi uji penetrasi.
3. setelah ke 3 *software* itu diinstall dan siap digunakan hal selanjutnya yang dilakukan adalah *scanning* aplikasi *web www.pentesterlab.com* untuk menenmukan celah keamanan yang ada

Untuk melakukan pengujian otomatis proses *scanning*, berikut skenario yang di lakukan pada *tool W3af* , *Wapiti* , dan *Arachni*

3.9 Skenario Percobaan

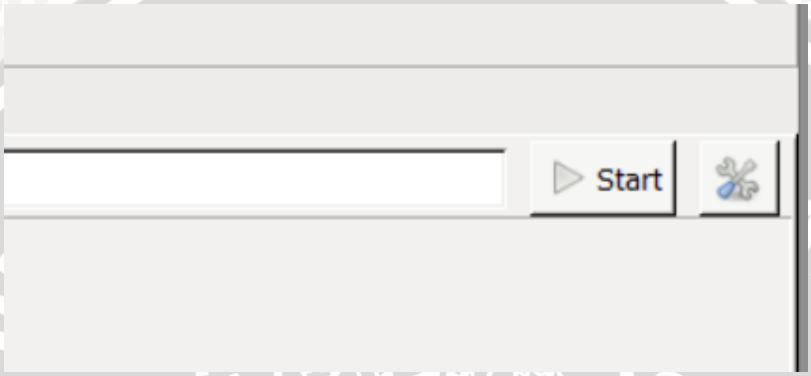
Pada percobaan ini, untuk mendapatkan data yang akan dianalisis digunakan skenario percobaan sebagai berikut:

1. Untuk pengumpulan informasi menggunakan teknik identifikasi manual dan otomatis agar memperoleh informasi tentang aplikasi web target yang dibutuhkan. Informasi ini digunakan untuk melakukan serangan terhadap kerentanan yang teridentifikasi pada aplikasi web target.
2. Identifikasi manual untuk melakukan uji penetrasi dengan menggunakan kode atau script pada web browser agar dapat mengetahui kerentanan yang di identifikasi pada aplikasi web, dan hasil dari identifikasi tersebut dapat mengetahui dari aplikasi web memiliki sebuah kerentanan.
3. Identifikasi otomatis menggunakan dengan tool w3af, wapiti, dan arachni. Identifikasi otomatis dengan tool aplikasi web yang digunakan untuk mengetahui banyaknya kerentanan aplikasi web yang dapat teridentifikasi dan dilakukan sebanyak 5 kali percobaan untuk mengetahui dampak tool uji kerentanan pada aplikasi web yang telah diuji

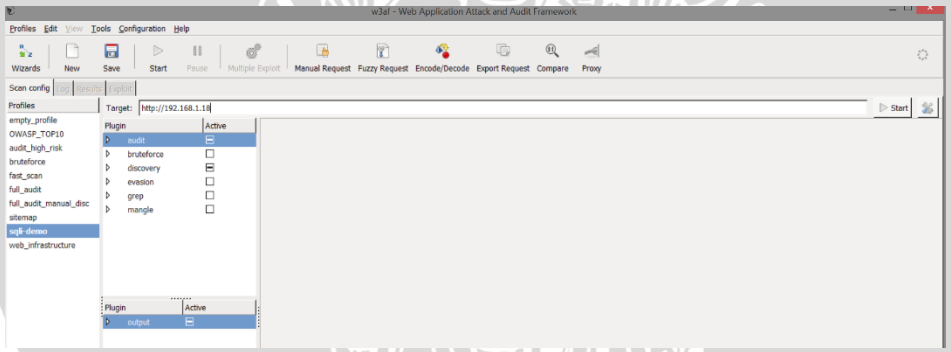
3.9.1 Skenario Pengujian Otomatis *W3af*

Berikut skenario pengujian otomatis pada *W3af* dalam uji penetrasi aplikasi *web www.pentesterlab.com* :

1. Buka Aplikasi *W3af* , dalam hal ini penguji menggunakan *W3af* berbasis GUI.
2. Setelah *W3af* muncul,masukin url target (kolom yang di sensor merah) dan ceklis mode serangan yang ingin anda lalukan. Di sini penguji menggunakan mode audit sebagai serangan
3. Setelah cheklist audit serangan,lanjutkan dengan klik 'start' dan tunggu proses *scanning*.



Gambar 3.2 Proses Pengujian pada *W3af*



Gambar 3.3 Proses Melakukan *Scanning* Pada *W3af*

4. Setelah proses *scanning* selesai, klik tab result untuk melihat hasil laporan *scanning*

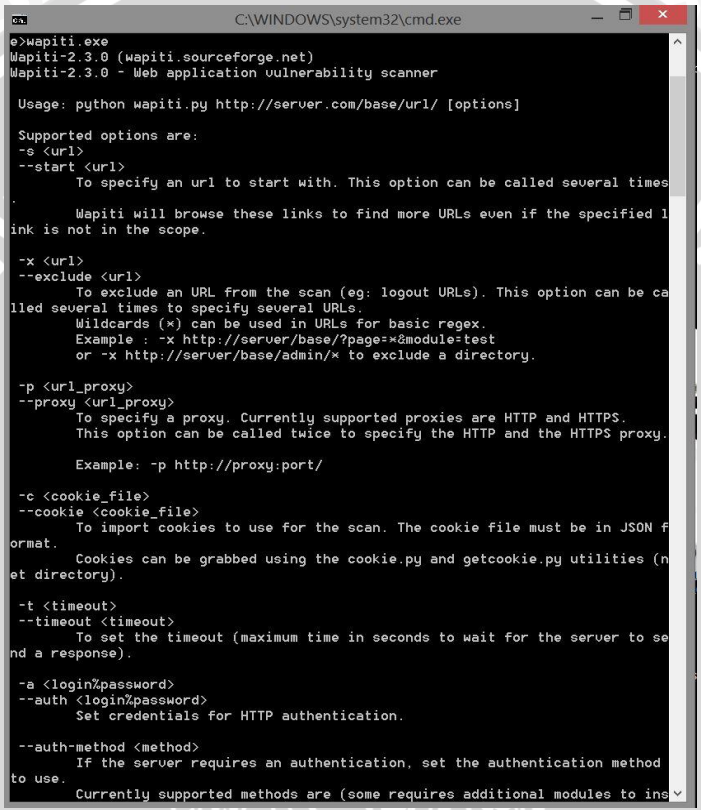
Pada penelitian ini, tool uji w3af digunakan untuk melakukan identifikasi pada aplikasi web yang dituju dengan mengetahui IP Host untuk melakukan identifikasi terhadap aplikasi web. Pengujian ini dilakukan agara mengetahui serangan-serangan terhadap aplikasi web yang dituju. Untuk melakukan pengujian ini pada tool w3af dengan membuat profile audit agar apa saja yang ingin diserang dengan tool w3af tersebut, setelah itu untuk memilih audit kerentanan apa saja yang ingin ditemukan pada tujuan aplikasi web tersebut. Identifikasi tersebut nantinya akan menemukan kerentanan setelah semuanya dapat ditemukan pada kerentanan yang telah di tuju pada aplikasi web. Pengujian dengan menggunakan tool w3af

bisa mengetahui hampir seluruh serangan yang di inginkan namun masih ada bug di dalam tool ini.

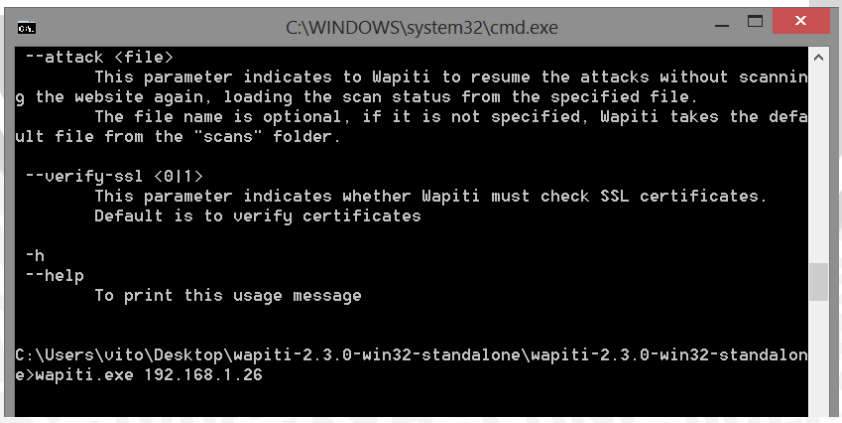
3.9.2 Skenario Pengujian Otomatis *Wapiti*

Pada langkah pengujian *Wapiti* untuk versi windows ini dijalankan melalui command prompt :

1. Untuk melakukan *scanning web for pentester* dengan *tool Wapiti* ini digunakan *command prompt* untuk melakukan identifikasi terhadap *web for pentester* dengan IP yang telah disediakan dari aplikasi web.



Gambar 3.4 Proses Pengujian Otomatis *Wapiti*



Gambar 3.5 Proses *Scanning* Pada *Wapiti*

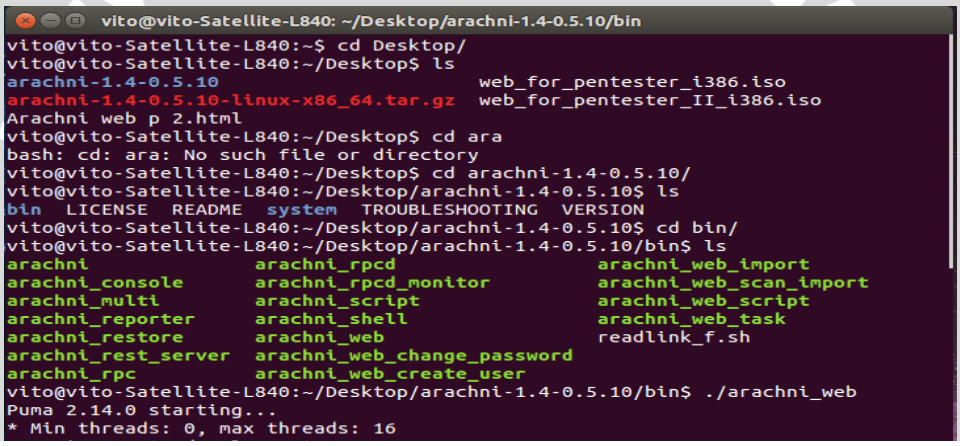
Beda dengan tool w3af, pengujian dengan tool wapiti ini menggunakan *command prompt* untuk melakukan identifikasi kerentanan pada aplikasi web.

Pengujian ini berbeda dengan tool w3af yang mana tool w3af dapat mengatur segala serangan ke aplikasi web yang diinginkan sedangkan wapiti hanya dapat melakukan identifikasi dengan fitur yang telah disediakan untuk melakukan identifikasi. Pengujian ini ditujukan untuk bisa menyerang dan mengidentifikasi untuk mengetahui kerentanan pada aplikasi web yang dituju. Dengan memasukkan IP host agar dapat mengetahui seleruh kerentanan yang didapat.

3.9.3 Skenario Pengujian Otomatis *Arachni*

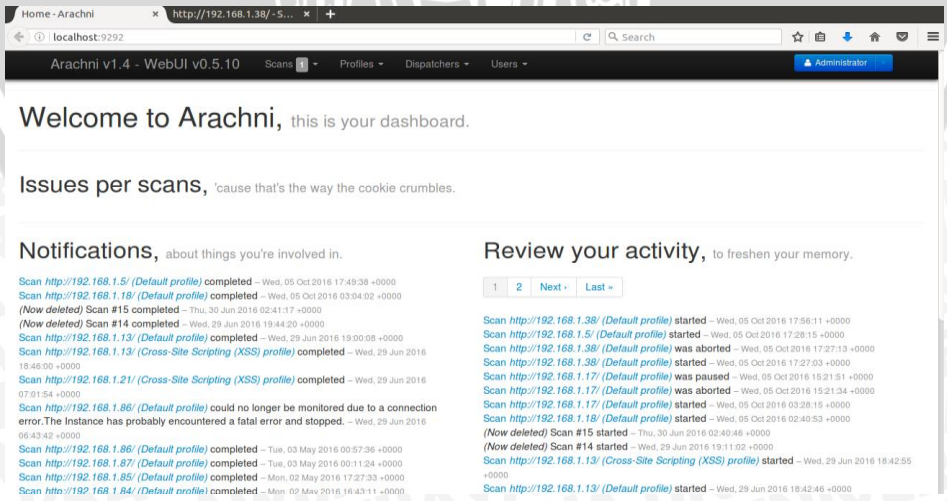
Pada pengujian *Arachni* ini dibutuhkannya sebuah operasi sistem *Ubuntu* dikarenakan *Arachni* hanya bisa di gunakan di *ubuntu*/*linux* untuk melakukan uji penetrasi. Untuk penjelasan tentang pengujian *Arachni* bisa dilihat pada gambar dibawah ini :

1. untuk melakukan pengujian *Arachni* ini gunakan terminal untuk membuka *Arachni_web* agar bisa mendapatkan localhostnya dari *Arachni scanner* setelah itu masukkan *./Arachni_web* untuk bisa masuk ke *web Arachni*:



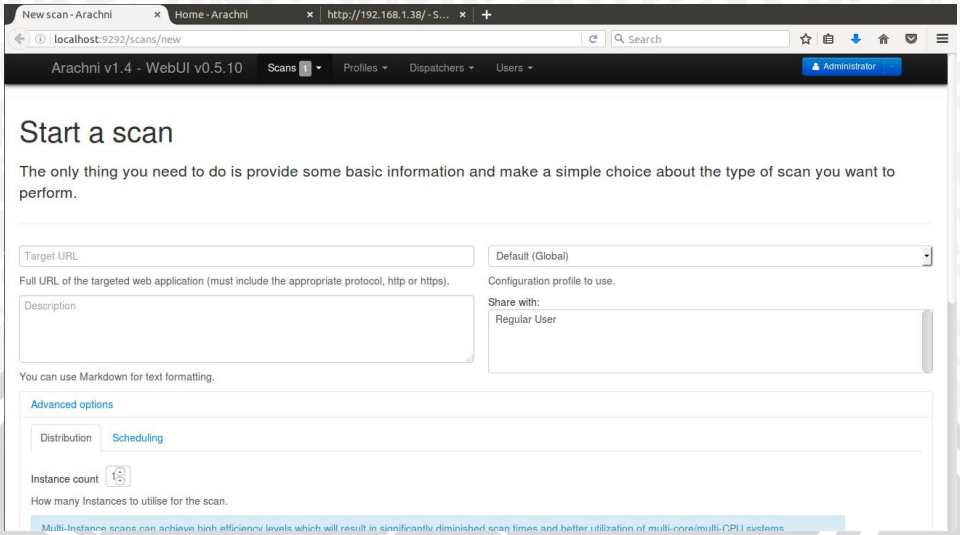
Gambar 3.6 Proses Pengujian Otomatis *Arachni*

2. Setelah semuanya berhasil bukalah browser untuk melakukan proses *scanning* dengan menggunakan localhost:9292 pada kolom browser :



Gambar 3.7 Proses *Scanning* Pada *Arachni*

3. Pilih scan untuk memulai pengujian *Arachni scanner* dan masukan ip addres dari *web* pentester *challenge* pada target URL untuk melakukan proses *scanning*, setelah sudah semua pilih GO untuk melakukan *scanning*



Gambar 3.8 Proses Memulai *Scanning* Pada *Arachni*

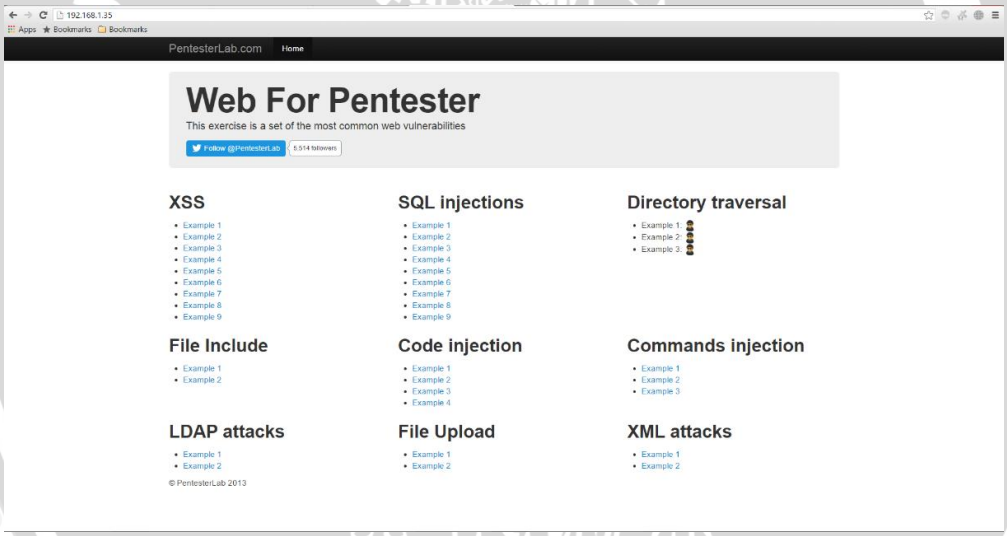
Penelitian dengan menggunakan tool arachni dengan menggunakan OS Ubuntu untuk melakukan identifikasi terhadap kerentanan aplikasi web dan memiliki sistem penjadwalan yang pada dasarnya memonitor segala kerentanan pada aplikasi web yang sedang berjalan. Dengan menggunakan IP untuk melakukan identifikasi kerentanan pada aplikasi web ini maka hasil dari identifikasi tersebut akan menampilkan dengan tampilan web agar terlihat lebih jelas.

BAB 4
HASIL

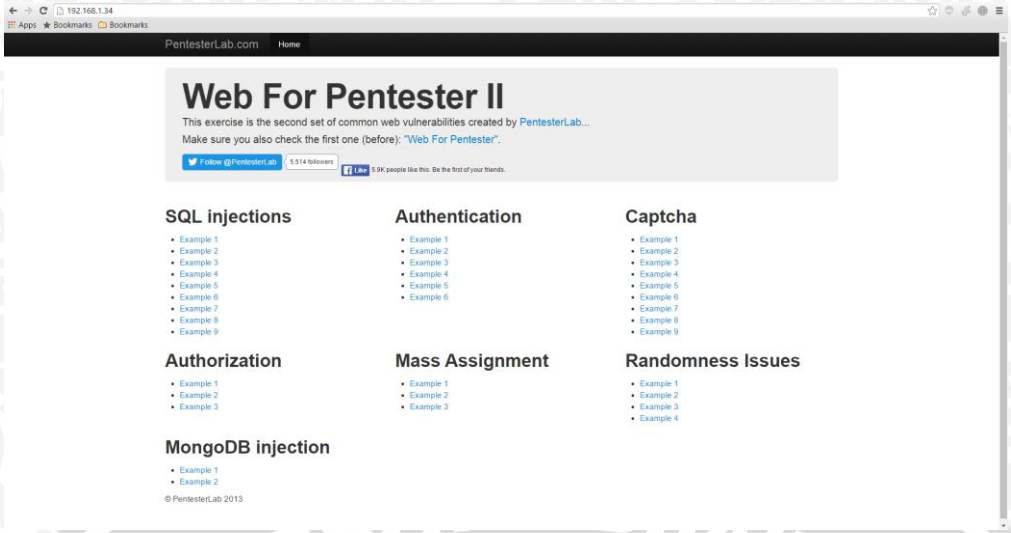
Pada bab ini akan menjelaskan tentang perbandingan alat uji penetrasi. Pertama setup tes akan dijelaskan dan bagian ini akan berakhir dengan hasil perbandingan dengan uji penetrasi *tool* yang telah dipilih. Perbandingan dari uji *tool* penetrasi ini diuji terhadap celah kerentanan pada aplikasi *web www.pentesterlab.com*.

4.1 Identifikasi

Uji *tool* penetrasi terdiri dari beberapa tes. Sebuah tes untuk melakukan penilaian perbandingan terhadap *penetration testing tool* yang dibuat oleh penulis. Aplikasi web *www.pentesterlab.com* yang di ambil sebagai uji coba dalam perbandingan *penetration testing tool*. Pada aplikasi *web www.pentesterlab.com* diuji dengan mengambil 2 *challenge* yang di pilih oleh penulis yaitu *web for pentester* dan *web for pentester 2* , di sini *penetration testing tool* di uji perbandingannya.



Gambar 4.1 *Web for pentester*



Gambar 4.2 *Web for pentester 2*

Hasil proses *scanning* dengan menggunakan *W3af*, *Arachni*, dan *Wapiti* adalah aplikasi *web* yang diuji dengan menggunakan apache Apache/2.2.16 (Debian) yang berjalan pada sistem operasi PHP/5.3.3-7+squeeze15.

4.2 Hasil Pengujian

Tahap selanjutnya dalam penilaian kerentanan atau *vulnerabilty* assesment adalah pengujian. Pada tahap ini terdapat dua metode yang digunakan yaitu metodel manual dan otomatis. Pada metode manual parameter-parameter kemanan yang telah ditentukan pada analisis kebutuhan akan diuji dengan melakukan serangan manual langsung ke aplikasi *web* tersebut. Sedangkan dalam metode otomatis dibutuhkan *penetration testing tool* untuk melakukan pengujian, pada pengujian ini *software W3af* , *Wapiti*, dan *Arachni* akan digunakan untuk membantu prose *scanning* untuk mendeteksi celah keamanan yang terdapat pada aplikasi *web*.

4.2.1 Hasil Pengujian Manual

Berdasarkan proses pengujian yang dilakukan secara manual dengan menentukan parameter-parameter celah keamanan pada layanan aplikasi *web* yang diuji hasil yang diperoleh adalah sebagian berikut :

4.2.1.1 XSS

Serangan *XSS* (*cross-site scripting)* biasanya digunakan untuk mencuri cookie, penyebaran malware, *session hijacking* / pembajakan session, dan pembelokkan tujuan / *malicious redirects*. Serangan ini tipikalnya adalah melakukan injeksi kode javascript terhadap sebuah *web*site sehingga browser mengeksekusi kode/script yang diperintahkan oleh penyerang. Kelemahan ini mudah didapat tapi susah untuk diatasi. Inilah alasannya mengapa *XSS* banyak ditemukan berbagai *web*site. Untuk melakukan pengujian manual ini dapat dilakukan dengan menyerang

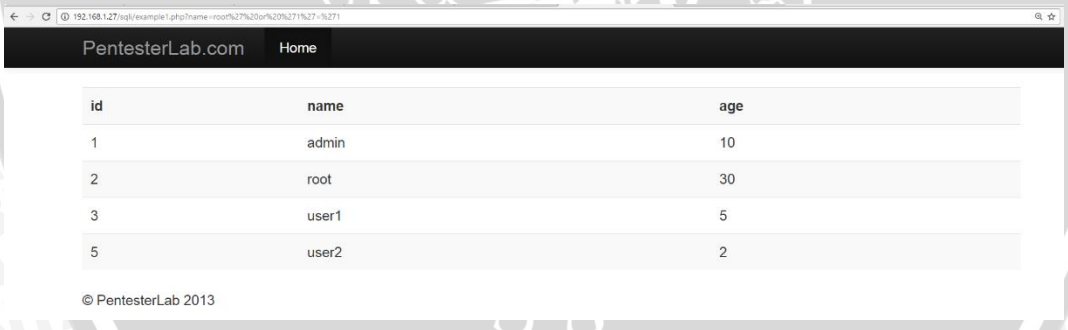
langsung pada aplikasi *web* host dari *web www.pentesterlab.com* , berikut adalah hasil dari pengujian manual :

Gambar 4.3 Hasil Pengujian Manual XSS

Pada pengujian manual pada *XSS* dapat diketahui bahwa dari Gambar 4.3 telah mendapatkan peringatan dari *challenge XSS* ini dimana penguji mengubah kode html pada browser tersebut dengan menambahkan *<script> alert(1)</script>* yang berfungsi untuk menampilkan peringatan pop up pada browser *XSS*.

4.2.1.2 SQL Injection

SQL Injection adalah teknik hacking untuk memanipulasi query (perintah database). Injeksi ini sering dilakukan inputan-inputan yang tidak di lakukan pengecekan kebenaran data yang di input. Inputan tersebut biasanya dimasukan pada box search atau bagian-bagian tertentu dari *web*site yang berinteraksi dengan database SQL dari situs tersebut. Perintah yang dimasukan para attacker biasanya adalah sebuah data yang mengandung link tertentu yang mengarahkan para korban ke *web*site khusus yang digunakan para attacker untuk mengambil data pribadi korban. Berikut adalah hasil pengujian manual *SQL Injection* :

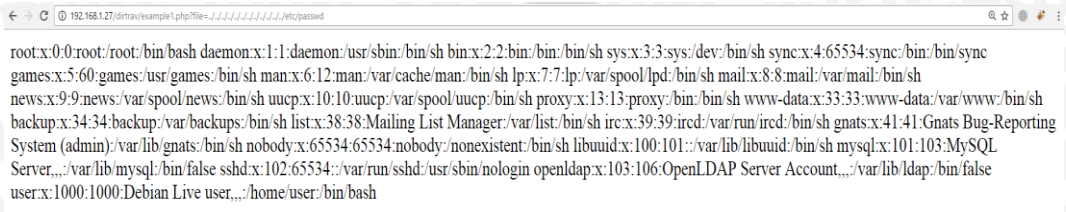


Gambar 4.4 Hasil Pengujian Manual SQL Injection

Pada hasil pengujian manual ini menunjukkan Gambar 4.4 *SQL Injection* tersebut telah merubah data tampilan sebelumnya yang berupa sebuah root saja. Dimana penguji telah memasukan beberapa perintah *SQL Injection* ke dalam box search.

4.2.1.3 *Directory Traversal*

Merupakan salah satu fungsi penting dari *Web server* yang aman adalah mengendalikan akses pada direktori yang terlarang. Mengeksploitasi *HTTP*untuk menghindari serangan keamanan *Web server* dan menggunakan perangkat lunak berbahaya untuk mengakses isi direktori yang dibatasi/disembunyikan. *Directory Traversal* adalah salah satu teknik untuk mengeksploitasi HTTP.

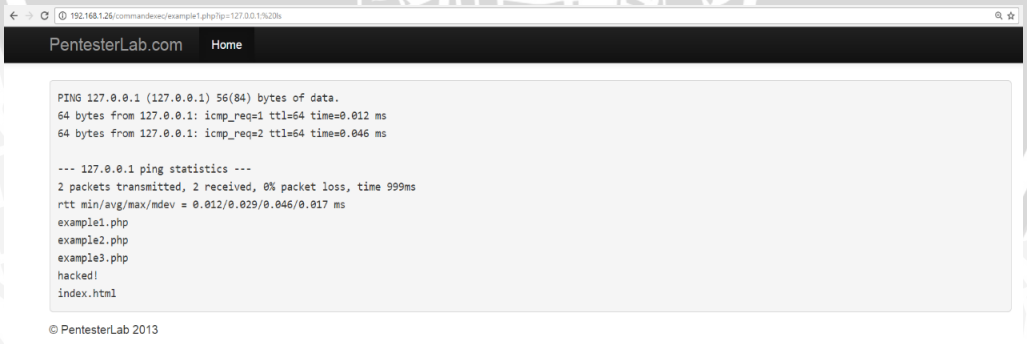


Gambar 4.5 Hasil Pengujian Manual Directory Traversal

Pada Gambar 4.5 Ketika kerentanan *Directory Traversal*,penyerang dapat mengakses direktori di atas direktori root. Setelah direktori ini diakses, perintah dapat dijalankan pada *server Web* dan data aman bisa disalin atau diubah. Keamanan dari *server Web* terancam.Kerentanan *Directory Traversal* juga dapat ditemukan dalam aplikasi *Web* yang dijalankan di *server Web*. Jika pengembang aplikasi gagal untuk menyediakan kode untuk memvalidasi masukan browser, hacker dapat melakukan percobaan dengan string input yang berbeda dan direktori akses diatas akar.

4.2.1.4 *Command Injection*

Merupakan sebuah serangan dengan tujuan menginjeksikan pada sistem operasi melalui aplikasi kerentanan. Serangan *Command Injection* yang mungkin ketika aplikasi user disediakan dengan data yang tidak aman untuk sistem shell.



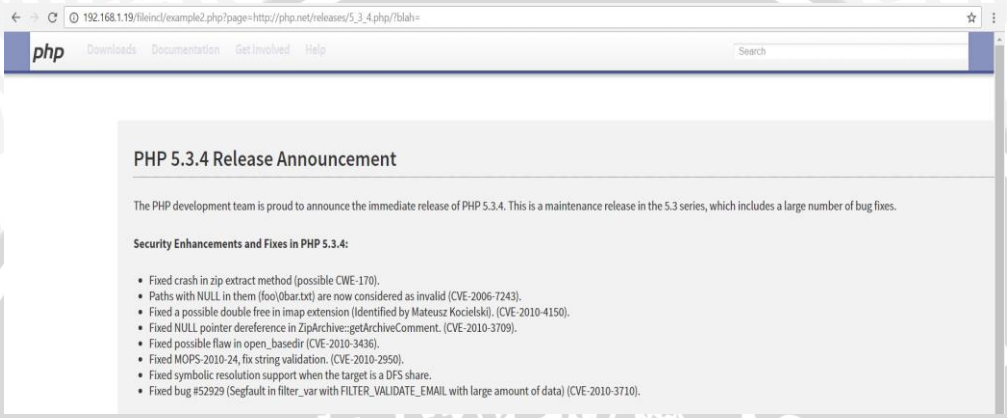
Gambar 4.6 Hasil Pengujian Manual Command Injection

Pada gambar 4.6 pengujian manual *Command Injection* ini merupakan penyerangan melalui *web* browser dengan mengijeksikan pada sistem operasi

melalui aplikasi kerentanan. Serangan yang dilakukan disediakannya dengan data yang tidak aman dari aplikasi *web* tersebut.

4.2.1.5 File Include

Pengujian kerentanan *File Include* datang dari kurangnya penyaringan ketika dikendalikan oleh user dan parameter yang digunakan sebagai sebagai bagian dari sebuah nama file di dalam memanggil fungsi. Jika panggilan kesalah satu metode ini rentan, penyerang akan dapat memanipulasi fungsi untuk mengisi kode sendiri. Pada Gambar 4.7 ini penguji mencoba kerentanan dengan memasukkan RFI (

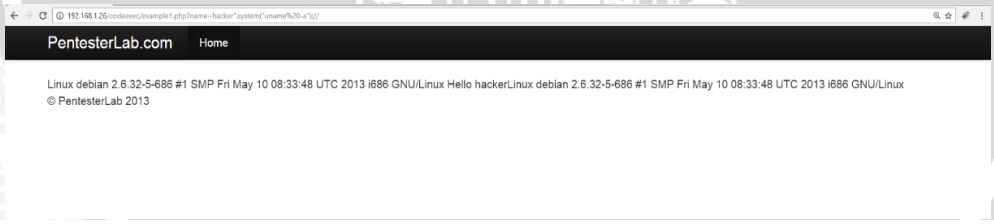


Gambar 4.7 Hasil Pengujian Manual File Include

Remote File Inclusion) ke URL pada aplikasi *web* dan menampilkan isi dari url tersebut, bahwa *web*site tersebut memiliki kerentanan pada bug RFI juga.

4.2.1.6 Code Injection

Code Injection merupakan serangan yang mana terdiri dari kode injeksi yang kemudian dieksekusi oleh aplikasi jenis serangan exploitasi ini mempunyai penanganan yang buruk dari data yang tidak dipercaya.



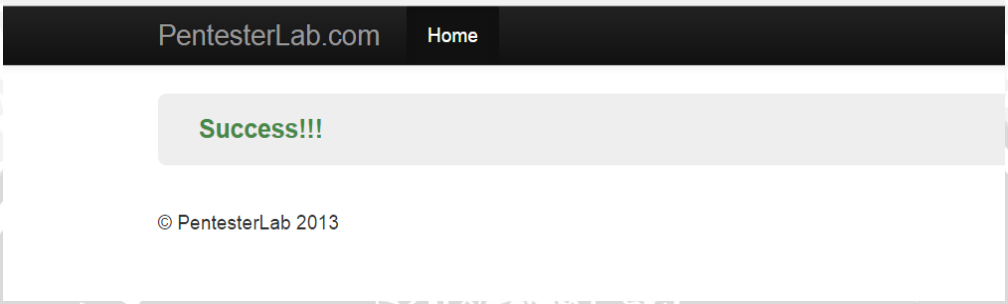
Gambar 4.8 Hasil Pengujian Manual *Code Injection*

Pada Gambar 4.8 pengujian manual *Code Injection* ini bahwa untuk melakukannya dengan payload umum, exploit yang digunakan untuk mengeksekusi shellcode.

Payload kemudan akan menjalankan shellcode yang dipilih untuk mendapatkan target komputer dan mendapatkan akses.

4.2.1.7 *Authentication*

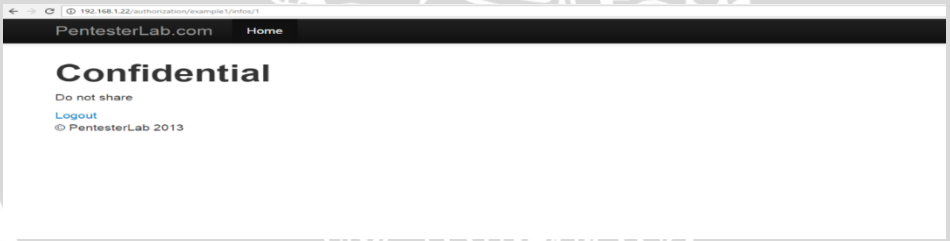
Merupakan suatu proses untuk melakukan validasi terhadap user credentials, yang ditujukan untuk menentukan apakah seorang user diperkenankan untuk mengakses jaringan atau computing resources. Bentuk *Authentication* yang paling sering di hadapi adalah saat diharuskan untuk memasukkan user name dan password. Kedua data tersebut kemudian dievaluasi untuk menentukan apakah anda adalah user yang sudah dikenal oleh sistem atau bukan-verikifasi identitas.



Gambar 4.9 Hasil Pengujian Manual *Authentication*

4.2.1.8 *Authorization*

Authorization adalah proses menentukan apa sajakah layanan yang bisa dinikmati pengguna yang telah jelas identitasnya (*authenticated user*). Jadi sebelum ada *Authorization*, harus melalui proses *Authentication*. Identitas yang

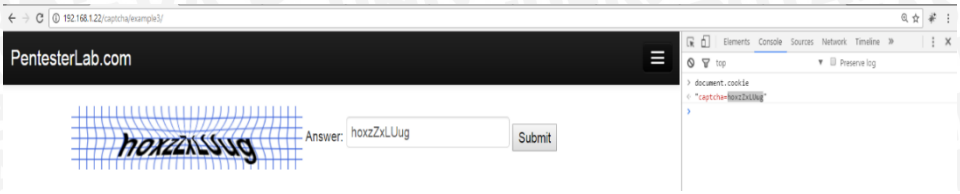


Gambar 4.10 Hasil Pengujian Manual *Authorization*

telah dibuktikan di proses *Authentication* menjadi dasar untuk menentukan layanan yang berhak dinikmati seorang pengguna.

4.2.1.9 *Captcha*

Merupakan sebuah protokol kriptograpfi, program yang dimaksudkan untuk membedakana manusia dari input mesin. suatu bentuk uji tantangan-tanggapan

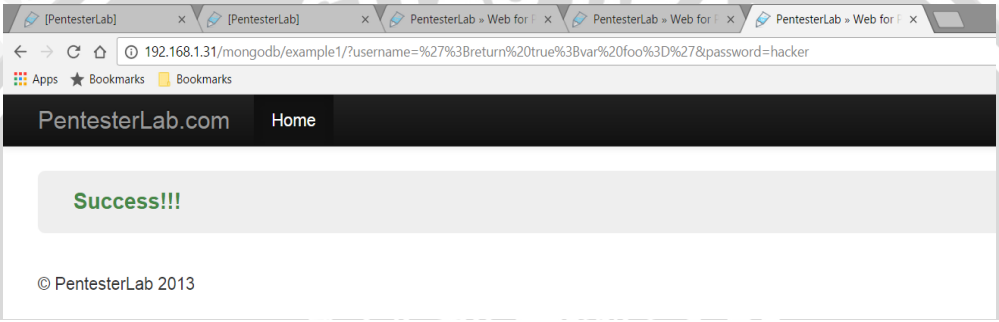


Gambar 4.11 Hasil Pengujian Manual *Captcha*

(challange-response test) yang digunakan dalam perkomputeran untuk memastikan bahwa jawaban tidak dihasilkan oleh suatu komputer. Proses ini biasanya melibatkan suatu komputer (*server*) yang meminta seorang pengguna untuk menyelesaikan suatu uji sederhana yang dapat dihasilkan dan dinilai oleh komputer tersebut. Karena komputer lain tidak dapat memecahkan *CAPTCHA*, pengguna manapun yang dapat memberikan jawaban yang benar akan dianggap sebagai manusia.

4.2.1.10 Mongo DB Injection

Merupakan sebuah salah satu produk database noSQL OPEN SOURCE yang menggunakan struktur JSON untuk menyimpan datanya. *MongoDB* sering digunakan untuk aplikasi berbasis Cloud, Grid Computing , atau Big Data, untuk mengetahui hasil dari pengujain manual dapat dilihat pada gambar 4.12



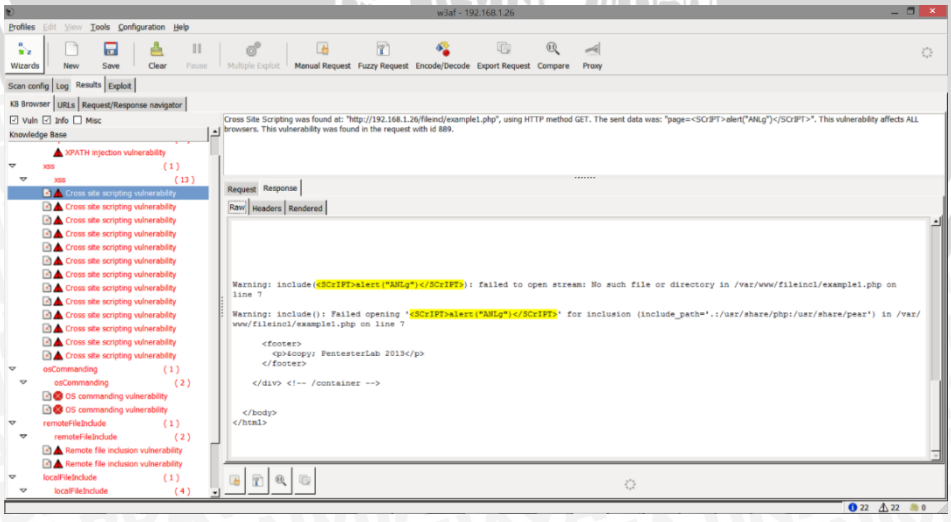
Gambar 4.12 Hasil Pengujian Manual MongoDB

4.3 Hasil Pengujian Otomatis

Berikut adalah hasil dari pengujian otomatis dari *W3af* , *Wapiti* ,*Arachni* :

4.3.1 Hasil Pengujian Otomatis *W3af*

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



Gambar 4.13 Hasil Pengujian Otomatis *W3af Web for pentester*

hasil dari gambar 4.13 *scanning web for pentester* di sini dengan menggunakan tool *W3af* telah mendeteksi beberapa jenis *vulnerability* tersebut :

- *XSS*

Vulnerability	tcp/80	Cross Site Scripting was found at: "http://192.168.1.26/xss/example1.php", using HTTP method GET. The sent data was: "name=<SCrIPT>alert("mVtV")</SCrIPT>". This vulnerability affects ALL browsers. This vulnerability was found in the request with id 931. **URL :** http://192.168.1.26/xss/example1.php Severity : Medium

Gambar 4.14 Hasil Kerentanan *XSS* dengan tool *W3af*

- *Command Injection*

Vulnerability	tcp/80	OS Commanding was found at: "http://192.168.1.26/commandexec/example1.php", using HTTP method GET. The sent data was: "ip=%7Cping+-c+9+localhost". This vulnerability was found in the request with id 2487. **URL :** http://192.168.1.26/commandexec/example1.php Severity : High

Gambar 4.15 Hasil Kerentanan *Command Injection* dengan tool *W3af*

- *File Include*

Vulnerability	tcp/80	Local File Inclusion was found at: "http://192.168.1.26/fileincl/example1.php", using HTTP method GET. The sent data was: "page=../../../../../../../../../../../../etc/passwd". This vulnerability was found in the request with id 4956. **URL :** http://192.168.1.26/fileincl/example1.php Severity : Medium

Gambar 4.16 Hasil Kerentanan *File Include* dengan tool *W3af*

- *Directory Traversal*

Vulnerability	tcp/80	Local File Inclusion was found at: "http://192.168.1.26/dirtrav/example1.php", using HTTP method GET. The sent data was: "file=../../../../../../../../../../../../etc/passwd". This vulnerabiliity was found in the request with id 5267. **URL :** http://192.168.1.26/dirtrav/example1.php Severity : Medium

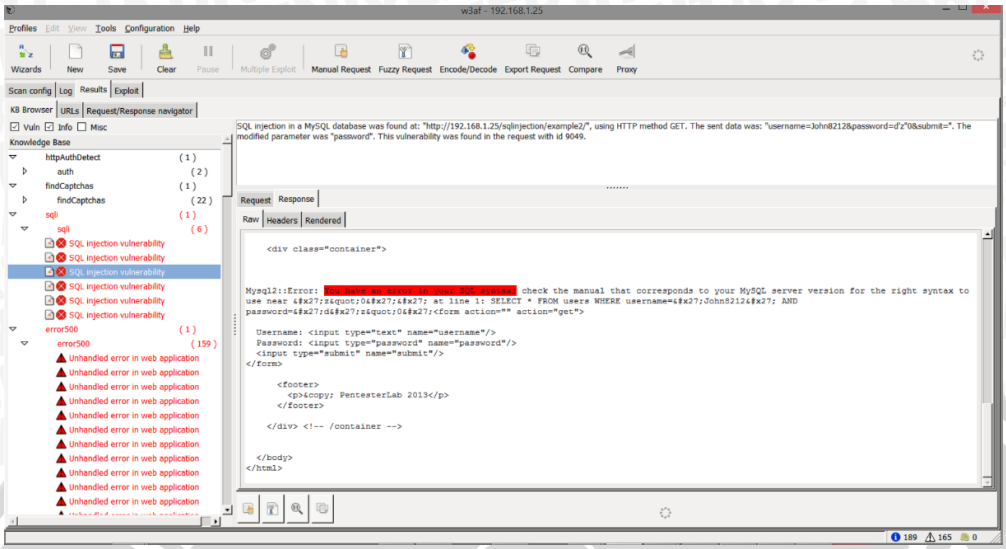
Gambar 4.17 Hasil Kerentanan *Directory Traversal* dengan tool *W3af*

- *Code Injection*

Information	tcp/80	Cross Site Scripting was found at: "http://192.168.1.26/codeexec/example1.php", using HTTP method GET. The sent data was: "name=<ScRIPT>a=/JS8T/%0Aalert(a.source)</SCRiPT>". This vulnerability affects ALL browsers. This vulnerability was found in the request with id 980. **URL :** http://192.168.1.26/codeexec/example1.php

Gambar 4.18 Hasil Kerentanan *Code Injection* dengan tool *W3af*

Setelah proses *scanning* pada *web for pentester 2* selesai maka akan muncul hasil *vulnerabilty* yang telah di*scanning* , berikut hasilnya :



Gambar 4.19 Hasil Pengujian Otomatis *W3af Web for pentester 2*

Hasil dari gambar 4.19 *scanning web for pentester* di sini dengan menggunakan tool *W3af* telah mendeteksi beberapa jenis *vulnerability* tersebut yaitu :

- *SQL Injection*

Vulnerability	tcp/80	SQL injection in a MySQL database was found at: "http://192.168.1.25/sqlinjection/example2/", using HTTP method GET. The sent data was: "username=John8212&password=d'z"0&submit=". The modified parameter was "password". This vulnerability was found in the request with id 9049. **URL :** http://192.168.1.25/sqlinjection/example2/ Severity : High

Gambar 4.20 Hasil Kerentanan *SQL Injection* dengan tool *W3af*

- *Authentication*

Vulnerability	tcp/80	An unidentified web application error (HTTP response code 500) was found at: "http://192.168.1.25/authentication/example6/submit". Enable all plugins and try again, if the vulnerability still is not identified, please verify mannually and report it to the w3af developers. This vulnerability was found in the request with id 1153. **URL :** http://192.168.1.25/authentication/example6/submit Severity : Medium

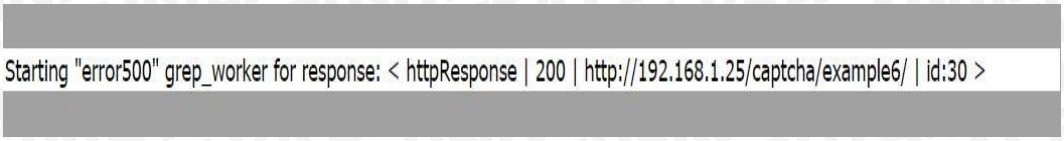
Gambar 4.21 Hasil Kerentanan *Authentication* dengan tool *W3af*

- Authorization

Starting "error500" grep_worker for response: < httpResponse | 200 | http://192.168.1.25/authorization/example3/ | id:252 >

Gambar 4.22 Hasil Kerentanan *Authorization* dengan tool *W3af*

- *Captcha*



Gambar 4.23 Hasil Kerentanan *Captcha* dengan tool *W3af*

- *Mongo DB Injection*

Gambar 4.24 Hasil Kerentanan *Mongo DB Injection* dengan tool *W3af*

4.3.1.1 Hasil Waktu Identifikasi *Tool W3af*

Dari identifikasi aplikasi web dengan menggunakan *tool W3af* penulis melakukan pencatatan waktu identifikasi terhadap aplikasi web yang diuji dengan *web for pentester* dan *web for pentester 2* pada gambar yang di bawah.

Gambar 4.25 Hasil Identifikasi Waktu Web For Pentester

Gambar 4.26 Hasil Identifikasi Waktu Web For Pentester 2

Berdasarkan hasil waktu Identifikasi dari tool w3af pada tabel berikut :

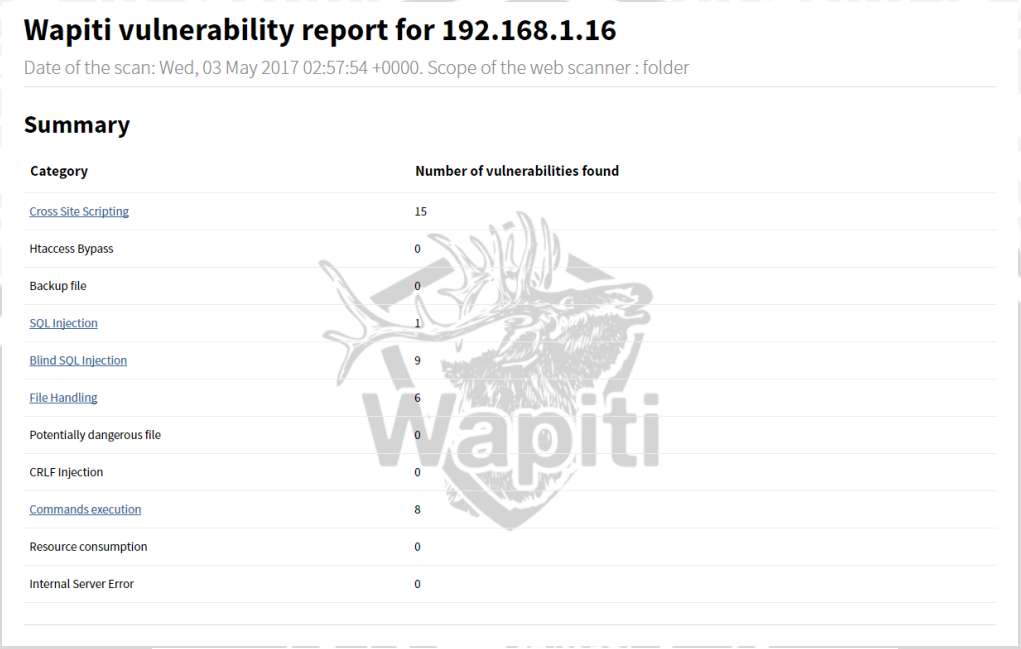
Aplikasi Web	Waktu
Web For Pentester	00.04.20
Web For Pentester 2	00.03.53

Tabel 4.1 Hasil Waktu Identifikasi *Tool W3af*

Tabel 4.0 menunjukkan hasil dari waktu identifikasi dengan tool w3af pada aplikasi web challenge web for pentester dengan waktu 00.04.20 sedangkan pada web for pentester 2 mendapatkan waktu identifikasi sekitar 00.03.53.

4.3.2 Hasil Pengujian Otomatis *Wapiti*

Hasil dari *scanning web for pentester* di sini dengan menggunakan *Wapiti* telah mendeteksi beberapa jenis *vulnerability* tersebut dibawah ini :



Wapiti vulnerability report for 192.168.1.16

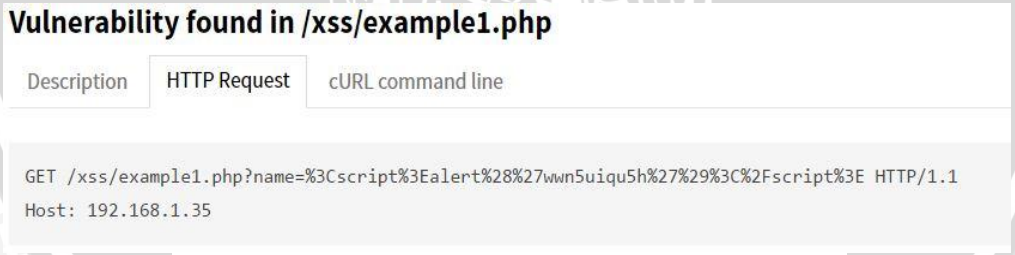
Date of the scan: Wed, 03 May 2017 02:57:54 +0000. Scope of the web scanner : folder

Summary

Category	Number of vulnerabilities found
Cross Site Scripting	15
Htaccess Bypass	0
Backup file	0
SQL Injection	1
Blind SQL Injection	9
File Handling	6
Potentially dangerous file	0
CRLF Injection	0
Commands execution	8
Resource consumption	0
Internal Server Error	0

Gambar 4.27 Hasil Pengujian Otomatis *Wapiti Web for pentester*

- *XSS*



Vulnerability found in /xss/example1.php

Description | HTTP Request | cURL command line

```
GET /xss/example1.php?name=%3Cscript%3Ealert%28%27wwn5uiqu5h%27%29%3C%2Fscript%3E HTTP/1.1
Host: 192.168.1.35
```

Gambar 4.28 Hasil Kerentanan *XSS* dengan tool *Wapiti*

- *SQL Injection*

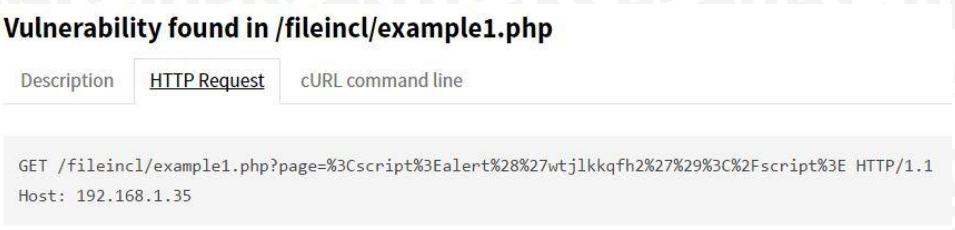
Vulnerability found in /sqli/example1.php

Description | HTTP Request | cURL command line

```
GET /sqli/example1.php?name=%27%20or%20sleep%287%29%231 HTTP/1.1
Host: 192.168.1.35
```

Gambar 4.29 Hasil Kerentanan *SQL Injection* dengan tool *Wapiti*

- *File Include*

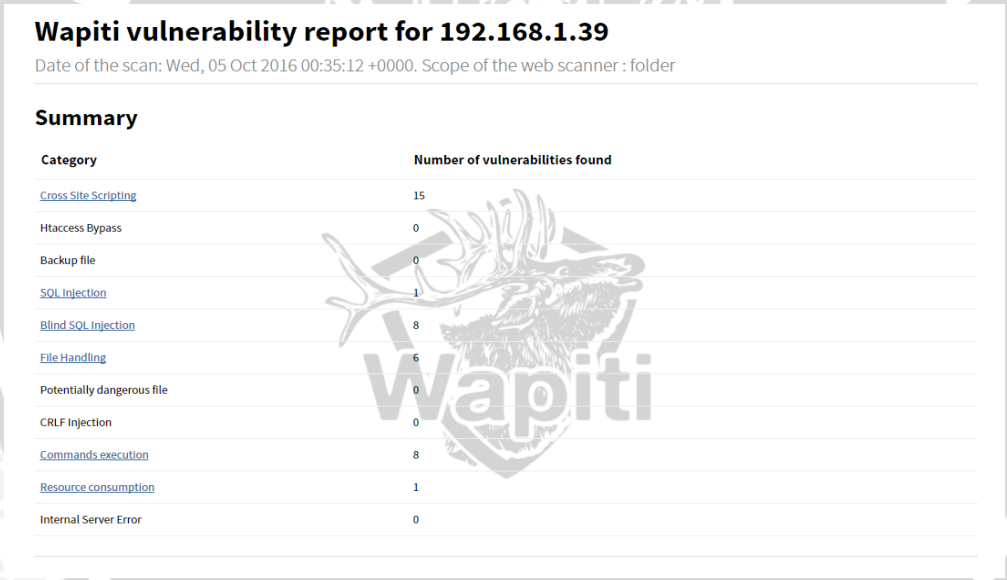


Gambar 4.30 Hasil Kerentanan *File Include* dengan tool *Wapiti*

- *Command Injection*

Gambar 4.31 Hasil Kerentanan *Command Injection* dengan tool *Wapiti*

Selanjutnya setelah mendapatkan hasil dari *web for pentester* adalah melakukan proses *scanning* pada *web*site *web for pentester 2* :



Gambar 4.32 Hasil Pengujian Otomatis *Wapiti Web for pentester 2*

Hasil dari *scanning web for pentester 2* di sini dengan menggunakan *Wapiti* telah mendeteksi beberapa jenis *vulnerability* tersebut yaitu :

- *Authentication*

Anomaly found in /authentication/example5/submit

Description | HTTP Request | cURL command line

```
The server responded with a 500 HTTP error code while attempting to inject a payload in the query string
```

```
GET /authentication/example5/submit?%BF%27%22%28 HTTP/1.1
Host: 192.168.1.34
```

```
curl "http://192.168.1.34/authentication/example5/submit?%BF%27%22%28"
```

Gambar 4.33 Hasil Kerentanan *Authentication* dengan tool *Wapiti*

- *Authorization*

Anomaly found in /authorization/example1/

Description | HTTP Request | cURL command line

```
The server responded with a 500 HTTP error code while attempting to inject a payload in the query string
```

```
GET /authorization/example1/?%BF%27%22%28 HTTP/1.1
Host: 192.168.1.34
```

```
curl "http://192.168.1.34/authorization/example1/?%BF%27%22%28"
```

Gambar 4.34 Hasil Kerentanan *Authorization* dengan tool *Wapiti*

- *Captcha*

Anomaly found in /captcha/example1/submit

Description | HTTP Request | cURL command line

```
The server responded with a 500 HTTP error code while attempting to inject a payload in the query string
```

```
GET /captcha/example1/submit?%BF%27%22%28 HTTP/1.1
Host: 192.168.1.34
```

```
curl "http://192.168.1.34/captcha/example1/submit?%BF%27%22%28"
```

Gambar 4.35 Hasil Kerentanan *Captcha* dengan tool *Wapiti*

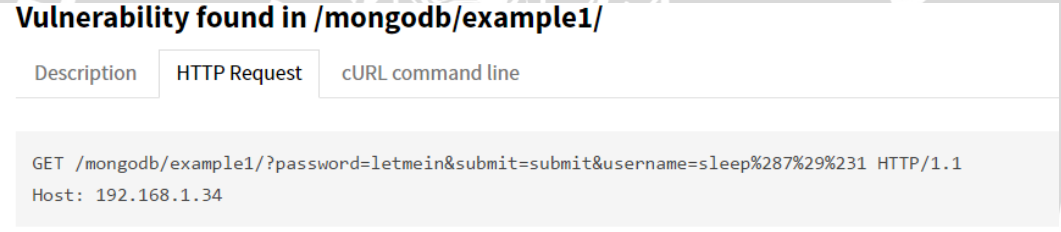
- *Directory Traversal*

Gambar 4.36 Hasil Kerentanan *Directory Traversal* dengan tool *Wapiti*

- *Code Injection*

Gambar 4.37 Hasil Kerentanan *Code Injection* dengan tool *Wapiti*

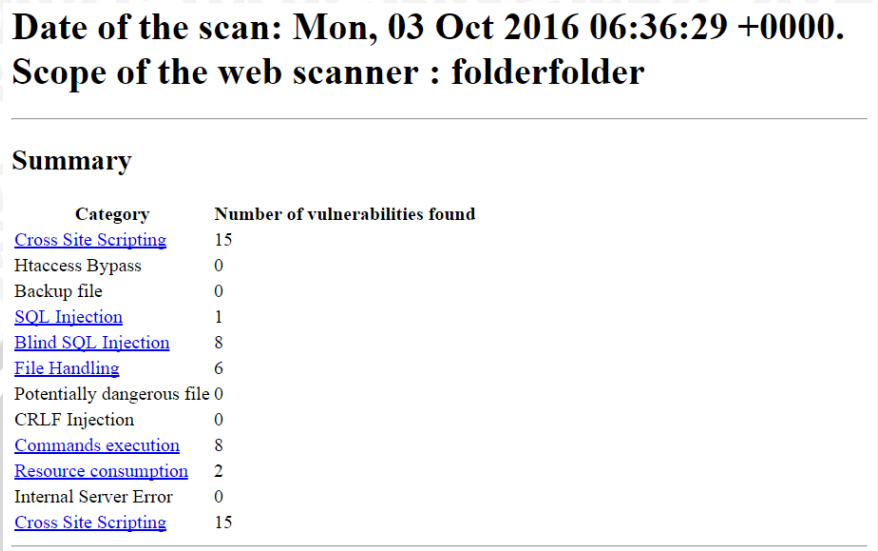
- *Mongo DB Injection*



Gambar 4.38 Hasil Kerentanan *Mongo DB Injection* dengan tool *Wapiti*

4.3.2.1 Hasil Waktu Identifikasi *Tool Wapiti*

Dari identifikasi aplikasi web dengan menggunakan *tool Wapiti* penulis melakukan pencatatan waktu identifikasi terhadap aplikasi *web* yang diuji dengan *web for pentester* dan *web for pentester 2* pada gambar yang di bawah.

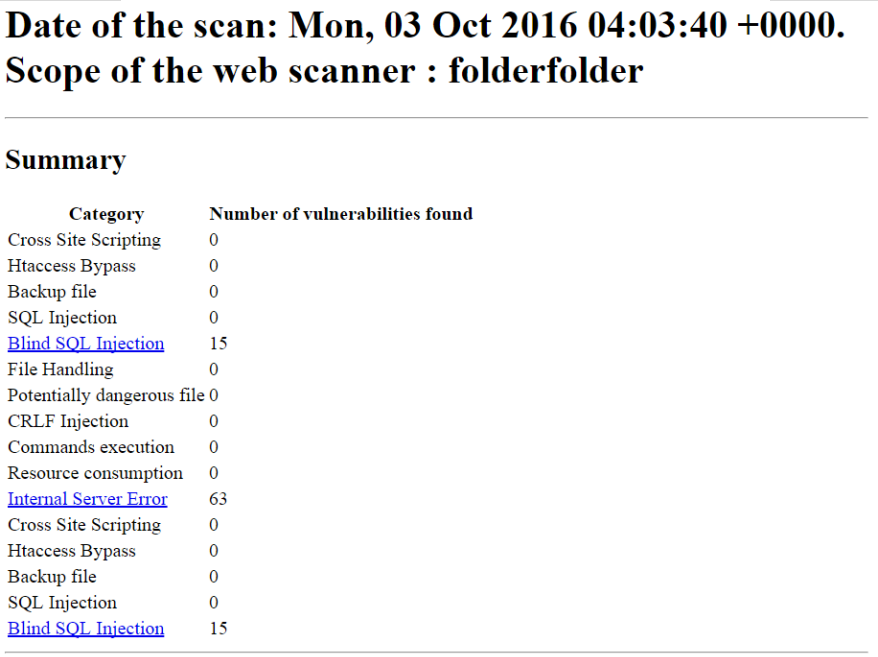


Date of the scan: Mon, 03 Oct 2016 06:36:29 +0000.
Scope of the web scanner : folderfolder

Summary

Category	Number of vulnerabilities found
Cross Site Scripting	15
Htaccess Bypass	0
Backup file	0
SQL Injection	1
Blind SQL Injection	8
File Handling	6
Potentially dangerous file	0
CRLF Injection	0
Commands execution	8
Resource consumption	2
Internal Server Error	0
Cross Site Scripting	15

Gambar 4.39 Hasil Identifikasi Waktu Web For Pentester



Date of the scan: Mon, 03 Oct 2016 04:03:40 +0000.
Scope of the web scanner : folderfolder

Summary

Category	Number of vulnerabilities found
Cross Site Scripting	0
Htaccess Bypass	0
Backup file	0
SQL Injection	0
Blind SQL Injection	15
File Handling	0
Potentially dangerous file	0
CRLF Injection	0
Commands execution	0
Resource consumption	0
Internal Server Error	63
Cross Site Scripting	0
Htaccess Bypass	0
Backup file	0
SQL Injection	0
Blind SQL Injection	15

Gambar 4.40 Hasil Identifikasi Waktu Web For Pentester

Berdasarkan hasil waktu Identifikasi dari *tool Wapiti* pada tabel berikut :

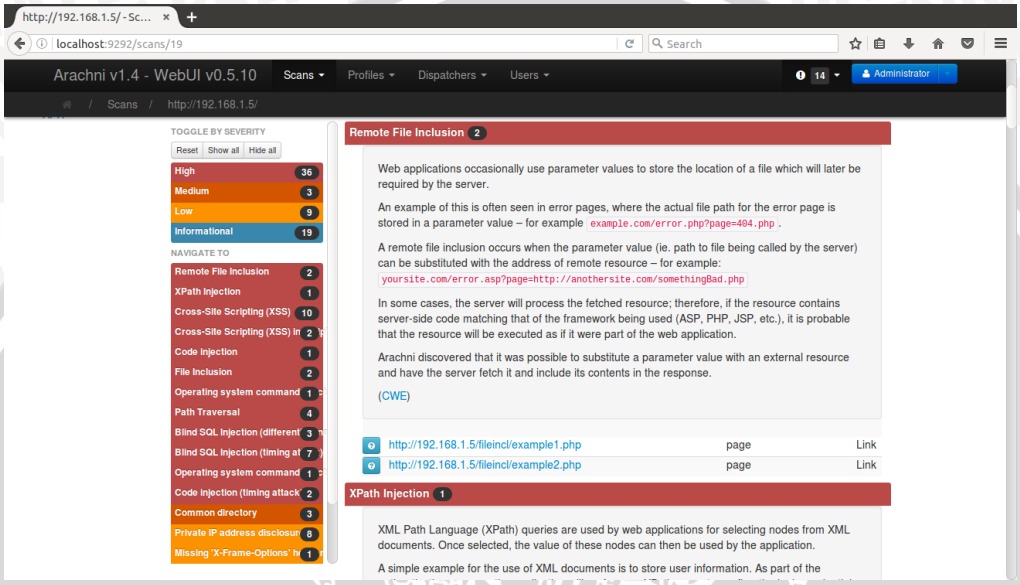
Aplikasi Web	Waktu
Web For Pentester	00.06.36
Web For Pentester 2	00.04.03

Tabel 4.2 Hasil Waktu Identifikasi *Tool Wapiti*

Tabel 4.2 menunjukkan hasil dari waktu identifikasi dengan *tool Wapiti* pada aplikasi *web challenge web for pentester* dengan waktu 00.06.36 sedangkan pada *web for pentester 2* mendapatkan waktu identifikasi sekitar 00.04.03.

4.3.3 Hasil Pengujian Otomatis *Arachni*

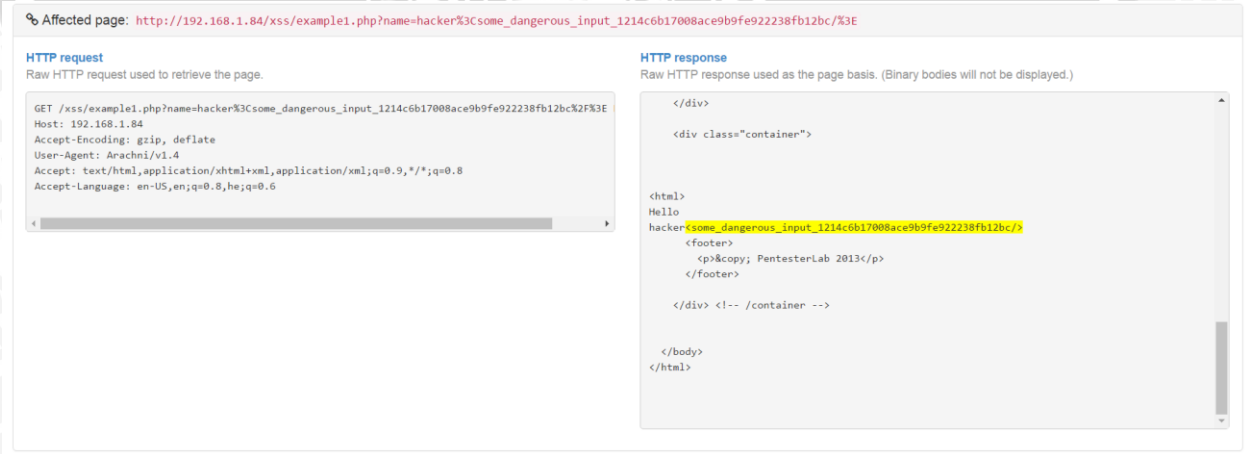
Setelah semua proses *scanning* berhasi dijalankan maka hasil dari identifikasi pada *Arachni scanner* bisa di liat dan di export melaluit HTML, XML, JSON, dan lain-lain :



Gambar 4.41 Hasil Pengujian Otomatis *Arachni Web for pentester*

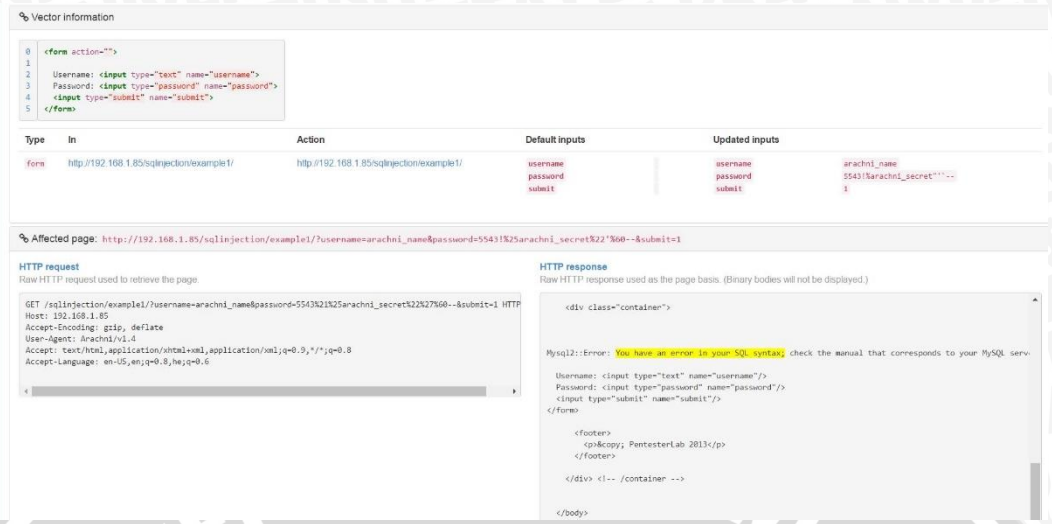
Hasil dari *scanning web for pentester* di sini dengan menggunakan *Wapiti* telah mendeteksi beberapa jenis *vulnerability* tersebut yaitu :

- *XSS*



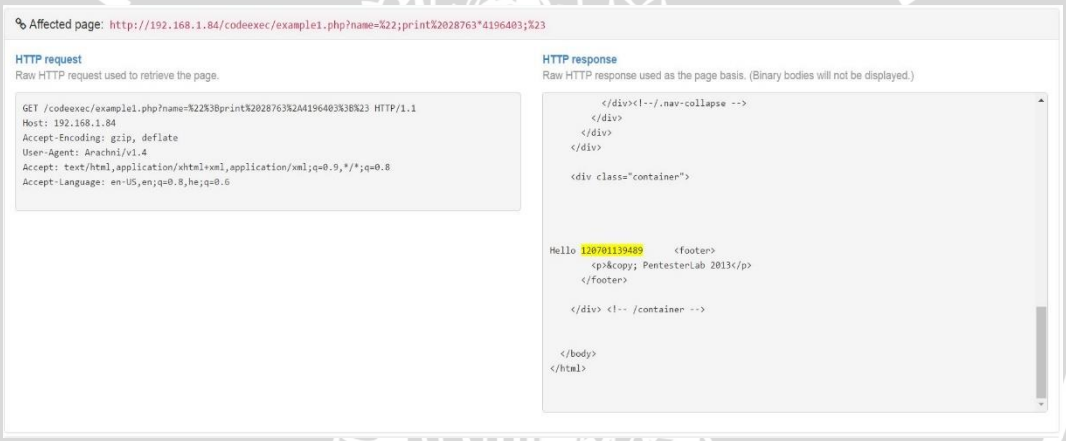
Gambar 4.42 Hasil Kerentanan *XSS* dengan tool *Arachni*

- *SQL Injection*



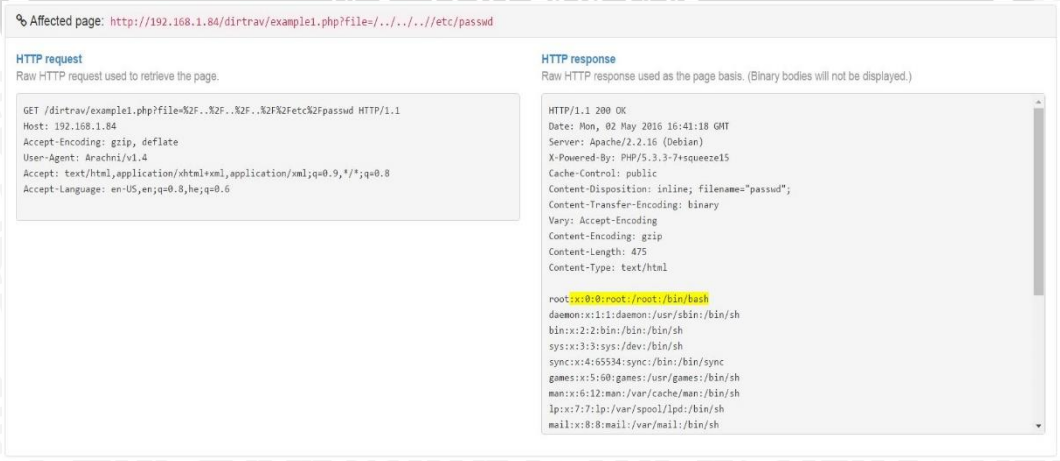
Gambar 4.43 Hasil Kerentanan *SQL Injection* dengan tool *Arachni*

- *Code Injection*



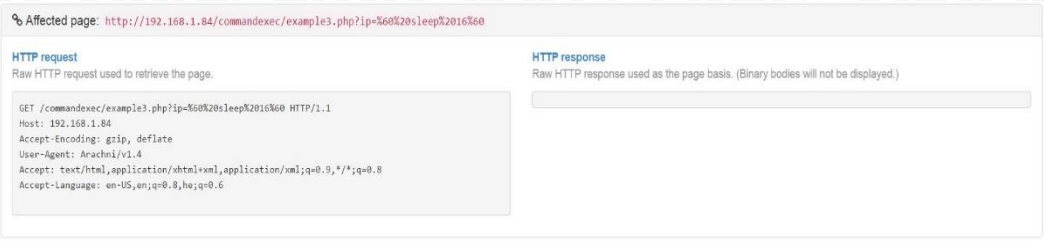
Gambar 4.44 Hasil Kerentanan *Code Injection* dengan tool *Arachni*

- *Directory Traversal*

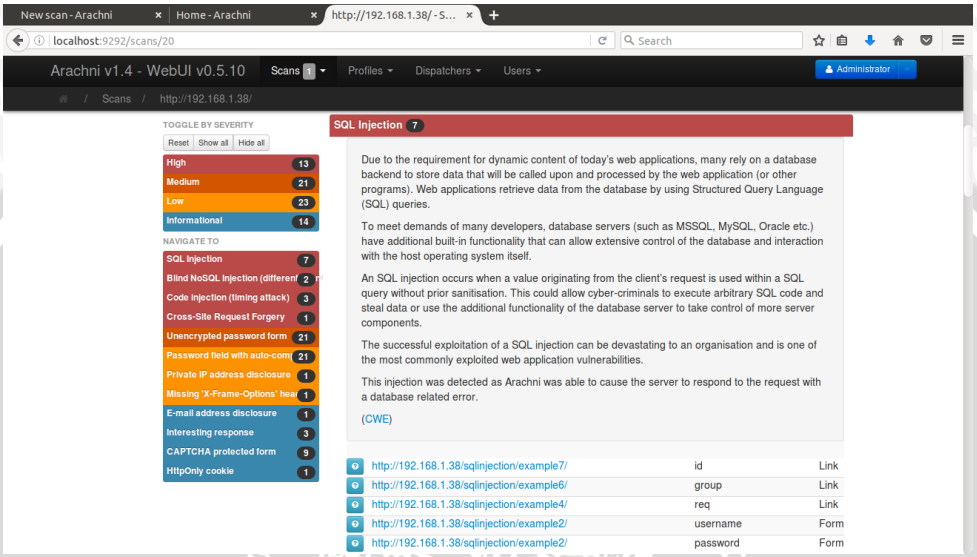


Gambar 4.45 Hasil Kerentanan *Directory Traversal* dengan tool *Arachni*

- *Command Injection*



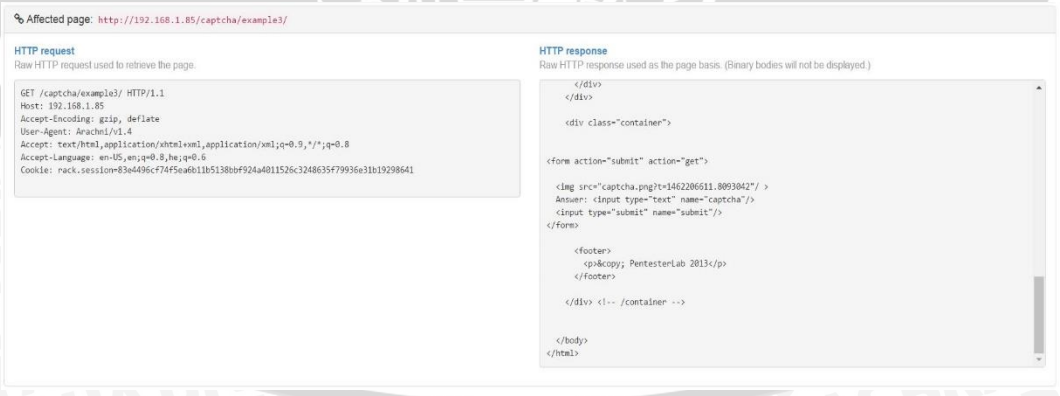
Gambar 4.46 Hasil Kerentanan *Command Injection* dengan tool *Arachni*



Gambar 4.47 Hasil Pengujian Otomatis *Arachni Web for pentester 2*

Hasil dari *scanning web for pentester 2* di sini dengan menggunakan *Wapiti* telah mendeteksi beberapa jenis *vulnerability* tersebut :

- *Captcha*

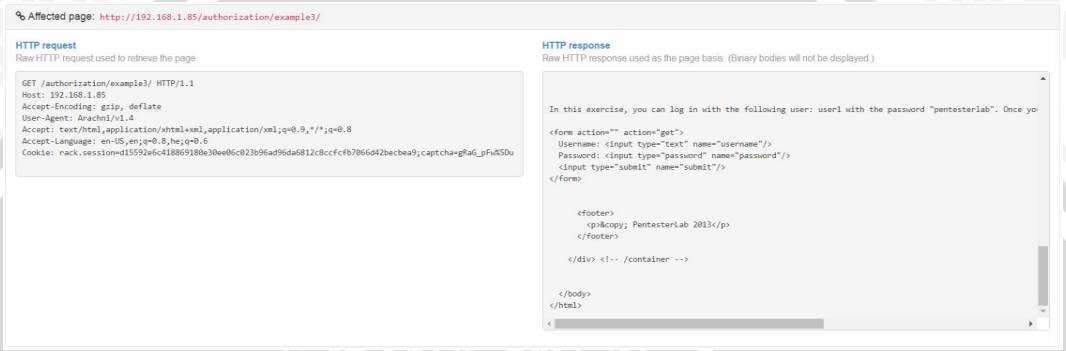


Gambar 4.48 Hasil Kerentanan *Captcha* dengan tool *Arachni*

- *Authentication*

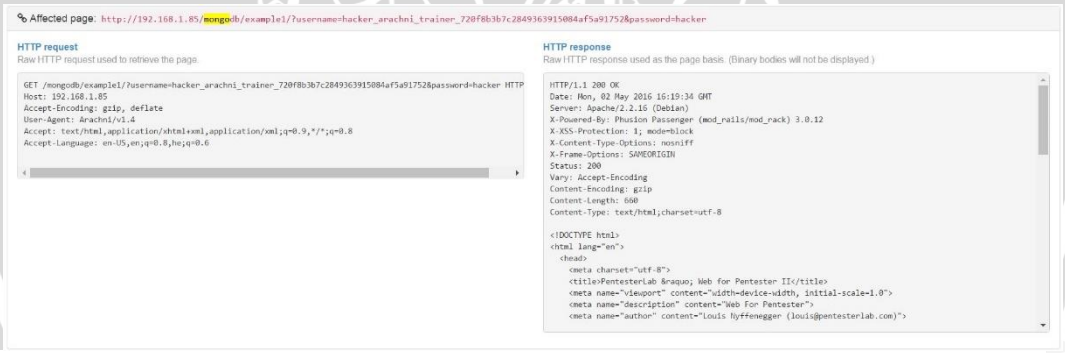
Gambar 4.49 Hasil Kerentanan *Authentication* dengan tool *Arachni*

- *Authorization*



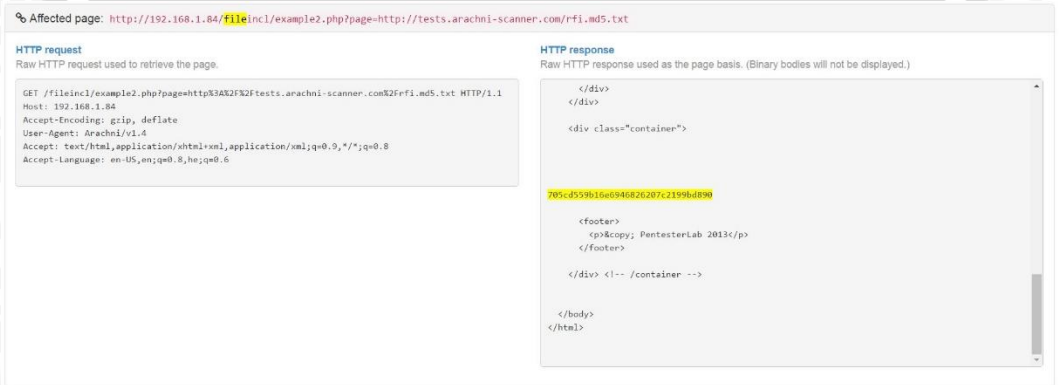
Gambar 4.50 Hasil Kerentanan *Authorization* dengan tool *Arachni*

- *Mongo DB*



Gambar 4.51 Hasil Kerentanan *Mongo DB* dengan tool *Arachni*

- *File Include*

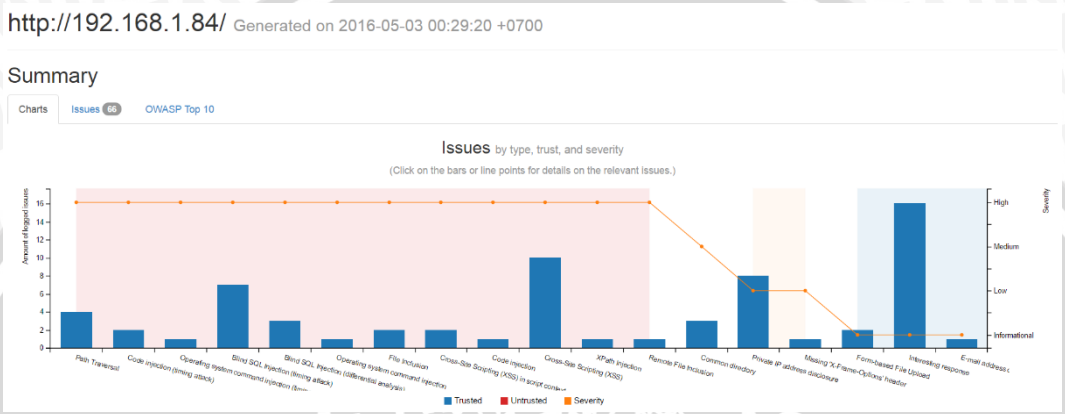


Gambar 4.52 Hasil Kerentanan *File Include* dengan tool *Arachni*

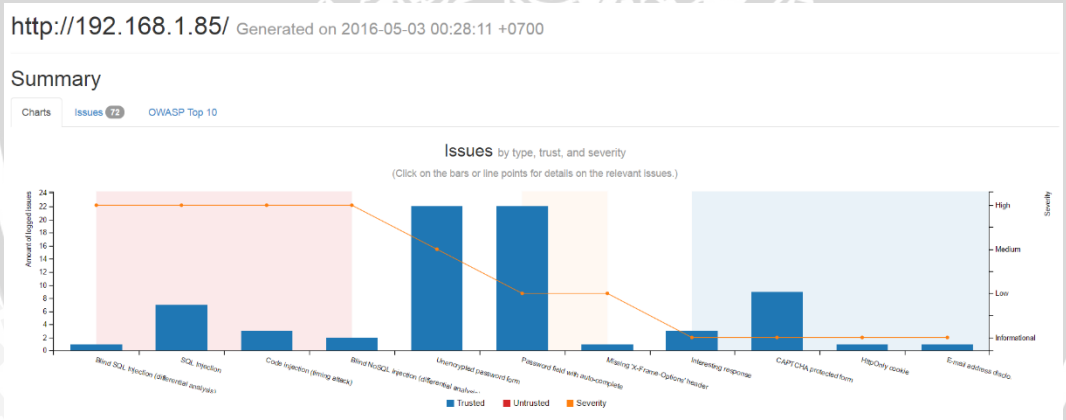
Setelah mengetahui hasil tersebut pada proses hasil identifikasi uji penetrasi pada *tool* yang ditetapkan maka dilakukan perbandingan dengan menggunakan tabel yang akan melihat *tool* mana saja yang dapat melakukan *scanning* terhadap pada *vulnerability* aplikasi *web www.pentesterlab.com*.

4.3.3.1 Hasil Waktu Identifikasi *Tool Arachni*

Dari identifikasi aplikasi web dengan menggunakan *tool Arachni* penulis melakukan pencatatan waktu identifikasi terhadap aplikasi *web* yang diuji dengan *web for pentester* dan *web for pentester 2* pada gambar dibawah.



Gambar 4.53 Hasil Identifikasi Waktu *Arachni Web For Pentester*



Gambar 4.54 Hasil Identifikasi Waktu *Arachni Web For Pentester 2*

Berdasarkan hasil waktu Identifikasi dari *tool Arachni* pada tabel berikut :

Aplikasi Web	Waktu
Web For Pentester	00.29.20
Web For Pentester 2	00.28.11

Tabel 4.3 Hasil Waktu Identifikasi *Tool Arachni*

Tabel 4.3 menunjukkan hasil dari waktu identifikasi dengan *tool Arachni* pada aplikasi *web challenge web for pentester* dengan waktu 00.29.20 sedangkan pada *web for pentester 2* mendapatkan waktu identifikasi sekitar 00.28.11.

Pengujian *One-Way ANOVA* dari hasil pengujian waktu pemrosesan untuk aplikasi web dari *tool w3af, wapiti,* dan *arachni*.

Untuk melihat apakah ada perbedaan yang signifikan dari data hasil pengujian waktu pemrosesan untuk aplikasi *web* dari *web for pentester* dan *web for pentester 2*. Hal ini dikarenakan *One-Way ANOVA* bisa digunakan untuk data yang lebih dari dua kelompok independen. Hasil pengujian *One-Way ANOVA* untuk data dari hasil pengujian waktu pemrosesan untuk aplikasi *web* dari *web for pentester 1* dan *web for pentester 2* untuk semua karakter dapat dilihat pada tabel 4.4

ANOVA

Waktu

	Sum of Squares	df	Mean Square	F	Sig.
Between Groups	2779737.333	2	1389868.667	288.564	.000
Within Groups	14449.500	3	4816.500		
Total	2794186.833	5			

Tabel 4.4 *One-Way ANOVA web for pentester dan web for pentester2.*

Ketentuan pengambilan keputusan didasarkan pada ketentuan sebagai berikut:
Hipotesis:

H0: Tidak ada perbedaan yang signifikan antara waktu pengujian waktu pemrosesan pada tool uji penetrasi w3af, wapit, dan arachni

H1: Terdapat perbedaan yang signifikan antara waktu pengujian waktu pemrosesan pada tool uji penetrasi w3af, wapit, dan arachni

Kriteria keputusan:

a. Jika probabilitas (Sig.) > 0,5 maka H0 diterima
b. Jika probabilitas (sig.) < 0,5 maka H0 ditolak

Diketahui nilai sig. dari hasil perhitungan *One-Way ANOVA* dari hasil pengujian waktu pemrosesan untuk *web for pentester* dan *web for pentester 2* untuk semua jumlah karakter adalah 0,000, maka H0 ditolak. Artinya adanya perbedaan yang signifikan antara pengujian waktu pemrosesan untuk perbandingan tool uji penetrasi *w3af, wapiti*, dan *arachni*.

4.3.4 Hasil Tabel Pengujian Otomatis

Pengechekan terhadap *vulnerabilty web for pentester* dan *web for pentester 2* dengan menggunakan tabel agar bisa mengetahui *tool* mana saja yang dapat menemukan keretanan dengan hasil yang baik. Berikut hasil tabel uji penetrasi dengan 5 kali percobaan aplikasi *web www.pentesterlab.com* :

PERCOBAAN 1			
VULNERABILTIY/ APPLICATION WEB SCANNER	*W3AF*	*ARACHNI*	*WAPITI*
XSS	✓	✓	✓

SQL INJECTION	✓	✓	✓
DIRECTORY TRAVERSAL	✓	✓	✓
CODE INJECTION	✓	✓	✓
COMMAND INJECTION	✓	✓	✓
AUTHENTICATION		✓	
AUTHORIZATION		✓	
CAPTCHA	✓	✓	
MONGO DB		✓	✓
FILE INCLUDE	✓	✓	✓

Tabel 4.5 Hasil Percobaan 1 pada Pengujian Otomatis

PERCOBAAN 2			
VULNERABILTIY/ APPLICATION WEB SCANNER	*W3AF*	*ARACHNI*	*WAPITI*
XSS	✓	✓	✓
SQL INJECTION	✓	✓	✓
DIRECTORY TRAVERSAL	✓	✓	✓
CODE INJECTION	✓	✓	✓
COMMAND INJECTION	✓	✓	✓
AUTHENTICATION		✓	
AUTHORIZATION	✓	✓	
CAPTCHA	✓	✓	
MONGO DB			✓
FILE INCLUDE	✓	✓	✓

Tabel 4.6 Hasil Percobaan 2 pada Pengujian Otomatis

PERCOBAAN 3			
VULNERABILTIY/ APPLICATION WEB SCANNER	*W3AF*	*ARACHNI*	*WAPITI*
XSS	✓	✓	✓
SQL INJECTION	✓	✓	✓
DIRECTORY TRAVERSAL	✓	✓	✓
CODE INJECTION	✓	✓	✓
COMMAND INJECTION	✓	✓	✓
AUTHENTICATION			
AUTHORIZATION		✓	
CAPTCHA	✓	✓	
MONGO DB			✓

FILE INCLUDE	✓	✓	✓

Tabel 4.7 Hasil Percobaan 3 pada Pengujian Otomatis

PERCOBAAN 4			
VULNERABILTIY/ APPLICATION WEB SCANNER	*W3AF*	*ARACHNI*	*WAPITI*
XSS	✓	✓	✓
SQL INJECTION	✓	✓	✓
DIRECTORY TRAVERSAL	✓	✓	✓
CODE INJECTION	✓	✓	✓
COMMAND INJECTION	✓	✓	✓
AUTHENTICATION			
AUTHORIZATION			
CAPTCHA	✓	✓	
MONGO DB		✓	✓
FILE INCLUDE		✓	✓

Tabel 4.8 Hasil Percobaan 4 pada Pengujian Otomatis

PERCOBAAN 5			
VULNERABILTIY/ APPLICATION WEB SCANNER	*W3AF*	*ARACHNI*	*WAPITI*
XSS	✓	✓	✓
SQL INJECTION	✓	✓	✓
DIRECTORY TRAVERSAL	✓	✓	✓
CODE INJECTION	✓	✓	✓
COMMAND INJECTION	✓	✓	✓
AUTHENTICATION			✓
AUTHORIZATION			
CAPTCHA	✓	✓	
MONGO DB		✓	✓
FILE INCLUDE	✓	✓	✓

Tabel 4.9 Hasil Percobaan 5 pada Pengujian Otomatis

Dari Tabel 4.1 hingga tabel 4.5 Ini adalah hasil dari tabel pengujian *penetration testing tool* sebanyak 5 kali pada aplikasi *web www.pentesterlab.com* pada *tool W3af,Arachni,*dan *Wapiti* dapat diketahui dari tabel tersebut adalah bahwa *tool Arachni* lebih banyak mendapatkan hasil *vulnerabilty* dari ke dua *tool* yang telah diuji. *Arachni* juga sebenarnya memiliki kelebihan pada uji penetrasi yaitu dapat mengidentifikasi aplikasi *web* dengan menampilkan hasil scan yang lebih teknis sedangkan hasil dari *Wapiti* dan *W3af* sendiri tidak terlalu banyak mendapatkan hasil *scanning* dari *web www.pentesterlab.com*. *Wapiti* dan *W3af* hanya mampu

dapat mendeteksi kerentanan seperti *XSS*, *SQL Injection*, *Directory Traversal*, *Command Injection*,dan *File Include*. *Wapiti* dan *W3af* juga memungkinkan penguji untuk mengaudit keamanan aplikasi *web* yang diuji,dan melakukan "*black-box*" scan,yaitu tidak dapat mengetahui sumber kode dari aplikasi tetapi akan melakukan scan halaman aplikasi yang dituju,dan mencari script dan bentuk dimana dapat memasukkan data.

BAB 5
PEMBAHASAN

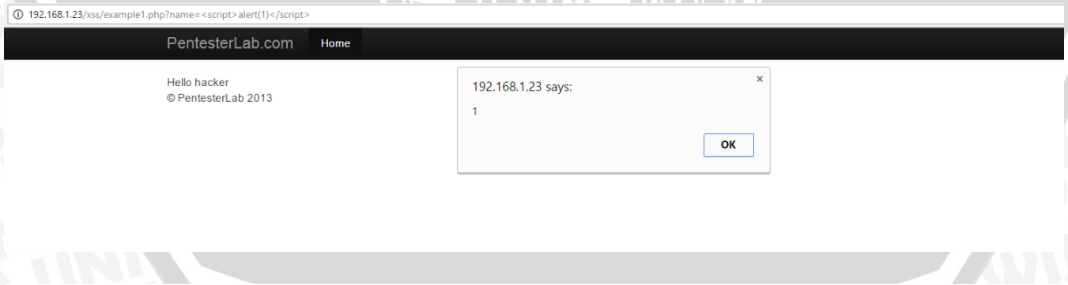
Pada bab 5 ini dilakukan analisis dan pembahasan secara menyeluruh terhadap hasil yang telah didapat pada proses analisa perbandingan *penetration testing tool* untuk aplikasi *web www.pentesterlab.com*. Hasil yang telah didapat pengujian tersebut dengan melalui proses secara manual yaitu dengan meyerang langsung *challenge* yang ada dari aplikasi *web www.pentesterlab.com* dengan mengetahui kerentanan apa saja yang dapat di ketahui pada aplikasi *web www.pentesterlab.com* tersebut, pada parameter yang dituju dengan celah kemana yang di uji langsung *seperti XSS,SQL Injection, Directory Traversal, Authentication, Captcha, Authorization, Mongo DB injection, File Include, Code Injection, dan Command Injection.* Serta pengujian otomatis secara otomatis yang dilakukan dengan bantuan *software W3af*, *Wapiti*, dan *Arachni* untu mengetahui celah keamanan yang tidak dapat terdeteksi saat proses pengujian secara manual untuk menemukan celah keamanan yang terdapat pada aplikasi *web www.pentesterlab.com*.

5.1 Pembahasan Pengujian

Pada bagian ini dilakukan pembahasan terhadap hasil yang didapat pada pengujian celah keamanan secara manual dan otomatis. Pada hasil celah keamanan yang didapat pada bab sebelumnya telah diketahui bahwa terdapat beberapa celah keamanan yang terdeteksi dari pengujian manual dan otomatis terhadap *web for pentester* 1 dan *web for pentester 2* yang telah di uji yang dijelaskan pada sub-bab berikut.

5.1.1 XSS

5.1.1.1 Pengujian Manual



Gambar 5.1 Hasil Pengujian Manual *XSS*

Gambar 5.1 menunjukkan Serangan *XSS* pada aplikasi *web www.pentesterlab.com* dilakukan dengan menginputkan kode-kode html pada aplikasi dan hasilnya aplikasi merespon dengan menyetujui inputan pengguna tersebut, namun saat dilakukan pengujian dengan menginputkan *script* pada inputan aplikasi menolak request pengguna dengan menghapus perintah *script*.

5.1.1.2 Pengujian Otomatis

A. *W3af*

Vulnerability	tcp/80	Cross Site Scripting was found at: "http://192.168.1.26/xss/example1.php", using HTTP method GET. The sent data was: "name=<SCrIPT>alert("mVtV")</SCrIPT>". This vulnerability affects ALL browsers. This vulnerability was found in the request with id 931. **URL :** http://192.168.1.26/xss/example1.php Severity : Medium

Gambar 5.2 Hasil Pengujian Otomatis *XSS W3af*

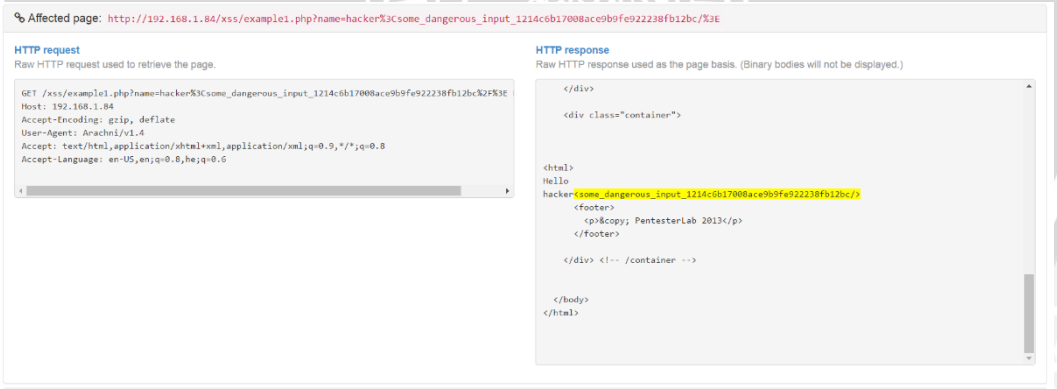
Gambar 5.2 menunjukkan pengujian tool *W3af* dalam identifikasi aplikasi web menemukan kerentanan XSS yang telah ditemukan dengan *HTTP Request* "name=<SCrIPT>alert<"mVtV")</SCrIPT>".

B. *Wapiti*

Gambar 5.3 Hasil Pengujian Otomatis *XSS Wapiti*

Gambar 5.3 menunjukkan identifikasi dari tool *Wapiti* menemukan kerentanan XSS yang telah di temukan pada *HTTP Request* GET /xss/example1.php?name=%3Ealert%28%27wwn5uiqu5h%27%29%3C%2Fscript%3E.

C. *Arachni*



Gambar 5.4 Hasil Pengujian Otomatis *XSS Arachni*

Gambar 5.4 Hasil identifikasi menunjukkan tool *Arachni* menemukan adanya kerentanan XSS pada aplikasi web dengan *HTTP Request* hacker<some_dangerous_input_1214c6b17008ace9b9fe922238fb12bc/>.

5.1.2 SQL Injection

5.1.2.1 Pengujian Manual

Gambar 5.5 Hasil Pengujian *Manual SQL Injection*

Gambar 5.5 menunjukkan Pengujian terhadap *SQL Injection* pada *challenge web for pentester* dengan menginputkan *" ' or '1='1 "* pada name di browser tersebut untuk menguji kerentanan aplikasi. Hasil yang didapat diketahui bahwa aplikasi *web www.pentesterlab.com* telah melakukan pencegahan dengan menginputkn merubah tanda " ' " dengan *"%27"*, dan menghilangkan spasi.Sehingga saat request dieksekusi input tersebut dapat terbaca sebagai inputan biasa yang dikenali sebagai username.

5.1.2.2 Pengujian Otomatis

A. *W3af*

Vulnerability	tcp/80	SQL injection in a MySQL database was found at: "http://192.168.1.25/sqlinjection/example2/", using HTTP method GET. The sent data was: "username=John8212&password=d'z"0&submit=". The modified parameter was "password". This vulnerability was found in the request with id 9049. **URL :** http://192.168.1.25/sqlinjection/example2/ Severity : High

Gambar 5.6 Hasil Pengujian Otomatis *SQL Injection W3af*

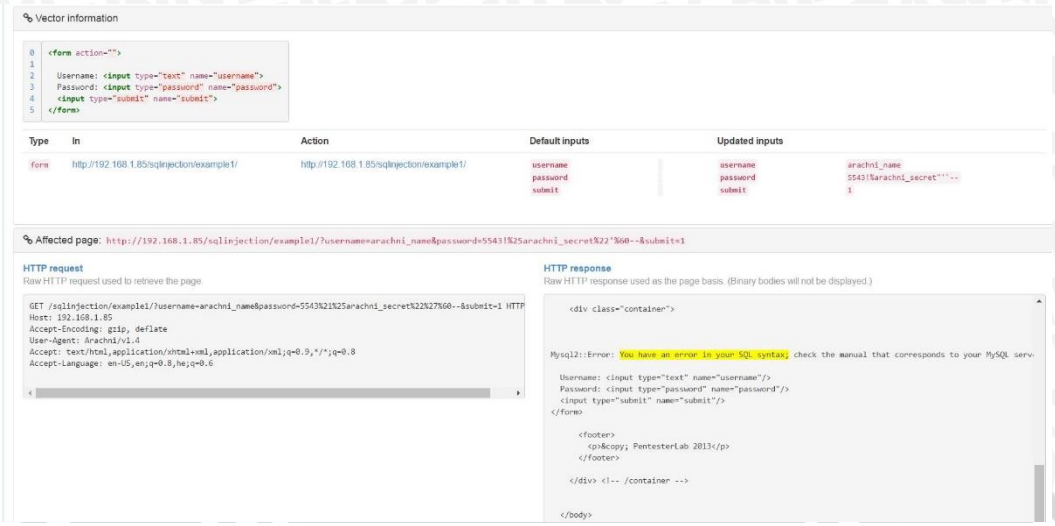
Gambar 5.6 menunjukkan hasil pengujian dengan tool *W3af*.Adanya kerentanan *SQL Injection* dengan *HTTP Request* "username=Johm8128password=d"z"0&submit=".

B. *Wapiti*

Gambar 5.7 Hasil Pengujian Otomatis *SQL Injection Wapiti*

Gambar 5.7 menunjukkan hasil identifikasi dengan menggunakan tool *Wapiti*. Adanya kerentanan SQL Injection dengan *HTTP Request* GET /sqli/example1.php?name=%27%20or%20sleep%28%29%231 HTTP/1.1.

C. *Arachni*

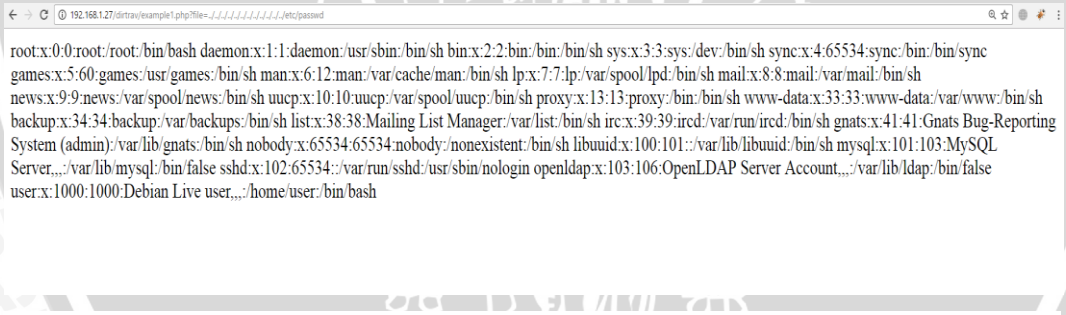


Gambar 5.8 Hasil Pengujian Otomatis *SQL Injection Arachni*

Gambar 5.8 menunjukkan hasil pengujian otomatis dengan tool *Arachni* dengan menemukan kerentana pada SQL Injection dengan *HTTP Request* ?usename=*Arachni*_name&password=5543!%25*Arachni*_secret%22”%60--&submit=1.

5.1.3 Directory Traversal

5.1.3.1 Pengujian Manual



Gambar 5.9 Hasil Pengujian Manual *Directory Traversal*

Gambar 5.9 menunjukkan pada pengujian manual *Directory Traversal* penyerang dapat mengakses direktori di atas dengan menggunakan teknik menguji sebuah kerentanan. Sebagai contoh jika cara yang digunakan dalam parameter adalah gambar/*photo.jpg* dan dapat diakses melalui element pada *browser*.Gambar tersebut mempunyai sebuah akses ke direktori untuk menembus sebuah kerentanan dan jika di test dengan menambahkan *string ../../../../../../../../../../../etc/passwd* dan mendapatkan *passwd file*,aplikasi *web* tersebut adalah mempunyai kerentanan.Setelah direktori ini diakses, perintah dapat dijalankan pada *server Web* dan data aman bisa disalin atau diubah.Keamanan dari *server Web* terancam.Kerentanan *Directory Traversal* juga dapat ditemukan dalam aplikasi *Web* yang dijalankan di *server Web*.

5.1.3.2 Pengujian Otomatis

A. *W3af*

Vulnerability	tcp/80	Local File Inclusion was found at: "http://192.168.1.26/dirtrav/example1.php", using HTTP method GET. The sent data was: "file=../../../../../../../../../../../../../etc/passwd". This vulnerability was found in the request with id 5267. **URL :** http://192.168.1.26/dirtrav/example1.php Severity : Medium

Gambar 5.10 Hasil Pengujian Otomatis *Directory Traversal W3af*

Gambar 5.10 menunjukkan hasil dari identifikasi *tool W3af* menemukan adanya kerentanan pada *directory traversal* dengan *HTTP Request* "file=../../../../../../../../../../../../etc/passwd".

B. *Wapiti*

Vulnerability found in /dirtrav/example1.php

Description | HTTP Request | cURL command line

```
GET /dirtrav/example1.php?file=..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2Fetc%2Fpasswd HTTP/1.1
Host: 192.168.1.35
```

Gambar 5.11 Hasil Pengujian Otomatis *Directory Traversal Wapiti*

Gambar 5.11 menunjukkan hasil dari identifikasi dari tool *Wapiti* yang menemukan kerentanan yang ditemukan pada *directory traversal* dengan *HTTP Request*/dirtrav/example.php?file=..%2F..%2F..%2F..%2F..%2F..%2F..%2F..%2Fetc %2Fpasswd HTTP/1.1.

C. *Arachni*

Affected page: http://192.168.1.84/dirtrav/example1.php?file=/../../..//etc/passwd

HTTP request
Raw HTTP request used to retrieve the page.

```
GET /dirtrav/example1.php?file=%2F..%2F..%2F..%2F%2Fetc%2Fpasswd HTTP/1.1
Host: 192.168.1.84
Accept-Encoding: gzip, deflate
User-Agent: Arachni/v1.4
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
Accept-Language: en-US,en;q=0.8,he;q=0.6
```

HTTP response
Raw HTTP response used as the page basis. (Binary bodies will not be displayed.)

```
HTTP/1.1 200 OK
Date: Mon, 02 May 2016 16:41:18 GMT
Server: Apache/2.2.16 (Debian)
X-Powered-By: PHP/5.3.3-7+squeeze15
Cache-Control: public
Content-Disposition: inline; filename="passwd";
Content-Transfer-Encoding: binary
Vary: Accept-Encoding
Content-Encoding: gzip
Content-Length: 475
Content-Type: text/html

root:x:0:0:root:/root:/bin/bash
daemon:x:1:1:daemon:/usr/sbin:/bin/sh
bin:x:2:2:bin:/bin:/bin/sh
sys:x:3:3:sys:/dev:/bin/sh
sync:x:4:65534:sync:/bin:/bin/sync
games:x:5:60:games:/usr/games:/bin/sh
man:x:6:12:man:/var/cache/man:/bin/sh
lp:x:7:7:lp:/var/spool/lpd:/bin/sh
mail:x:8:8:mail:/var/mail:/bin/sh
```

Gambar 5.12 Hasil Pengujian Otomatis *Directory Traversal Arachni*

Gambar 5.12 menunjukkan adanya kerentanan pada SQL Injection dengan menggunakan tool *Arachni*. Adanya kerentanan dengan *HTTP Request* GET /dirtrav/example1.php?file=%2F.. %2F.. %2F.. %2Fect2%Fpasswd HTTP/1.1.

5.1.4 Command Injection

5.1.4.1 Pengujian Manual

Gambar 5.13 Hasil Pengujian Manual *Command Injection*

Gambar 5.13 menunjukkan pengujian manual *Command Injection* ini merupakan penyerangan melalui *web* browser dengan mengijeksikan pada sistem operasi melalui aplikasi kerentanan. Serangan yang dilakukan disediakannya dengan data yang tidak aman dari aplikasi *web* tersebut.

5.1.4.2 Pengujian Otomatis

A. *W3af*

Vulnerability	tcp/80	OS Commanding was found at: "http://192.168.1.26/commandexec/example1.php", using HTTP method GET. The sent data was: "ip=%7Cping+-c+9+localhost". This vulnerability was found in the request with id 2487. **URL :** http://192.168.1.26/commandexec/example1.php Severity : High

Gambar 5.14 Hasil Pengujian Otomatis *Command Injection W3af*

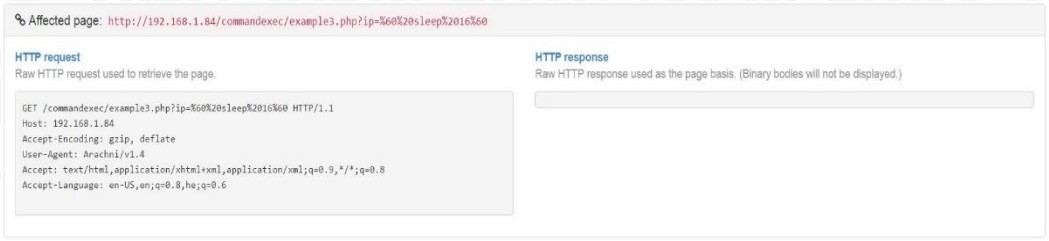
Gambar 5.14 menunjukkan pengujian otomatis dengan tool *W3af* menemukan adanya kerentanan pada command injection dengan *HTTP Request* “ip%7Cping+c+9+localhost”.

B. *Wapiti*

Gambar 5.15 Hasil Pengujian Otomatis *Command Injection Wapiti*

Gambar 5.15 menunjukkan hasil identifikasi dari tool *Wapiti*. Adanya kerentanan yang pada command injection dengan *HTTP Request* GET /commandexec/example1.php?ip=%3Benv HTTP/1.1.

C. *Arachni*

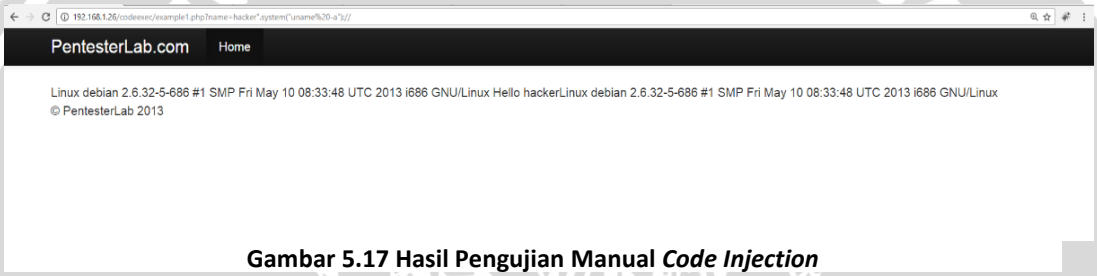


Gambar 5.16 Hasil Pengujian Otomatis *Command Injection Arachni*

Gambar 5.16 menunjukkan hasil dari identifikasi kerentanan command injection dengan menggunakan tool *Arachni* dengan *HTTP Request* GET /commandexec/example3.php?ip=%60%20sleep%2016%60 HTTP/1.1.

5.1.5 Code Injection

5.1.5.1 Pengujian Manual



Gambar 5.17 Hasil Pengujian Manual *Code Injection*

Gambar 5.17 menunjukkan pengujian manual *Code Injection* ini bahwa untuk melakukannya dengan payload umum, exploit yang digunakan untuk mengeksekusi *shellcode*.Payload kemudan akan menjalankan shellcode yang dipilih untuk mendapatkan target komputer dan mendapatkan akses.

5.1.5.2 Pengujian Otomatis

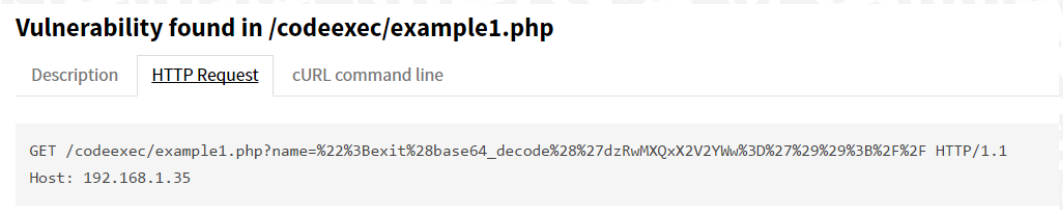
A. *W3af*

Information	tcp/80	Cross Site Scripting was found at: "http://192.168.1.26/codeexec/example1.php", using HTTP method GET. The sent data was: "name=<ScRIPT>a=/JS8T/%0Aalert(a.source)</SCRiPT>". This vulnerability affects ALL browsers. This vulnerability was found in the request with id 980. **URL :** http://192.168.1.26/codeexec/example1.php

Gambar 5.18 Hasil Pengujian otomatis *Code Injection W3af*

Gambar 5.18 menunjukkan hasil dari pengujian otomatis dengan menggukanan tool *W3af*. Adanya hasil identifikasi dari kerentanan code injection dengan *HTTP Request* <ScRIPT>a=/JS8T/%0Aalert(a.source)</SCRiPT>.

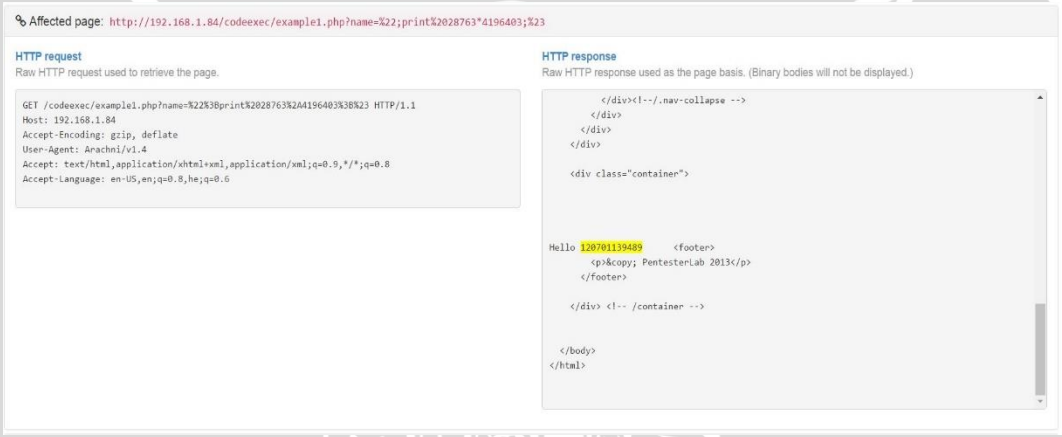
B. *Wapiti*



Gambar 5.19 Hasil Pengujian otomatis *Code Injection Wapiti*

Gambar 5.19 menunjukkan hasil dari pengujian otomatis dengan menggukanan tool *Wapiti*. Adanya hasil identifikasi dari kerentanan code injection dengan *HTTP Request* GET /codeexec/example1.php?name=%22%3Bexit%28%base64_decode%28%27dzRwMXQxX2V2YWw%3D%27%29%29%%3B%2F%2F HTTP/1.1.

C. *Arachni*

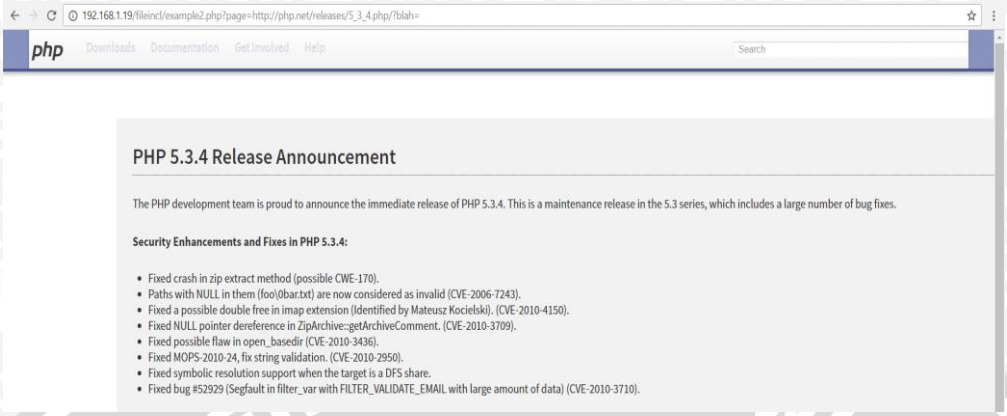


Gambar 5.20 Hasil Pengujian otomatis *Code Injection Arachni*

Gambar 5.20 menunjukkan hasil dari pengujian otomatis dengan menggukanan *tool Arachni*. Adanya hasil identifikasi dari kerentanan *code injection* dengan *HTTP Request* GET /codeexec/example1.php?name=%22%3Bprint%2028763%2A4q96403%3B%23 HTTP/1.1.

5.1.6 File Include

5.1.6.1 Pengujian Manual



Gambar 5.21 Hasil Pengujian Manual *File Include*

Gambar 5.21 menunjukkan kerentanan pada *File Include* yang datang dari kurangnya dari penyaringan ketika dikendalikan oleh user dari aplikasi *web* dengan parameter yang di gunakan sebagai bagian dari sebuah nama file dalam memanggil sebuah fungsi untuk melakukan panggilan kesalah satu metodi ini memiliki kerentanan,dan bahwa penyerang dapat melakukan memanipulasi fungsi untuk mengisi kode tersebut. Dapat di coba dengan remote *File Include*.

5.1.6.2 Pengujian Otomatis

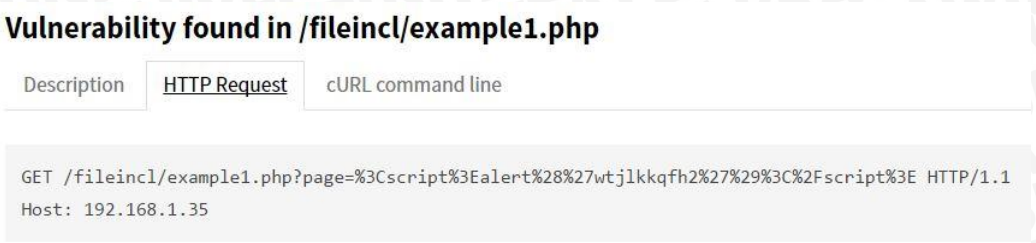
A. *W3af*

Vulnerability	tcp/80	Local File Inclusion was found at: "http://192.168.1.26/fileincl/example1.php", using HTTP method GET. The sent data was: "page=../../../../../../../../../../../../etc/passwd". This vulnerability was found in the request with id 4956. **URL :** http://192.168.1.26/fileincl/example1.php Severity : Medium

Gambar 5.22 Hasil Pengujian Otomatis *File Include W3af*

Gambar 5.22 menunjukkan hasil dari pengujian otomatis dengan menggukanan *tool W3af*. Adanya hasil identifikasi dari kerentanan *File Include* dengan *HTTPRequest* “page=../../../../../../../../../../../../etc/passwd”.

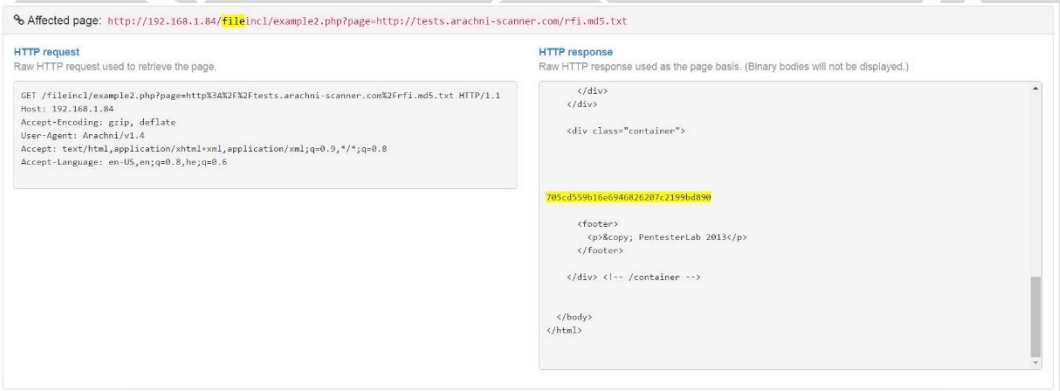
B. *Wapiti*



Gambar 5.23 Hasil Pengujian Otomatis *File Include Wapiti*

Gambar 5.23 menunjukkan hasil dari pengujian otomatis dengan menggukanan *tool Wapiti*. Adanya hasil identifikasi dari kerentanan *File Include* dengan *HTTP Request* GET /fileinlc/example1.php?page=%3Cscript%3Ealert%28%27wtjkkqfh2%27%29%3C%2Fscripts%3E HTTP/1.1.

C. *Arachni*

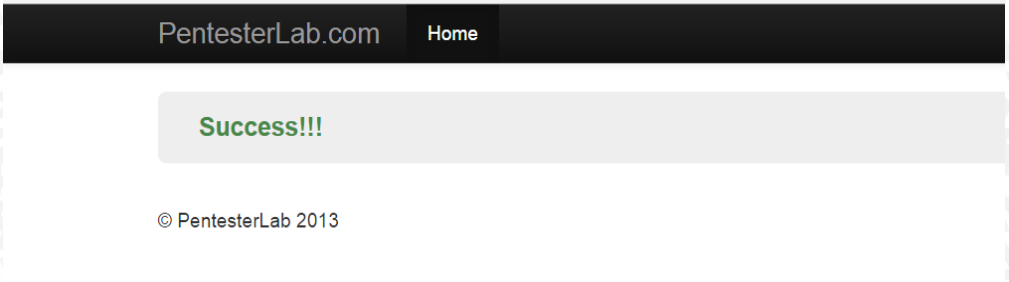


Gambar 5.24 Hasil Pengujian Otomatis *File Include Arachni*

Gambar 5.24 menunjukkan hasil dari pengujian otomatis dengan menggukanan *tool Arachni*. Adanya hasil identifikasi dari kerentanan *File Include* dengan *HTTP Request* GET /fileincl/example2.php?page=http%3A%2F%2Ftest.*Arachni*.scanner.com%2Frfi.md5.txt HTTP/1.1.

5.1.7 Authentication

5.1.7.1 Pengujian Manual



Gambar 5.25 Hasil Pengujian Manual *Authentication*

Gambar 5.25 menunjukkan pengujian manual *Authentication* ini dengan cara memasukkan username dan password secara umum pada *challenge Authentication* tersebut agar pada proses authentification yang berhasil biasanya ditandai dengan *username* dan *password* yang tepat yang dimasukkan oleh pemakai aplikasi. Setiap pemakai aplikasi diberi *username* dan *password* yang berbeda-beda dan hanya diketahui oleh user yang bersangkutan. Jika suatu login berhasil, maka dipastikan bahwa yang login adalah user yang bersangkutan karena hanya dialah yang tahu *username* dan *passwordnya*

5.1.7.2 Pengujian Otomatis

A. *W3af*

Vulnerability	tcp/80	An unidentified web application error (HTTP response code 500) was found at: "http://192.168.1.25/authentication/example6/submit". Enable all plugins and try again, if the vulnerability still is not identified, please verify mannually and report it to the w3af developers. This vulnerability was found in the request with id 1153. **URL :** http://192.168.1.25/authentication/example6/submit Severity : Medium

Gambar 5.26 Hasil Pengujian Otomatis *Authentication W3af*

Gambar 5.26 menunjukkan adanya error pada kerentanan *authentication* pada *W3af* dikarenakan *HTTP*respon code 500 yang menyatakan untuk memberi akses ijin semua plugin dan mencoba kembali, apabila kerentanan tidak dapat teridentifikasi, dapat melaporkan kembali secara manual pada develpoer *W3af*.

B. *Wapiti*

Gambar 5.27 Hasil Pengujian Otomatis *Authentication Wapiti*

Gambar 5.27 menunjukkan adanya error pada kerentanan *authentication* pada *Wapiti* dikarenakan *HTTP* respon code 500 ketika mencoba menginjeksi sebuah payload kedalam query string.

C. Arachni

Gambar 5.28 Hasil Pengujian *Authentication Arachni*

Gambar 5.28 menunjukkan hasil dari pengujian otomatis dengan menggukanan *tool Arachni*. Adanya hasil identifikasi dari kerentanan a*uthentication* tersebut *tool Arachni* menemukan bahwa halaman yang terpengaruh berisi masukan password,namun hasil tersebut tidak dikirim server menggunakan *HTTPS*.

5.1.8 Authorization

5.1.8.1 Pengujian Manual

Gambar 5.29 Hasil Pengujian Manual *Authorization*

Gambar 5.29 menunjukkan hasil pengujian manual *Authorization* ini adalah permasalahan yang pada umumnya dengan aplikasi *web* yang dirancang dengan tidak baik, bahkan jika tidak dapat mengakses halaman posting-autentikasi

penguji masih dapat mengakses halaman lain jika sudah tahu URL tersebut. pengujian manual ini dengan menginputkan */infos/1, /infos/2* tanpa menggunakan login lagi dan halaman tersebut dapat memasukin akses yang dituju.

5.1.8.2 Pengujian Otomatis

A. *W3af*

Gambar 5.30 Hasil Pengujian Otomatis *Authorization W3af*

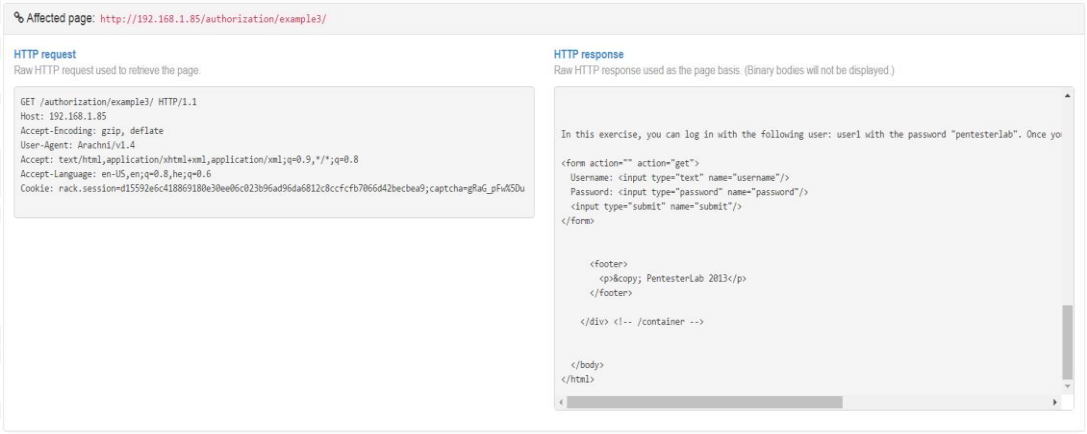
Gambar 5.30 menunjukkan adanya error pada kerentanan *authentication* pada *W3af* dikarenakan *HTTP*respon code 500 yang menyatakan untuk memberi akses ijin semua plugin.

B. *Wapiti*

Gambar 5.31 Hasil Pengujian Otomatis *Authorization Wapiti*

Gambar 5.31 menunjukkan adanya error pada kerentanan *authentication* pada *Wapiti* dikarenakan *HTTP* respon code 500 ketika mencoba menginjeksi sebuah *payload* kedalam *query string*.

C. *Arachni*

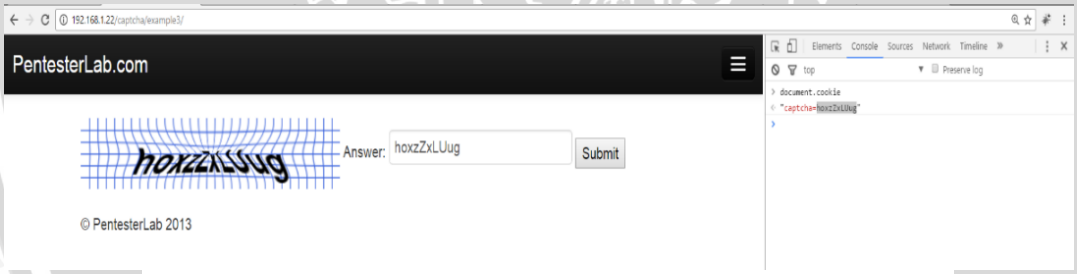


Gambar 5.32 Hasil Pengujian Otomatis *Authorization Arachni*

Gambar 5.32 menunjukkan hasil dari pengujian otomatis dengan menggukanan *tool Arachni*. Adanya hasil identifikasi dari kerentanan a*uthorization* tersebut *tool Arachni* menemukan bahwa halaman yang terpengaruh berisi masukan password,tetapi hasil tersebut tidak dikirim server menggunakan *HTTPS*.

5.1.9 Captcha

5.1.9.1 Pengujian Manual

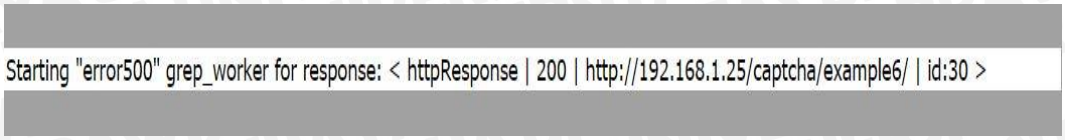


Gambar 5.33 Hasil Pengujian Manual *Captcha*

Gambar 5.33 menunjukkan pengujian ini dibocorkan oleh aplikasi , dengan memeriksa respon dikirim kembali oleh *server*, penguji dapat mengambil *Captcha* dari *cookie* menggunakan konsol *JavaScript* dan panggilan *document.cookie*.Pengujian *Captcha* ini menggunakan *javascript* untuk mengetahui langsung apa yang ditampilkan gambar pada *Captcha* dan menggunakan *javascript* agar dapat mudah mengetahui dari arti gambar *Captcha* tersebut.

5.1.9.2 Pengujian Otomatis

A. *W3af*



Gambar 5.34 Hasil Pengujian Otomatis *Captcha W3af*

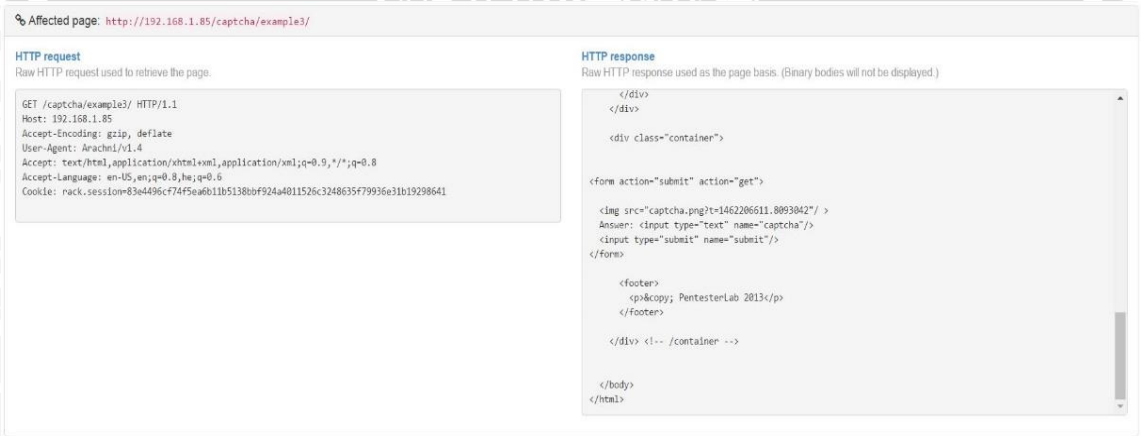
Gambar 5.34 menunjukkan adanya error pada kerentanan *captcha* pada *W3af* dikarenakan *HTTP*respon code 500 yang menyatakan untuk memberi akses ijin semua plugin.

B. *Wapiti*

Gambar 5.35 Hasil Pengujian Otomatis *Captcha Wapiti*

Gambar 5.35 menunjukkan adanya error pada kerentanan *captcha* pada *Wapiti* dikarenakan *HTTP*respon code 500 ketika mencoba menginjeksi sebuah payload kedalam query string.

C. *Arachni*

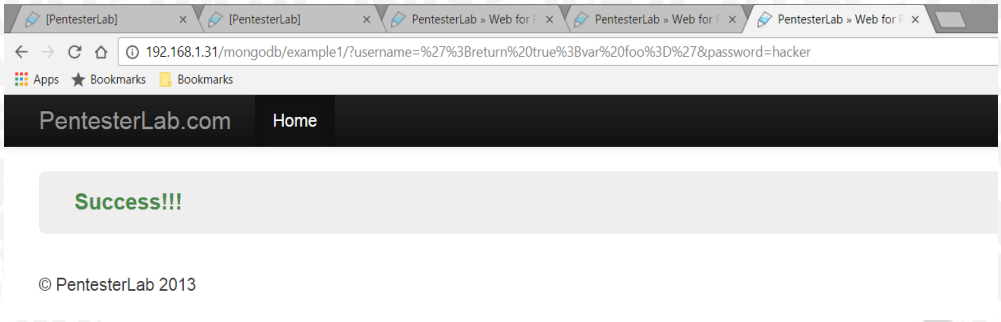


Gambar 5.36 Hasil Pengujian Otomatis *Captcha Arachni*

Gambar 5.36 menunjukkan dari identifikasi dengan tool *Arachni* tidak menemukan sebagai kerentanan, tetapi sebagai prompt untuk uji penetrasi secara manual.

5.1.10 Mongo DB Injection

5.1.10.1 Pengujian Manual



Gambar 5.37 Hasil Pengujian Manual *Mongo DB Injection*

Gambar 5.37 menunjukkan *Mongo DB* merupakan salah satu produk *database noSQL* yang menggunkan struktur data JSON untuk menyimpan data tersebut. Dari hasil pengujian manual tersebut dapat dilihat dari kolom *browser* adalah berupa perintah *database* yang tidak menggunakan relasi antar tabel dan tidak menyimpan data dalam format tabel kaku seperti layaknya relasional database.

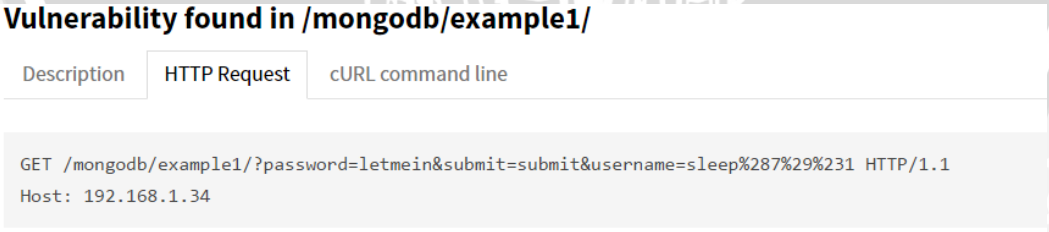
5.1.10.2 Pengujian Otomatis

A. *W3af*

Gambar 5.38 Hasil Pengujian Otomatis *Mongo DB Injection W3af*

Gambar 5.38 menunjukkan adanya error pada kerentanan *captcha* pada *W3af* dikarenakan *HTTP*respon code 500 yang menyatakan untuk memberi akses ijin semua plugin.

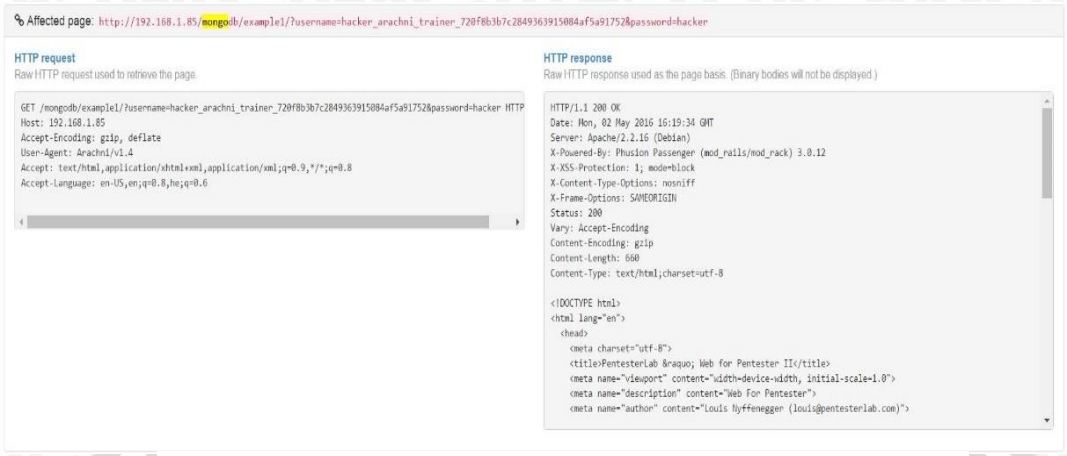
B. *Wapiti*



Gambar 5.39 Hasil Pengujian Otomatis *Mongo DB Injection W3af*

Gambar 5.39 menunjukkan hasil dari pengujian otomatis dengan menggukanan *tool Wapiti*. Adanya hasil identifikasi dari kerentanan *Mongo DB Injection* dengan *HTTPRequest* GET /mongodb/example1/?password=letmein&submit=submit&username=sleep%287%29%231 HTTP/1.1

C. *Arachni*



Gambar 5.40 Hasil Pengujian Otomatis *Mongo DB Injection Arachni*

Gambar 5.39 menunjukkan dari pengujian otomatis dengan menggukanan *tool Arachni*. Adanya hasil identifikasi dari kerentanan *Mongo DB Injecrion* tersebut *tool Arachni* menemukan bahwa halaman yang terpengaruh berisi masukan password,tetapi hasil tersebut tidak dikirim server menggunakan HTTPS.

BAB 6
PENUTUP

6.1 Kesimpulan

Berdasarkan analisa dari hasil pengujian penetrasi pada *challenge* aplikasi *web www.pentesterlab.com* yang telah dilakukan, dapat disimpulkan bahwa :

1. Identifikasi yang dilakukan dengan menggunakan tools w3af ,wapiti, dan arachni dapat dibandingkan dengan menggunakan 5 kali percobaan agar setiap aplikasi web yang di identifikasi dapat hasil yang berbeda.
2. Hasil analisis yang didapat dengan menggunakan tool *w3af, wapiti,* dan *arachni* setelah melakukan 5 kali percobaan memberikan hasil tool *arachni* yang mendapatkan kerentanan pada identifikasi tersebut lebih banyak dari tool w3af dan wapiti.

6.2 Saran

Dalam analisa perbandingan *penetration testing tool* untuk aplikasi *web* masih terdapat banyak tambahan-tambahan yang dapat digunakan untuk melakukan penelitian selanjutnya adalah sebagai berikut :

1. *Penetration testing tool* dapat di tambahkan agar pada proses analisis dapat memberikan hasil yang lebih baik lagi
2. Dapat mengetahui lagi eror-eror dari setiap *tool* penterasi yang di dapat dari hasil *scanning* pada aplikasi *web www.pentesterlab.com*

DAFTARA PUSTAKA

[1] Ahn, Luis von, 2000. *The CAPTHCA* Web Page: http://www.captcha.net.

[2] Bacudio, Aileen G. 2011. *An Overview Of Penetration Testing*.

[3] Bairwa, Sheetal. 2014. Vulnerabilty *Scanner*s : *A Proactive Approach To Asses Web Application Security*

[4] Goel, Jai Narayan 2015. *Vulnerabilty Assessment & Penetration Testing as a Cyber Defence Technology*.

[5] Halfond, William G.J. 2011. *Improving Penetration Testing Through Static And Dynamic Analysis*

[6] Laskos, A.: *Arachni – Web Application Vulnerability Scanning Framework*.2011. https://github.com//*Arachni*

[7] Loo, Frank van der. 2011. *Comparison of penetration testing tool for web aplication*

[8] Mukhopadhyay , Indraneel. 2014. *Web Penetration Testin Using Nessus and Metasploit Tool*

[9] OWASP GUIDE. https://www.owasp.org/index.php/

[10] Riancho, A.: *W3af*. 2011 http:// *W3af*.sourceforge.net.

[11] Singh, Akhand Pratap. 2015. *Penetration Testing and Vulnerability Analysis in Computer Networks*

[12] Surribas, N.: *Wapiti*.2009 http://*Wapiti*.sourceforge.net.

[13] Suter, Rico , 2012. *MongoDB An Introduction and Perfomance Analysis*

[14] UU INFORMASI DAN TRANSAKSI ELEKTRONIK 2008