



PENGUJIAN DAN ANALISIS KEAMANAN *WEBSITE* INSTITUT TEKNOLOGI
PADANG MENGGUNAKAN *ACUNETIX VULNERABILITY SCANNER*

TESIS

Diajukan untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Magister Komputer

AFIF ZIRWAN
201321002

PROGRAM MAGISTER (S2)
PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS PUTRA INDONESIA “YPTK” PADANG

MARET 2022

**PROGRAM MAGISTER (S2)
PROGRAM STUDI TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER**

Tanda Persetujuan Diberikan Kepada

**NAMA : AFIF ZIRWAN
NOBP : 201321002**

**PENGUJIAN DAN ANALISIS KEAMANAN *WEBSITE*
INSTITUT TEKNOLOGI PADANG MENGGUNAKAN *ACUNETIX*
*VULNERABILITY SCANNER***

**Disetujui Untuk Diajukan Pada Ujian Akhir, Sidang Tertutup
Program Magister (S2)
Program Studi Teknik Informatika
Fakultas Ilmu Komputer
Universitas Putra Indonesia “YPTK” Padang**

MENYETUJUI

PEMBIMBING I

PEMBIMBING II

**Prof. Dr. Jufriadif Na'am, S.Kom., M.Kom.
NIDN: 1003026702**

**Dr. Yuhandri, S.Kom., M.Kom.
NIDN: 1015057301**

Telah dinyatakan lulus Ujian Tesis pada Sidang Tertutup Program Magister (S2)
Program Studi Teknik Informatika Fakultas Ilmu Komputer Universitas Putra
Indonesia “YPTK” Padang pada Bulan Maret 2022 dengan hasil Baik.

Padang, Maret 2022

Tim Penguji

Penguji I:

Prof. Dr. Sarjon Defit, S. Kom, M. Sc.
NIDN: 1007087002

Penguji II:

Dr. Ir. H. Sumijan, M.Sc.
NIP: 196605071994031004

Mengesahkan
Dekan Fakultas Ilmu Komputer
Universitas Putra Indonesia “YPTK” Padang

Dr. Yuhandri, S.Kom., M.Kom.
NIDN: 1015057301

TESIS

Diajukan untuk Memenuhi Salah Satu Syarat
Memperoleh Gelar Magister Komputer

“Saya akui karya ini adalah hasil kerja saya sendiri kecuali kutipan dan ringkasan yang masing-masing telah saya jelaskan sumbernya”

Tanda Tangan :

Nama Penulis :

Tanggal :

KATA PENGANTAR

Segala puji hanya milik Allah SWT. Shalawat dan salam selalu tercurahkan kepada Rasulullah SAW. Berkat limpahan dan rahmat-Nya penulis telah mampu menyelesaikan tesis ini dengan baik.

Dalam penyusunan tugas akhir ini, tidak sedikit hambatan yang dihadapi. namun penulis menyadari bahwa kelancaran dalam penyusunan materi ini tidak lain berkat bantuan, dorongan dari berbagai pihak dan penulis ucapkan terima kasih kepada berbagai pihak yang telah membantu kelancaran dalam penyelesaian tugas akhir ini, di antaranya:

1. Ibu **Dr. Zerni Melmusi, M.M., Ak.**, selaku ketua Yayasan Perguruan Tinggi Komputer (YPTK) Padang.
2. Bapak **Prof. Dr. Sarjon Defit, S. Kom, M. Sc.**, selaku Rektor Universitas Putra Indonesia (YPTK) Padang.
3. Bapak **Dr. Yuhandri, S. Kom., M. Kom.** selaku Dekan Fakultas Ilmu Komputer Universitas Putra Indonesia (YPTK) Padang dan pembimbing II dalam penelitian ini.
4. Bapak **Ir. Gunadi Widi Nurcahyo, MSc., PhD.** selaku Ketua Program Studi Magister Ilmu Komputer Universitas Putra Indonesia (YPTK) Padang
5. Bapak **Prof. Dr. Jufriadif Na'am, S.Kom., M.Kom.** selaku pembimbing I dalam penelitian ini.

7. Kampus Institut Teknologi Padang yang telah mengizinkan penulis untuk melakukan penelitian untuk menyelesaikan tesis ini.

Penulis menyadari bahwa dalam penulisan tesis ini masih terdapat banyak kekurangan dan kelemahannya, untuk itu penulis sangat mengharapkan masukan berupa kritik dan juga saran yang bersifat membangun. Penulis berharap semoga tesis ini dapat bermanfaat bagi pembaca dan semua pihak yang berkepentingan dengan tesis ini.

Padang, Maret 2022
Penulis,

Afif Zirwan

ABSTRAK

Pandemi Covid-19 secara tidak langsung mempercepat Revolusi Industri 4.0 dimana teknologi menjadi poin penting dalam pergerakan industri. Perubahan yang dipaksa cepat, kadang kala melupakan beberapa pengembang aplikasi untuk melakukan pengujian terhadap produk yang akan diimplementasikan, karena dituntut cepat untuk mengimbangi perubahan tersebut. Institut Teknologi Padang (ITP) juga melakukan perubahan cepat terhadap *website* resminya, untuk menyediakan informasi aktual dari kegiatan kampus dimasa pandemi ini. Penelitian ini bertujuan untuk melakukan pengujian dan analisa sejauh mana keamanan *website* ITP dan memberikan saran pemecahan masalah dari hasil analisa. Pengujian dilakukan dengan menggunakan *tools* Acunetix Vulnerability Scanner. Metode yang digunakan adalah analisa deskriptif, yaitu data yang diperoleh disajikan dalam bentuk tabel, sehingga memberikan kejelasan dari hasil analisa yang dilakukan. Dari data yang diperoleh, *website* ITP berada pada *threat level* 3 yang termasuk kategori *High* dengan 119 *alert* atau celah yang ditemukan yang terdiri dari 94 pada *level high* dan 25 pada *level medium*. Berdasarkan analisa, perbaikan dan pengujian yang dilakukan pada penelitian ini terhadap *website* ITP, menghasilkan *threat level* sudah pada *level* 1, dimana pada *level high* jumlah celah sudah menjadi 0 dan *medium* juga sudah menjadi 0 yang dapat disimpulkan *website* ITP sudah tergolong aman dari celah keamanan.

Kata Kunci : Revolusi 4.0, Keamanan, Algoritma, Infrastruktur, Acunetix Vulnerability Scanner.

ABSTRACT

The Covid-19 pandemic has indirectly accelerated the Industrial Revolution 4.0 where technology has become an important point in industrial movement. Changes that are forced to be fast, sometimes escape some application developers to test the product to be implemented, because they are required to quickly compensate for these changes. The Padang Institute of Technology (ITP) also made rapid changes to its official website, to provide actual information on campus activities during this pandemic. This study aims to test and analyze the extent of the security of the ITP website and provide troubleshooting suggestions from the results of the analysis. Testing is done using the Acunetix Vulnerability Scanner tool. The method used is descriptive analysis, namely the data obtained are presented in tabular form, thus providing clarity of the results of the analysis carried out. From the data obtained, the ITP website is at threat level 3 which is included in the High category with 119 alerts or gaps found consisting of 94 at the high level and 25 at the medium level. Based on the analysis, improvement and testing carried out in this study on the ITP website, the resulting threat level is already at level 1, where at the high level the number of gaps has become 0 (zero) and the medium has also become 0 (zero) which can be concluded that the ITP website is classified as safe from security holes.

Keyword : Revolution 4.0, Security, Algorithms, Infrastructure, Acunetix Vulnerability Scanner.

DAFTAR ISI

BAB	JUDUL	HALAMAN
	ABSTRAK	ix
	ABSTRACT.....	x
	DAFTAR ISI	xi
	DAFTAR TABEL	xiii
	DAFTAR GAMBAR	xvi
 I	 PENDAHULUAN	
	1.1 Latar Belakang	1
	1.2 Perumusan Masalah.....	3
	1.3 Batasan Masalah.....	3
	1.4 Tujuan Penelitian.....	3
	1.5 Manfaat Penelitian.....	4
	1.6 Sistematika Penulisan.....	4
 II	 TINJAUAN PUSTAKA	
	2.1 <i>Vulnerability</i>	6
	2.2 <i>Vulnerability Assessment</i>	6
	2.3 Jenis-Jenis <i>Vulnerability Assessments</i>	8
	2.4 Pengertian Keamanan Jaringan	9
	2.5 Serangan Dalam Kemanan Komputer	10
	2.6 Acunetix WVS	18
	2.7 Penelitian Terdahulu	19

III	METODOLOGI PENELITIAN	
	3.1 Pendahuluan	23
	3.2 Kerangka Kerja Penelitian	24
IV	ANALISA DAN PERANCANGAN	
	4.1 Tahapan Analisa dan Perancangan.....	28
	4.2 Data	28
	4.3 Menganalisa Sistem.....	35
	4.3.1 Laporan Hasil Perbaikan	40
	4.4 Perancangan	42
	4.4.1 Desain <i>Output</i>	42
	4.4.2 Desain <i>Input</i>	45
	4.4.3 Desain <i>Database</i>	46
V	IMPLEMENTASI DAN HASIL	
	5.1 Implementasi	49
	5.2 Analisa dan Perbaikan.....	49
	5.3 Hasil	70
	5.3.1 Implementasi Sistem	72
VI	KESIMPULAN	
	6.1 Kesimpulan.....	83
	6.2 Saran.....	83
	DAFTAR PUSTAKA.....	84

DAFTAR TABEL

TABEL	JUDUL	HALAMAN
2.1	Gerbang Logika OR	13
2.2	Penelitian Terdahulu Tentang <i>Vulnerability</i>	19
4.1	Data Penelitian	33
4.2	Perbaikan <i>Bug XSS</i>	39
4.3	Laporan Perbaikan	40
4.4	Rancangan Tabel <i>Vulnerability</i>	47
4.5	Rancangan Tabel <i>Case</i>	47
5.1	Perbaikan <i>Bug XSS</i>	53
5.2	Perbaikan <i>Bug Application Error Messages</i>	56
5.3	Perbaikan <i>Bug Development Configuration File</i>	59
5.4	Perbaikan <i>Bug Error Mesaage on Page</i>	62
5.5	Perbaikan <i>Bug Host Header Attack</i>	65
5.6	Perbaikan <i>Bug HTML Without CSRF Protection</i>	69
5.7	Perbandingan Data Setelah Perbaikan	71
5.8	Tabel Kesimpulan Perbaikan	72

DAFTAR GAMBAR

TABEL	JUDUL	HALAMAN
2.1	<i>Ilustrasi Man In The Middle</i>	17
2.2	<i>Type MITM Attack</i>	17
3.1	Kerangka Kerja Penelitian	24
4.1	Bagan Alir Analisa dan Perancangan.....	28
4.2	Halaman <i>Website</i> Institut Teknologi Padang	29
4.3	<i>New Website Scan</i>	29
4.4	<i>Scan Type Website</i>	30
4.5	<i>Options Scanning Website</i>	31
4.6	<i>Target Scanning Website</i>	31
4.7	<i>Login Scanning Website</i>	32
4.8	Tahapan Terakhir <i>Initialize Scanning</i>	32
4.9	<i>Flowchart Vulnerability Assesment</i>	36
4.10	<i>File head.php</i>	37
4.11	Temuan <i>Bug</i> Pada <i>File Head.php</i>	37
4.12	Fungsi <i>Filtering XSS</i>	38
4.13	<i>Global XSS Filtering</i>	38
4.14	<i>Global XSS Filtering</i>	38
4.15	Hasil Perbaikan XSS	39
4. 16	Desain Halaman Utama	42
4.17	Tabel Data Pencarian	43
4.18	Desain Halaman Detail	44
4.19	Input Data Master <i>Vulnerability</i>	45
4.20	Input Data <i>Bug's</i>	46

5.1	Hasil Scanning <i>Website</i>	50
5.2	<i>File</i> head.php.....	51
5.3	Temuan Bug Pada <i>File</i> Head.php	51
5.4	Fungsi <i>Filtering</i> XSS	52
5.5	Global XSS <i>Filtering</i>	52
5.6	Global XSS <i>Filtering</i>	52
5.7	Hasil Perbaikan XSS	53
5.8	<i>Application Error Messages</i>	54
5.9	<i>File</i> index.php	55
5.10	Konfigurasi <i>Environment</i> Aplikasi	56
5.11	Hasil Perbaikan <i>Application Error Message</i>	56
5.12	<i>Develoment Configuration File</i>	57
5.13	Lokasi File composer.json	57
5.14	<i>Permission File</i> composer.json	58
5.15	Pengaturan <i>Permission File</i>	59
5.16	Hasil Perbaikan <i>Development Configuration File</i>	60
5.17	<i>Lokasi File Controller Admin</i>	61
5.18	<i>File Controller Admin</i>	61
5.19	Penambahan <i>Controller Index</i>	62
5.20	Hasil Perbaikan <i>Error Message On Page</i>	63
5.21	<i>File</i> head.php.....	64
5.22	<i>Host Header Attack</i>	64
5.23	Konfigurasi <i>Header</i> Pada NGINX	65
5.24	Konfigurasi <i>Header</i> Pada Aplikasi	65
5.25	Hasil Perbaikan <i>Host Header Attack</i>	66
5.26	HTML <i>Form Without CSRF Protection</i>	66
5.27	<i>File</i> v_home.php	67
5.28	<i>File</i> v_search.php	68
5.29	<i>File</i> v_search.php	68
5.30	Aktivasi Fungsi <i>CSRF Protection</i>	68
5.31	Mengganti Penulisan <i>Tag Form</i> File v_home.php.....	69
5.32	Mengganti Penulisan <i>Tag Form</i> File v_search.php	69
5.33	Hasil Perbaikan HTML <i>form without CSRF Protection</i>	70
5.34	Hasil <i>Scanning</i> Ulang Setelah Diperbaiki	70

5.35	Hasil <i>Scanning</i> Setelah Perbaikan.....	71
5.36	Aplikasi <i>Knowledge Transfer</i>	72
5.37	Detail Informasi Perbaikan XSS	72
5.38	Detail Informasi Perbaikan <i>Application Error Message</i>	73
5.39	Detail Informasi Perbaikan <i>Development Configuration File</i>	73
5.40	Detail Informasi Perbaikan <i>Error Message In Page</i>	74
5.41	Detail Informasi Perbaikan <i>Host Header Attack</i>	74
5.42	Detail Informasi Perbaikan <i>HTML Without CSRF Protection</i>	75

BAB I

PENDAHULUAN

1.1 Latar Belakang

Industri 4.0 atau revolusi industri keempat merupakan istilah yang umum digunakan untuk tingkatan perkembangan industri teknologi di dunia. Pada tingkatan keempat ini, dunia memang fokus kepada teknologi-teknologi yang bersifat digital, oleh karena itu teknologi informasi sangat-sangat menjadi tumpuan untuk industri yang bertujuan untuk mempermudah dan mempercepat proses-proses untuk membuat produk.

Pandemi *covid-19* secara tidak langsung mempercepat revolusi industri 4.0, dimana banyak kegiatan dilakukan tidak dengan bertatap muka secara langsung, melainkan melalui pertemuan *online* melalui *gadget* masing-masing. Begitu juga dengan pencarian informasi, semua dilakukan dengan cepat melalui internet. Dengan faktor tersebut, banyak industri dan instansi berpacu untuk melakukan pembaharuan terhadap layanan informasi mereka yang agar pengiriman data dan informasi meningkat. Disamping keuntungan tersebut, tingkat resiko dan ancaman penyalahgunaan teknologi informasi juga menjadi semakin meningkat (Bustami dan Bahri, 2020).

Perubahan yang sangat cepat, kadang kala melupakan *developer* dalam melakukan pengujian terhadap aplikasi yang dibangun. Pengujian merupakan proses yang sangat penting didalam pengembangan perangkat lunak yang berkualitas tinggi, karena dari beberapa kesalahan yang dianggap tidak penting beresiko sangat berbahaya hal ini menjadi celah (*vulnerability*) bagi *attacker* untuk memanfaatkan informasi yang di curi melalui serangan kepada aplikasi. Kebutuhan akan *vulnerability assessment* selama ini biasanya dipandang sebelah mata, karen hanya dianggap sebagai kegiatan formalitas dan sedikit orang yang melakukan kegiatan ini (Goel dan Mehtre, 2015). Salah satu sistem yang umumnyamenjadi sasaran *hacker*

dan *cracker* adalah aplikasi berbasis *website*. Hal tersebut dikarenakan pemanfaatan aplikasi mengalami pertumbuhan yang sangat pesat saat ini (Al Fajar, 2020). Serangan yang dilakukan dapat berupa *Cross Site Scripting* (XSS), *Cross Site Request Forgeri* (CSRF), *SQL injection* dan lain sebagainya (Riadi et al, 2020). Serangan *SQL injection* merupakan sebuah aksi *hacking* yang dilakukan di aplikasi *client* dengan cara memodifikasi perintah SQL yang ada di memori aplikasi *client* dan mengeksploitasi aplikasi menggunakanh basis data untuk penyimpanan data (Ade Bastian et al, 2020). Penelitian lainnya juga mendapatkan hasil jenis serangan seperti *Local File Inclusion* (LFI) dan parameter tampering (Gupta et al, 2020).

Vulnerability testing banyak digunakan untuk meningkatkan kesadaran tentang pentingnya keamanan informasi (Riadi et al, 2021). Penilaian kerentanan bisa mendeteksi hampir semua celah kerentanan yang biasanya terjadi pada sebuah sistem (Pohan, 2021). Beberapa faktor dari dari celah keamanan bisa terjadi karena kurangnya sistem pengamanan *website* dan kekeliruan *programmer* ketika melakukan *coding* (Syarifudin, 2018).

Salah satu cara untuk melakukan evaluasi keamanan *website* menggunakan perangkat lunak yang khusus dirancang untuk mengetahui kerentanan yang ada pada suatu sistem yaitu *acunetix vulnerability scanner* (Mayasari et al, 2020), Pentest-tools.com, *acunetik WVS*, *vulnerability scanner*, OWASP ZAP (Irawadi Alwi dan Umar, 2020). Pengujian ini juga tidak terbatas pada aplikasi yang di kustom sendiri, aplikasi CMS (*Content Management System*) seperti OJS juga menjadi target uji (Wibowo dan Purwo Wicaksono, 2019). Pengujian *vulnerability* dilakukan untuk pengukuran atau *assessment* yang mutlak dilakukan untuk mendapatkan peningkatan kualitas dan salah satu cara pengukuran terhadap keamanan sistem. Hasil dari *assesment* menjadi bahan pertimbangan bagi *developer* untuk mengambil tindakan pencegahan dan mengetahui cara kerja dari *attackers* (Orisa dan Ardita, 2021).

Begitu juga dengan Institut Teknologi Padang (ITP), baru saja meluncurkan *website* resmi versi ke 3. Alasan ITP meluncurkan versi ialah untuk menutupi celah keamaan yang terjadi pada versi 2 dimana pada versi ini masih menggunakan teknolgi *scripting* yang masih lama dimana rentan menjadi target serangan oleh *attacker*. Kerentanan pada keamanan *website* merupakan hal yang harus diperhatikan bagi setiap institusi agar terhindar dari tindakan kejahatan di dunia

maya (*cyber crime*) (Irawadi Alwi dan Umar, 2020). Oleh karena itu pengujian *vulnerability* penting dilakukan, yang hasilnya bukan menggaransi sistem akan bebas dari resiko serangan, tetapi dapat meminimalisir serangan yang dapat disalahgunakan, karena untuk menjelajahi semua aspek harus dilakukan pengujian tingkat lanjut (T dan Sasikala D, 2019).

Berdasarkan paparan diatas, penulis ingin melakukan analisa dan pengujian serta terhadap versi 3 dari *website* ini, dengan tujuan agar dapat memberikan saran perbaikan dan peningkatan dari kemanan *webisite* ini. Berdasarkan uraian diatas, penulis mengangkat permasalahan diatas sebagai judul penelitian yang berjudul **“Pengujian dan Analisis Keamanan Website Institut Teknologi Padang Menggunakan Acunetix Vulnerability Scanner”**.

1.2 Perumusan Masalah

Berdasarkan permasalahan yang ada, agar tesis ini sesuai dengan tujuan yang ingin dicapai, maka penulis merumuskan beberapa permasalahan sebagai berikut :

1. Bagaimana melakukan pengujian kemanan terhadap *website* ITP dengan menggunakan *tools* Acunetix WVS.
2. Bagaimana menganalisis celah keamanan yang ditemukan.
3. Bagaimana memberikan solusi dari permasalahan yang ditemukan.

1.3 Batasan Masalah

Agar pembahahasan pada penelitian ini tidak menyimpang maka penulis membatasi ruang lingkup objek penelitian. Adapun ruang lingkup penelitian antara lain:

1. Penelitian ini digunakan untuk mengetahui celah keamanan dari *website* ITP.
2. Penelitian ini menggunakan *tools* Acunetix WVS untuk pengujian,
3. Penelitian dilakukan dengan prinsip keamanan sistem informasi yaitu *confidentiality* (Kerahasiaan) dan *integrity* (Kesatuan).

1.4 Tujuan Penelitian

Tujuan yang ingin diperoleh dari penelitian ini agar lebih bermanfaat kedepannya adalah:

1. Melakukan pengujian terhadap keamanan *website* ITP dengan menggunakan *tools* Acunetix WVS.
2. Melakukan analisa terhadap kewanaman yang ditemukan.
3. memberikan saran peningkatan keamanan dan perbaikan *website* ITP.

1.5 Manfaat Penelitian

Penelitian ini diharapkan memberikan manfaat kedepannya, yang beberapa diantaranya adalah:

1. ITP dapat mengetahui celah kewanaman yang terdapat pada *website*.
2. Meningkatkan sistem keamanan dari *website* ITP.
3. Dapat menjadi konsultan keamanan sistem informasi bagi ITP.

1.6 Sistematika Penulisan

Sistematika yang digunakan dalam penyusunan tesis ini adalah sebagai berikut:

Bab I PENDAHULUAN

Bab ini berisikan tentang latar belakang, perumusan masalah, batasan masalah, tujuan penelitian, manfaat penelitian, dan sistematika penelitian

Bab II TINJAUAN PUSTAKA

Bab Ini berisikan tentang *vulnerability*, *vulnerability assesment*, Serangan Dalam Keamanan Komputer dan *tools* Acunetix WVS.

Bab III METODOLOGI PENELITIAN

Bab ini menjelaskan jenis penelitian yang dilakukan, pendekatan yang digunakan, sumber data, lokasi penelitian, metode dan alat pengumpulan data serta teknik pengolahan data dan analisa.

Bab IV ANALISA DAN PERANCANGAN

Bab ini berisi tentang analisa sistem yang akan diuji, bagaimana *tools* Acunetix WVS bekerja dan mengelompokkan hasil *assesment* dan cara pengujian serta analisa perbaikan dari hasil *assesment*.

Bab V IMPLEMENTASI DAN HASIL

Bab ini berisi tentang implementasi perancangan yang telah dilakukan serta detail perbaikan dan hasil dari pengujian yang dilakukan.

Bab VI KESIMPULAN DAN SARAN

Bab ini berisikan kesimpulan dari penyusunan tesis serta saran-saran untuk pengembangan selanjutnya.

BAB II

TINJAUAN PUSTAKA

2.1 *Vulnerability*

Celah keamanan (*vulnerability*) sistem jaringan komputer merupakan sebuah kelemahan, kekurangan atau celah pada sistem, yang dapat dimanfaatkan oleh satu atau lebih dari penyerang untuk melakukan serangan yang dapat membahayakan kerahasiaan, integritas, atau ketersediaan suatu sistem (Retna Mulya & Tarigan, 2018). *Vulnerability* adalah suatu kelemahan yang mengancam nilai *integrity*, *confidentiality*, dan *availability* dari suatu aset (Irawadi Alwi & Umar, 2020). *Vulnerability Assessment* merupakan bagian dari *risk assessment* yang terdiri dari *risk analysis*, *policy development*, *training and implementation*, dan *vulnerability assessment* dan *penetration testing* (Wibowo & Purwo Wicaksono, 2019).

2.2 *Vulnerability Assessment*

Vulnerability Assessment (VA) adalah proses pemindaian sistem atau *software* dan jaringan untuk mengetahui kelemahan dan celah yang ada, celah ini memberikan *backdoor* ke penyerang untuk menyerang korban. sistem memiliki *access control vulnerability*, *boundary condition vulnerability*, *input validation vulnerability*, *authentication vulnerabilities*, *configuration weakness vulnerabilities*, dan *exception handling vulnerabilities* (Goel & Mehtre, 2015).

Dalam tataran konsep proses VA merupakan proses yang sangat kompleks karena melibatkan seluruh komponen dalam TI. Dalam tataran teknis proses yang beresiko mengingat adanya peluang untuk merusak atau mengganggu kinerja sistem yang berlangsung. Proses VA secara garis besar dapat dibagi dalam tiga tahapan (Priandoyo, 2006):

1. Penentuan Batasan Proyek

Penentuan batasan proyek merupakan tahapan yang paling kritikal, penentuan batasan ini harus diikuti oleh semua pihak yang terkait. Mulai dari *manager* TI, *manager* keuangan / pengguna akhir, hingga operator itu sendiri, bila proses *assessment* melibatkan pihak ketiga maka tentunya selain harus dihadiri assesor yang sudah melakukan survei awal sebelumnya. Pembatasan proyek ini diperlukan agar VA tidak terlalu luas sehingga merambah ke hal-hal lain yang kurang signifikan atau agar VA tidak terlampaui sempit sehingga melewatkan hal-hal yang lebih kritikal. Pembatasan proyek ini juga meliputi besarnya dana yang akan diperlukan, hingga jumlah tenaga yang akan mengerjakannya.

2. Pelaksanaan *Assessment*

Setelah batasan proyek ditentukan maka proses VA masuk dalam tahap pelaksanaan. Dalam tahapan ini diasumsikan semua kebutuhan *assessment* telah didefinisikan sebelumnya. Jika perusahaan menginginkan pengukuran akan SOP (standar operasional dan prosedur) tentunya sudah ada standar penerapan dilapangan yang akan dijadikan acuan. Begitu juga bila pengukuran terhadap standar keamanan sistem maka sudah ada acuan bagaimana standar keamanan sistem yang diterapkan dilapangan. Data-data acuan ini tersedia dalam ranah publik, dikeluarkan oleh vendor produk maupun dikeluarkan oleh organisasi independen yang melakukan rating terhadap resiko keamanan.

3. Pelaporan Akhir

Hasil temuan dilapangan dengan alat bantu VA *scanning* tidak serta merta ditindaklanjuti oleh *client*, tentunya diperlukan evaluasi dan pemeriksaan kembali dari sisi penggunaanya. Evaluasi ini perlu melihat sejauh mana kebutuhan penggunaanya terhadap sistem yang dinilai rawan, adakah kontrol pengganti (*compensating control*) bila sistem tersebut mengalami gangguan. Assesor pun harus melakukan kategorisasi terhadap hasil temuannya. Paling tidak perlu ada tiga kategori seperti resiko tinggi, sedang dan rendah yang penentuan tingkat kerawanan ini merupakan hasil diskusi antara assesor dan *client* sebelumnya. Matriks tingkat kerawanan inilah yang akan diajukan. Laporan dibuat dalam dua versi, yang pertama adalah versi lengkap yang akan diberikan pada *security officer* ataupun sistem *administrator* di perusahaan tersebut. Sedangkan yang kedua dalam versi yang lebih ringkas diberikan pada

pimpinan di perusahaan yang berisi masukan dalam tingkat kebijaksanaan dan strategi.

Proses VA ini dilaksanakan secara terkendali dimana tahapan yang satu tidak bisa mendahului tahapan yang lain. Setiap tahap yang dilakukan harus didasari atas koordinasi setiap pihak yang terkait dalam proses bisnis.

2.3 Jenis-Jenis *Vulnerability Assessments*

Dalam penerapannya *vulnerability assesment*, memiliki 5 jenis assesment dimana tiap pengaplikasian-nya ini dikelompokkan berdasarkan objek yang akan di scan, yaitu:

1. *Network-Based Scans*

Digunakan untuk mengidentifikasi kemungkinan serangan keamanan jaringan. Jenis peninjauan ini juga dapat mendeteksi sistem yang rentan pada jaringan kabel atau nirkabel.

2. *Host-Based Scans*

Digunakan untuk mencari dan mengidentifikasi kerentanan di server, workstation atau *host* jaringan lainnya. Jenis peninjauan ini biasanya akan memeriksa *port* dan layanan yang mungkin terlihat oleh *network-based scans*, tetapi penilaian ini menawarkan tingkat visibilitas yang lebih besar pada pengaturan konfigurasi dan menampung riwayat sistem yang telah diamati.

3. *Wireless Network Scans*

Biasanya fokus pada titik-titik serangan infrastruktur jaringan nirkabel. Selain mengidentifikasi jalur akses yang terindikasi, pengamatan jaringan nirkabel juga dapat memvalidasi bahwa jaringan perusahaan terkonfigurasi dengan aman.

4. *Application Scans*

Dapat digunakan untuk menguji situs web dan mendeteksi kerentanan perangkat lunak serta kesalahan dalam konfigurasi aplikasi jaringan atau web.

5. *Database Scans*

Dapat digunakan untuk mengidentifikasi titik lemah dalam basis data sehingga dapat mencegah serangan berbahaya, seperti serangan injeksi SQL.

2.4 Pengertian Keamanan Jaringan

Keamanan jaringan merupakan sebuah upaya untuk melakukan kendali terhadap akses sumberdaya jaringan, akses jaringan dikontrol supaya hanya bisa diakses oleh siapa saja yang berhak dan menghalangi dari yang tidak berhak. Menurut Garfinkel Keamanan jaringan merupakan pakar dalam bidang *security*. Keamanan jaringan terdiri dari beberapa aspek yaitu, *privacy*, *integrity*, *authentication* dan *availability* untuk penjelasannya sebagai berikut:

1. *Privacy* atau *confidentiality*

Privacy atau kerahasiaan informasi, inti dari aspek ini adalah bagaimana menjaga kerahasiaan dari informasi agar tidak dapat dilihat atau diakses oleh orang yang tidak memiliki hak sama sekali.

2. *Integrity*

Aspek *integrity* atau biasa juga disebut sebagai integritas mencakup keutuhan atau orisinalitas informasi. maksud dari aspek ini adalah bagaimana informasi agar tetap utuh. Sebuah informasi tidak boleh diubah, baik ditambah maupun dikurangi informasi tersebut kecuali atas izin dari pihak yang memiliki kewenangan.

3. *Authentication*

Aspek *authentication*, mensyaratkan ketika pengiriman suatu informasi bisa untuk diidentifikasi dengan baik dan benar serta memberikan jaminan bahwa identitas yang diperoleh adalah benar atau tidak palsu.

Berkaitan dengan aspek *security*, seorang pakar *security* yang bernama W. Stallings. Juga mengemukakan pendapatnya dengan adanya beberapa kemungkinan serangan yang terjadi terhadap keamanan suatu sistem informasi, yaitu *interruption*, *interception*, *modification* dan *fabrication*. Tiga serangan ini digolongkan kedalam serangan aktif, sedangkan *interception* bisa kita golongkan kedalam bentuk serangan

pasif. Dalam jenis serangan ini penyerang akan mengirim suatu aliran data ke salah satu atau kedua kelompok yang terlibat komunikasi dan akan menotong aliran data

2.5 Serangan Dalam Kemanan Komputer

Keamanan komputer hal yang sangat penting, seiring dengan pentingnya informasi yang ada pada jaringan. serangan awal pada komputer disebut dengan *portscanning*. Keberhasilan penyerang dalam melakukan *port scanning* akan memberikan peluang pada penyerang untuk melakukan serangan pada jaringan komputer. Ada beberapa serangan terhadap website menurut (Efyz Zam, 2011) :

1. *Local File Inclusion (LFI) & Remote File Inclusion (RFI)*

Local File Inclusion (LFI) adalah sebuah celah pada sebuah situs *web* sehingga memungkinkan seseorang bisa mengakses semua *file* di dalam *server* dengan hanya melalui URL. *Local File Inclusion (LFI)* berkaitan erat dengan traversal direktori yang dilakukan secara lokal dari server web. LFI biasanya terjadi karena variabel selama pembuatan situs web buruk dikendalikan sehingga memungkinkan penyerang untuk mengakses folder secara langsung. Fungsi yang dapat mengarah pada *local file inclusion* pada situs web sebagai berikut;

- a. `include ()`;
- b. `include_once ()`;
- c. `require ()`;
- d. `require_once ()`;

Dengan kondisi dalam konfigurasi php di server:

- a. `allow_url_include = on`
- b. `allow_url_fopen = on`
- c. `magic_quotes_gpc = of`

Pengembangan metode LFI adalah *remote connect-back shell*, di mana penyerang mendapatkan shell dari server target dan penyerang dapat melakukan berbagai perintah linux pada sistem yang ada. Sedangkan *Remote*

File inclusion (RFI) adalah sebuah celah dimana sebuah situs mengizinkan seseorang *meng-include-kan file* dari luar server. *Remote file inclusion* merupakan jenis serangan yang memungkinkan penyerang untuk menyisipkan *file* dari luar server. Serangan-serangan ini memiliki bahaya dampak karena *file* disisipkan dalam bentuk *shell*. dengan adanya *shell* ini seorang *hacker* dapat memperoleh akses ke sistem yang melakukan berbagai aktivitas seperti melihat direktori, mengubah halaman (*defacing*) atau mencuri informasi sensitif dari *file* yang ada dalam sistem.

Biasanya penyerang akan mencoba menggunakan *remote file inclusion* pertama untuk menembus di situs web, tetapi jika penyerang mengetahui bahwa bentuk penetrasi yang dilakukan gagal (biasanya karena "allow_url_include = off" in the php.ini), akan mengejar serangan dengan metode di mana halaman penyertaan file lokal disisipkan pada server yang sama. Bentuk *file inclusion* serangan berisiko tinggi, karena penyerang dapat memperoleh akses shell, dan pada akhirnya mempengaruhi Eksploitasi lokal di mana penyerang mendapatkan hak akses penuh ke sistem.

2. *Cross-Site Scripting* (XSS)

Serangan *Cross Site Scripting* (XSS) merupakan jenis serangan injeksi dengan mengirimkan kode-kode *script* berbahaya dalam bentuk *script* ke sisi browser pada pengguna akhir yang berbeda. Terkadang, dari pada menyerang sebuah server yang lebih sulit, seorang *hacker* bisa saja memanfaatkan kelemahan yang ada pada sisi *client*. Lagipula, aksi *hacking* yang satu ini agak sulit dideteksi karena bekerja dari sisi *client*. *Cross site scripting* yang disingkat XSS, bukan CSS, karena CSS digunakan untuk istilah *cascading style sheets* yang merupakan salah satu bahasa pemrograman *web* untuk mengendalikan beberapa komponen dalam sebuah situs *web* sehingga akan lebih terstruktur dan seragam yang dipakai untuk memformat tampilan *web* yang dibuat dengan bahasa HTML dan XHTML. Dengan XSS, serangan dilakukan dengan cara menginjeksi/memasukan script ke dalam situs *web* melalui sebuah *browser*. Aksi XSS ini adalah dengan memanfaatkan metode HTTP (*Hypertext Transfer Protocol*) GET atau HTTP POST.

3. *Phising*

Pada teknik *hacking* yang satu ini, tekniknya adalah berusaha membuat seseorang mengunjungi situs yang salah sehingga memberikan informasi rahasia berupa *username* dan *password* maupun hal lainnya. Umumnya pelaku membuat situs yang memiliki nama domain mirip dengan aslinya, Istilah *phising* identik dengan *web spoofing* dan *DNS spoofing*. Teknik *phising* ini juga dikenal dengan sebutan *fake login*, dimana seorang *login* di halaman yang bukan sebenarnya.

Dari definisi *phising* dapat diketahui cara kerja *phising* yang dilakukan untuk menjebak korban oleh sang penjebak (*phisher*). Sehingga mendapatkan informasi rahasia *user* dengan cara menggunakan email dan situs web palsu yang tampilannya menyerupai tampilan asli atau resmi web sebenarnya. Informasi yang didapat atau dicari oleh *phiser* berupa *password account* atau nomor kartu kredit korban. Penjebak (*phisher*) menggunakan email, *banner* atau *pop-up* window untuk menjebak *user* agar mengarahkan ke situs web palsu (*fake webpage*), dimana *user* diminta untuk memberikan informasi pribadinya. Disinilah *phisher* memanfaatkan kecerobohan dan ketidak telitian *user* dalam web palsu tersebut untuk mendapatkan informasi. Berikut ini adalah aspek-aspek ancaman yang terinfeksi oleh *phising*:

a. Manipulasi *Link*

Sebagian teknik *phising* menggunakan manipulasi link sehingga yang terlihat seperti alamat dari institusi yang asli. URL yang salah ejaannya atau penggunaan *subdomain* adalah trik umum digunakan oleh *phisher*.

b. *Filter Evasion*

Phisher telah menggunakan gambar (bukan teks) sehingga mengecoh pengguna sehingga menyerahkan informasi pribadinya. Ini adalah alasan Gmail atau Yahoo akan mematikan gambar secara default untuk email yang masuk. Untuk membuat e-mail *phising* tampak lebih asli, para *phisher/scammer* melakukan berbagai cara diantaranya sebagai berikut:

- 1) Sebuah link yang dihubungkan ke halaman web yang sah, tetapi sebenarnya membawa anda ke sebuah laman web *phising*

2) Atau mungkin *pop-up* yang tampak persis seperti halaman resmi.

4. SQL Injection

SQL *injection* merupakan sebuah aksi *hacking* yang dilakukan di aplikasi *client* dengan cara memodifikasi perintah atau *syntax* SQL. Serangan injeksi SQL menargetkan aplikasi web interaktif yang menggunakan layanan *database*. Aplikasi tersebut menerima masukan pengguna, seperti bidang formulir, dan kemudian menyertakan ini input dalam permintaan *database*, biasanya pernyataan SQL. Injeksi SQL, penyerang memberikan masukan pengguna yang dihasilkan dalam permintaan *database* yang berbeda dari yang dimaksudkan oleh pemrogram aplikasi. Artinya, interpretasi dari input pengguna sebagai bagian dari *syntax* SQL yang lebih besar, menghasilkan *syntax* SQL dari bentuk yang berbeda dari yang dimaksudkan semula, terdapat dua jenis SQL *injection* :

- a. *Blind SQL Injection*, teknik ini dilakukan dengan memasukkan *syntax*/perintah SQL pada sebuah *web* yang memiliki *vulnerability* (kerentanan/celah keamanan) untuk melihat isi dari *database* SQL tersebut.
- b. *Advance SQL Injection*, merupakan teknik tingkat lanjut buat SQL *Injection*, dimana tidak hanya bisa mengakses *database*, tetapi bisa membuat *shell* atau pun *backdoor* pada *site* target. Pada teknik ini, *hacker* akan mencoba melakukan serangan SQL *Injection* dan memasang *shell* di dalam situs menggunakan *syntax* SQL.

SQL *injection* adalah jenis serangan keamanan dimana input pengguna jahat diurai sebagai bagian dari pernyataan SQL dikirim ke *database* yang mendasarinya. Banyak variasi yang ada, tetapi yang utamanya adalah bahwa pengguna memanipulasi pengetahuan tentang bahasa *query* untuk mengubah permintaan *database*. Tujuan dari serangan ini adalah untuk *query database* dengan cara yang bukan maksud dari *programmer* aplikasi.

Pada *SQL injection*, penggunaan gerbang logika merupakan dasar sederhana dari sintak yang digunakan, salah satu gerbang logika yang dipakai untuk *SQL injection* adalah OR, seperti berikut:

Tabel 2. 1 Gerbang Logika OR

A	B	A OR B
0	0	0
1	1	1
1	0	1
1	1	1

Dengan menggunakan gerbang logika OR maka sintak yang awalnya terkunci dengan logika AND akan lepas karena kondisi or yang dimana salah satu parameter saja benar atau bernilai *true* maka keluaran dari sintak SQL akan bernilai *true*.

Teknik *SQL injection* terdiri dari beberapa diantaranya sebagai berikut:

- a. *Tautologies*, Salah satu metode untuk mendapatkan akses yang tidak sah ke data adalah dengan memasukkan tautologi ke dalam *query* pada SQL, jika di mana klausa pernyataan *SELECT* atau *UPDATE* *disjuncted* dengan tautologi, maka setiap baris dalam tabel *database* adalah termasuk dalam set hasil.
- b. *UNION Query*, merupakan perintah SQL yang memungkinkan dua *query* untuk bergabung dan dikembalikan sebagai satu hasil yang ditetapkan. Misalnya, *SELECT* kolom1, kolom2, kolom3 dari tabel1 *UNION SELECT* kolom4, kolom5, kolom6 dari tabel2 akan mengembalikan satu set hasil yang terdiri dari hasil kedua *query*.
- c. *Additional Statements*, merupakan pendekatan yang sama seperti menggunakan *UNION* tetapi *query* pertama dengan *query* baru tidak mengembalikan penambahan hasil, pertanyaan-pertanyaan baru ini

sering melakukan tindakan tertentu pada *database* yang dapat memiliki konsekuensi. Microsoft SQL Server menyediakan dua tingkat *system* prosedur tersimpan yang dirancang untuk membantu *administrator*. Sehingga tidak sengaja akan memberikan fitur yang kuat untuk penyerang, yang pertama adalah perintah `xp cmdshell` (string) fungsi ini dapat *UNION 'd* dengan *query* lainnya, pada dasarnya bahwa perintah yang diteruskan ke fungsi akan dieksekusi pada server sebagai perintah shell.

- d. *Using Comments*, SQL mendukung komentar dalam *query*, sebagian besar implementasi SQL, seperti penggunaan T-SQL dan PL / SQL – atau menggunakan tanda # untuk menjadikan perintah SQL didepan karakter tersebut menjadi tidak dieksekusi, karena akan menjadi *comment* (perintah yang tidak dijalankan). Teknik ini biasanya digunakan untuk mencari celah *login*. Dengan menambahkan tanda diatas maka perintah setelah *username* tidak akan di eksekusi, yang berakibat *login* sistem tersebut hanya menggunakan *username* sebagai parameter pengecekan.

5. Defacing

Defacing adalah kegiatan *hacking* yang mengubah tampilan sebuah situs *web*. Sedangkan cara yang digunakan bisa bermacam-macam seperti SQL *Injection*, mencari password, dan cara lainnya. Serangan *Defacing* telah lama dianggap sebagai salah satu ancaman utama bagi situs web dan web aplikasi perusahaan, perusahaan, dan organisasi pemerintah. Serangan *Defacing* dapat membawa konsekuensi serius bagi pemilik situs web, termasuk gangguan langsung operasi situs web dan kerusakan reputasi pemilik, yang dapat mengakibatkan kerugian finansial yang besar. Banyak solusi telah diteliti dan digunakan untuk memantau dan mendeteksi serangan perusakan situs web, seperti yang didasarkan pada perbandingan *checksum comparison*, *diff comparison*, dan analisis *DOM tree*, penejasannya sebagai berikut :

a. Checksum Comparison

Deteksi *Defacing* situs web dan aplikasi web menggunakan *checksum comparison* adalah salah satu metode paling sederhana untuk

menemukan perubahan di halaman web. Pertama, *checksum* konten halaman web dihitung menggunakan algoritma *hashing*, seperti MD5 atau SHA1, dan disimpan dalam profil deteksi. Kedua, halaman web dipantau dan *checksum* baru dari konten halaman web dihitung dan kemudian dibandingkan dengan *checksum* yang sesuai disimpan dalam profil deteksi. Jika dua nilai *checksum* tidak sama, alarm dinaikkan. Teknik ini tampaknya bekerja dengan baik untuk halaman web statis. Namun, teknik ini tidak berlaku untuk halaman dinamis, misalnya halaman web *e-commerce* atau situs web karena konten mereka sering berubah.

b. *Diff comparison*

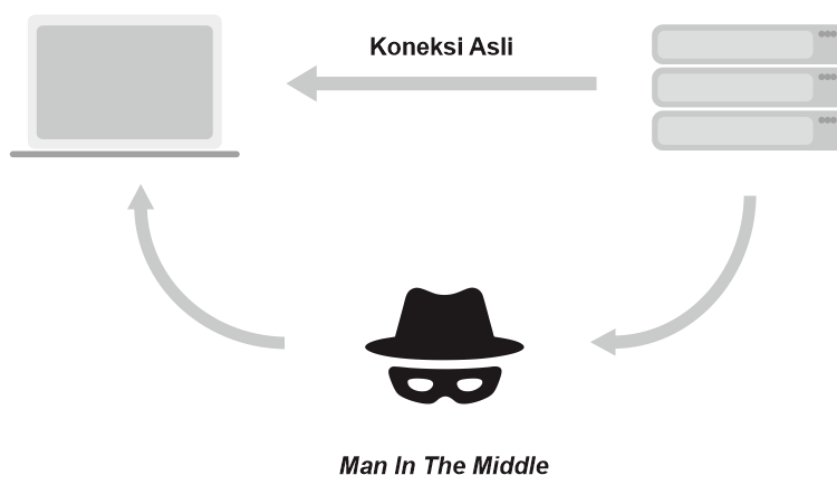
Metode *diff comparison* menggunakan alat DIFF yang umumnya tersedia di Linux dan UNIX Lingkungan. DIFF digunakan untuk membandingkan konten halaman web saat ini dan kontennya yang disimpan dalam profil untuk menemukan perubahan. Hal yang paling sulit untuk dilakukan adalah memutuskan ambang anomali. sebagai masukan untuk proses pemantauan setiap halaman web. Singkatnya, teknik perbandingan Diff adalah relatif efektif untuk sebagian besar halaman dinamis jika ambang anomali dipilih dengan benar.

c. Analisis DOM *Tree*

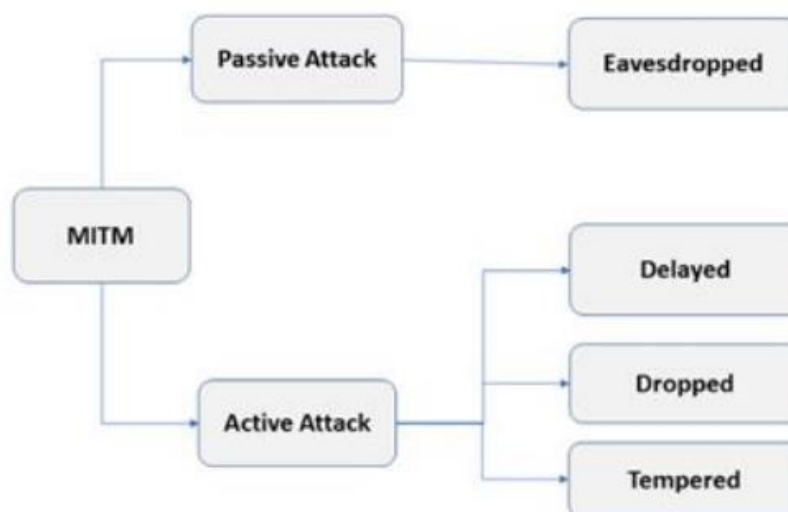
DOM adalah API yang menentukan struktur logis dari halaman web, atau dokumen HTML. DOM Dapat digunakan untuk memindai dan menganalisis struktur halaman web. Analisis *tree* DOM dapat digunakan untuk mendeteksi perubahan struktur halaman web, bukan perubahan dalam konten halaman web. Pertama, struktur halaman diekstraksi dari konten halaman dalam kondisi kerja normal dan disimpan dalam profil. Kemudian, struktur halaman yang dipantau diekstraksi dan kemudian dibandingkan dengan struktur halaman yang disimpan di profil untuk mencari perbedaan Umumnya, teknik ini bekerja dengan baik untuk halaman web terstruktur yang stabil. Namun tidak dapat mendeteksi modifikasi yang tidak sah dalam konten halaman web.

6. MITM (*Man In The Middle*) Attack

Menurut Shubh dan Sharma (2016), Serangan *man-in-the-middle* adalah serangan dimana penyerang diam-diam melakukan *relay* dan mungkin mengubah komunikasi antara dua pihak yang percaya mereka saling berkomunikasi satu sama lain. Serangan ini memungkinkan si penyerang untuk mencuri data seperti *username* dan *password* hanya dengan menjadi penengah antara *client* dan *server*. Serangan MITM dibagi menjadi dua jenis seperti aktif dan serangan pasif. Selanjutnya, serangan aktif dipisahkan menjadi tiga jenis seperti tertunda, jatuh dan marah.



Gambar 2.1 Ilustrasi *Man In The Middle*



Gambar 2.2 *Type MITM Attack*

a. *Delayed Message in MITM*

Aplikasi seperti VANET (*Vehicular network*), perbankan transaksi dan medan perang sangat terpengaruh dengan MITM dengan keterlambatan pesan karena sifat sensitif dari keterlambatan pesan karena sifat sensitif dari pesan yang dapat dibuatnya. Misalnya mempertimbangkan skenario medan perang di mana penundaan pesan tunggal oleh node berbahaya dapat menyebabkan situasi bencana di mana node yang sah tidak dapat menerima pesan tepat waktu. Akibatnya, situasi seperti itu juga dapat membahayakan kehidupan manusia.

b. *Dropped message in MITM*

Selama serangan pesan yang dijatuhkan, penyerang dengan sengaja menjatuhkan pesan otentik PG dengan menekan lebih jauh penyebaran PG. Akibatnya, penyerang menghalangi pengguna asli (AU) dari menerima pesan yang terkait dengan langkah-langkah keselamatan dan non-keselamatan. Misalnya, selama VANET ketika kendaraan menyiarkan informasi yang sah tentang jalan kondisi, dan jika informasi itu dijatuhkan maka manusia hidup akan berada dalam bahaya.

2.6 Acunetix WVS

Acunetix adalah program pengujian keamanan *web* yang mendeteksi dan mengontrol secara menyeluruh terutama *website* berbasis HTML dan JavaScript. Siklus Hidup Pengembangan Perangkat Lunak (SDLC) terintegrasi dengan manajemen proyek atau sistem pelacakan *bug*, termasuk laporan yang komprehensif. Acunetix mendeteksi dan melaporkan celah keamanan aplikasi *website*. Acunetix adalah salah satu program paling sukses di pasar untuk mendeteksi celah keamanan SQL Injection dan XSS (Ula, 2019). Acunetix mengelompokkan hasil VA menjadi 4, yaitu *High Risk Alert* (sangat rentan), *Medium Risk Alert Level 2* (disebabkan oleh kesalahan konfigurasi server dan kelemahan pengkodean situs), *Low Risk Alert Level 1* (berasal dari kurangnya enkripsi lalu lintas data atau keamanan direktori), *Informational Alert* (bersifat informasi yang dianggap bisa menjadi celah seperti IP *address*, alamat email dan lain lain).

2.7 Penelitian Terdahulu

Terdapat beberapa penelitian terdahulu mengenai *vulnerability* dengan beberapa objek yang berbeda dapat dilihat pada Tabel 2.1. sebagai berikut:

Tabel 2.2 Penelitian Terdahulu Tentang *Vulnerability*

No	Peneliti	Judul	Metode	Hasil
1	Harry Purmanta Siagian, M. Akbar, Andri (2018)	Vulnerability Assessment Pada Web Server	<ol style="list-style-type: none">1. <i>Vulnerable applications version</i>, digunakan untuk mengecek aplikasi yang sudah kadaluarsa ter-<i>install</i> pada <i>web server</i>.2. Penetrasi dengan Teknik penyerangan;<ul style="list-style-type: none">• SQL <i>injection</i> dengan SQLMap• XSS• Deniel Of Service	<ul style="list-style-type: none">- Hasil uji coba menggunakan aplikasi-aplikasi tersebut didapatkan beberapa kerentanan yang di akibatkan dari beberapa aplikasi yang terinstal di <i>server</i> kadaluwarsa.- Teknik pengujian untuk <i>web server</i> masih begitu banyak yang belum dicobakan pada penelitian ini dan itu berarti belum semua celah pada www.binadarma.ac.id.
2	Jai Narayan Goel, BM Mehtre (2015)	Vulnerability Assessment & Penetration Testing as a Cyber Defence Technology	<p>Vulnerability Assessment and Penetration Testing (VAPT)</p> <ol style="list-style-type: none">1. Reconnaissance2. Vulnerability Detection3. Information Analysis and Plannig4. Penetration Testing5. Privilege Escalation6. Result Analysis7. Reporting	<ul style="list-style-type: none">- VAPT sebagai teknologi pertahanan <i>Cyber</i> yang kuat.- Pengujian VAPT dapat menghentikan kasus serangan <i>cyber</i> dan memberikan penguatan keamanan sistem.

No	Peneliti	Judul	Metode	Hasil
			8. Clean-up	
3	Imam Riadi, Herman, Aulyah Zakilah Ifani (2021)	Optimasi Keamanan Web Server terhadap Serangan Broken Authentication Menggunakan Teknologi Blockchain	<ol style="list-style-type: none"> 1. Membuat <i>flowchart</i> dimana <i>blockchain</i> akan dipasangkan pada sistem login. 2. Melakukan aktivasi <i>blockchain</i> Ethereum 3. Melakukan ujicoba dengan <i>tools</i> Burp Suite 	Teknologi <i>blockchain</i> yang dipasang dapat menambah keamanan dari sistem login yang di ujicoba, dengan teknologi <i>blockchain</i> dapat meng-enkripsi <i>username</i> dan <i>password</i> .
4	Erick Irawadi Alwi, Herdianti, Fitriyani Umar (2020)	Analisis Keamanan Website Menggunakan Teknik Footprinting dan Vulnerability Scanning	<ol style="list-style-type: none"> 1. Menentukan ruang lingkup 2. <i>Footprinting</i> dan <i>information Gathering</i> (Google, Ping, Nmap-Zenmap, Whois) 3. Vulnerability Scanning (Pentest-tools.com, Acunetix, OWASP ZAP) 4. Vulnerability Analisis 5. Laporan 	ditemukannya beberapa <i>vulnerability</i> pada <i>website</i> target, peneliti membuat rekomendasi perbaikan dalam bentuk report (laporan) untuk pengelola <i>website</i> sehingga dapat menjadi pegangan untuk admin pengelola <i>website</i> dalam melakukan perbaikan celah keamanan (<i>vulnerability</i>)
5	Febri Al Fajar (2020)	Analisis Keamanan Aplikasi Web Prodi Teknik Informatika Uika Menggunakan Acunetix Web Vulnerability	<ol style="list-style-type: none"> 1. Melakukan <i>scanning</i> dengan Acunetix WVS 2. Melakukan ujicoba manual berapa celah kamanan yang di temukan WVS ke web target penelitian. 	Hasil dari pengujian dapat ditemukan berbagai level kerentanan dari level kerentanan <i>Low</i> pada domain ti.ft.uika-bogor.ac.id sampai level kerentanan <i>High</i> pada sub domain lainnya yang berupa sub domain fakultas. Dari hasil analisis yang diperoleh dan dapat dilihat berbagai <i>web alerts</i> yang terdapat pada sebuah Aplikasi web tersebut. Adapun berbagai <i>web alerts</i> yang berhasil ditemukan berupa SQL

No	Peneliti	Judul	Metode	Hasil
				Injection, Cross Site Scripting dan berbagai <i>web alerts</i> lainnya
6	Rini Mayasari, Azhari Ali Ridha, Didi Juardi, Kiki Ahmad Baihaqi (2020)	Analisis Vulnerability pada Website Universitas Singaperbangsa Karawang menggunakan Acunetix Vulnerability	<ol style="list-style-type: none"> 1. Melakukan <i>scanning</i> dengan Acunetix WVS 2. Penentuan Risk Vulnerability dengan WVS 	Tingkat kerentanan website Universitas Singaperbangsa berada pada level 2 yaitu Medium, sehingga kemungkinan untuk mengakses dan mengumpulkan informasi sensitif, karena dengan informasi tersebut penyusup bisa dengan mudah mengeksploitasi kelemahan yang ada.
7	Mira Orisa, Michael Ardita (2021)	Vulnerability Assesment Untuk Meningkatkan Kualitas Keamanan Web	<ol style="list-style-type: none"> 1. Menggunakan NMAP sebagai Vulnerability <i>tools scanner</i>. 2. Mendeteksi protokol HTTP 3. Menscanning <i>open proxy</i> HTTP 4. Melakukan DNS <i>brute</i> 5. Mendeteksi <i>Cross Site Scripting</i> 6. Mendeteksi <i>SQL inhection</i> 	Dapat melakukan pengecekan terhadap serangan <i>denial of service</i> , dapat menemukan <i>vulnerability</i> xss pada file php, dan dapat menemukan <i>vulnerability</i> terhadap serangan SQL Injection.
8	Feri Wibowo, Harjono, Agung Purwo Wicaksono (2019)	Uji Vulnerability pada Website Jurnal Ilmiah Universitas Muhammadiyah Purwokerto Menggunakan OpenVAS dan	<ol style="list-style-type: none"> 1. <i>Open Vulnarebaliay Assessment</i> 2. System (OpenVAS) dan Acunetix Web Vulnerability Scanner (Acunetix WVS) 	OpenVAS menemukan celah kelemahan sejumlah 9 data, sedangkan Acunetix WVS menemukan celah kelemahan sejumlah 166 data. Dengan perbandingan waktu scanning yang cukup jauh yaitu OpenVAS membutuhkan waktu 60 menit sedangkan Acunetix WVS membutuhkan waktu 954 menit.

No	Peneliti	Judul	Metode	Hasil
		Acunetix WVS		
9	Gitanjali Simran T, Sasikala D (2019)	Vulnerability Assessment of Web Applications using Penetration Testing	<ol style="list-style-type: none"> 1. Batasan proyek; pemahaman terhadap proses bisnis sistem yang akan diuji, pemahaman kompleksitas sistem dan penentuan waktu dan biaya 2. Proses <i>Vulnerability Assessment</i> menggunakan OpenVAS dan AcunetixWVS 3. Melakukan perbandingan hasil <i>scanning</i> antara dua <i>tools</i> uji. 	Penilaian kerentanan dan tes penetrasi yang dilakukan dapat menemukan sebagian besar kerentanan yang diketahui, tes ini saja tidak cukup untuk mengesahkan sistem sebagai bebas risiko. Untuk mengeksplorasi semua aspek tes yang lebih khusus dan spesifik sistem harus dilakukan yang dapat mencakup analisis kode manual dan menulis skrip khusus
10	Yosua Ade Pohan, Yuhandri Yunus, Sumijan (2021)	Meningkatkan Keamanan Webserver Aplikasi Pelaporan Pajak Daerah Menggunakan Metode Penetration Testing Execution Standar	<ol style="list-style-type: none"> 1. Mengidentifikasi masala aplikasi <i>webserver</i> 2. Melakukan observasi lapangan server aplikasi <i>webserver</i> 3. Penerapan metode <i>penetration testing execution standart</i> 4. Hasil dan pembahasan 5. Kesimpulan saran perbaikan <i>webserver</i> 	Kategori keamanan aplikasi yang sebelumnya pada kategori Medium dengan 7 buah jenis kerentanan dapat diturunkan menjadi kategori <i>low</i> dengan hanya 1 buah jenis kerentanan, sehingga keamanan <i>webserver</i> aplikasi pelaporan pajak daerah dapat ditingkatkan.

BAB III

METODOLOGI PENELITIAN

3.1 Pendahuluan

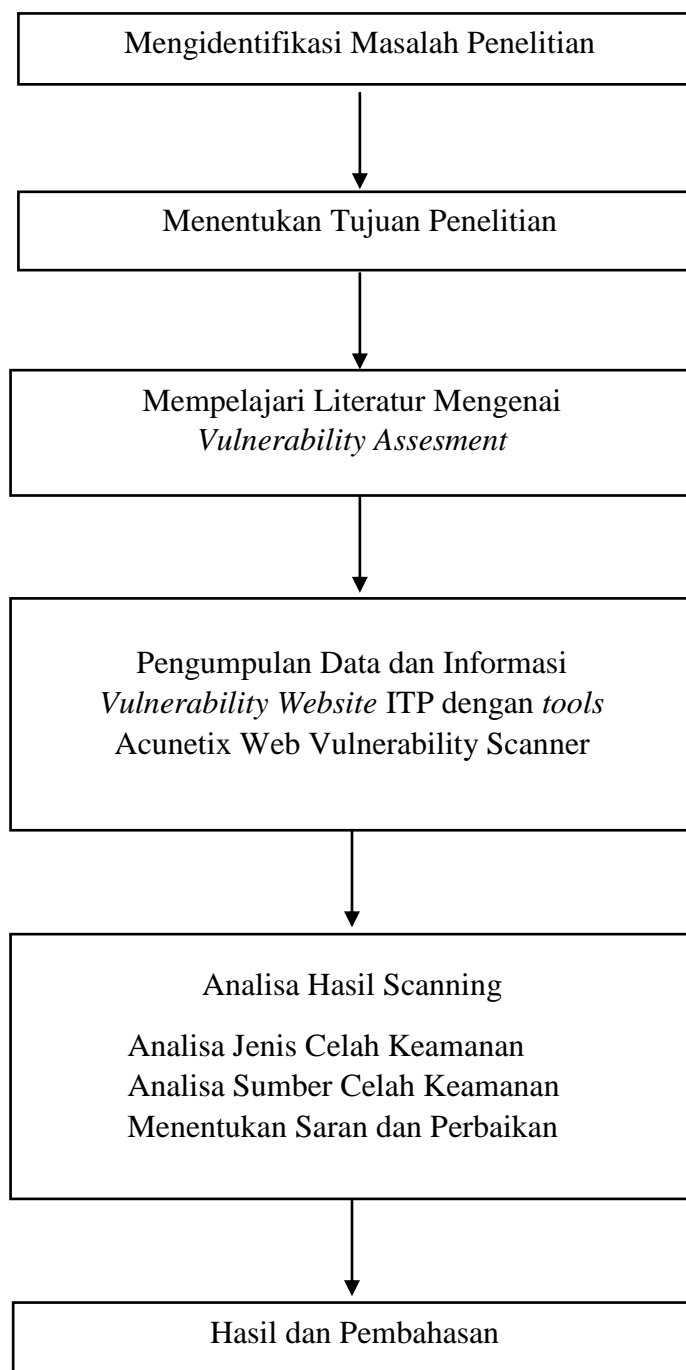
Metodologi penelitian merupakan suatu pendekatan yang meninjau beberapa bagian dalam menemukan masalah dan menanganinya menggunakan teknik yang logis yang sesuai dengan kebenaran yang sedang dipertimbangkan. Metodologi penelitian memiliki tiga komponen yang harus dipertimbangkan secara tepat, khususnya komponen tujuan penyelidikan, komponen tenaga yang akan diselesaikan untuk mencapai tujuan, dan teknik ilmiah.

Metodologi penelitian mengidentifikasi semua tahapan yang digunakan dalam pembuatan struktur kerja atau biasa dikenal dengan kerangka kerja. Kerangka kerja digunakan untuk membuat tahapan – tahapan yang akan diselesaikan dalam penelitian, sehingga tahapan tersebut mempengaruhi setiap tahapan dalam mencapai tujuan penelitian.

Tujuan penelitian ini adalah untuk menganalisis *vulnerability* yang terdapat pada *website* Institut Teknologi Padang dengan menggunakan *tools* Acunetix Web *vulnerability scanner*. Data yang sudah diperoleh akan ditelaah celah keamanan yang ditemukan satu persatu berdasarkan jenis celah keamanan. Dengan mengelompokkan jenis celah keamanan memudahkan untuk menganalisis. Informasi ini dijadikan landasan untuk mengetahui apa yang menjadi penyebab dan pemberian solusi untuk masalah celah keamanan ini.

3.2 Kerangka Kerja Penelitian

Kerangka kerja penelitian adalah suatu tahapan dalam menyelesaikan suatu permasalahan yang akan diteliti. Kerangka kerja penelitian dapat dilihat pada Gambar 3.1.



Gambar 3.1 Kerangka Kerja Penelitian

Kerangka kerja merupakan penggambaran terperinci yang didasari dengan cara yang terorganisir sehingga penelitian dapat mencapai tujuannya dan memiliki penilaian yang semestinya. Berdasarkan kerangka kerja pada Gambar 3.1 diatas, setiap tahapan kerangka kerja dapat dijelaskan sebagai berikut:

1. Mengidentifikasi Masalah Penelitian

Identifikasi masalah adalah tahapan untuk menemukan masalah sebelum melakukan penelitian. Identifikasi masalah diselesaikan dengan bergerak menuju objek pemeriksaan. Alasan melakukan tahapan ini adalah untuk menentukan permasalahan yang terjadi dengan baik, sehingga diyakini bahwa penelitian dapat memberikan jawaban yang paling ideal untuk mengatasi permasalahan tersebut. Permasalahan yang dihadapi adalah *Vulnerability Assessment* yang seharusnya dilakukan sebelum *release* dianggap tidak terlalu penting dan hanya berpedoman pada *black box test* atau uji fungsionalitas sistem. Pada *website* ITP belum pernah dilakukan *vulnerability assesment* sama sekali, yang dimana kerentanan pada keamanan *website* merupakan hal yang harus diperhatikan bagi setiap institusi agar terhindar dari tindakan kejahatan di dunia maya.

2. Menetapkan Tujuan Penelitian

Tahapan penentuan tujuan ini merupakan tahapan di mana peneliti mengemukakan tujuan dari penelitian agar tidak keluar dari hasil yang ingin diperoleh. Tujuan penelitian ini adalah untuk menganalisa *vulnerability* atau celah keamanan yang ada pada *website* ITP dan memberikan solusi terhadap masalah yang ditemukan, dengan tujuan laporan dari hasil penelitian ini dapat menjadi acuan bagi pengembang atau administrator sistem untuk melakukan perbaikan dan pengembangan sistem.

3. Mempelajari Literatur Mengenai *Vulnerability Assessment*

Mempelajari literatur merupakan fase dalam mencari tahu tentang *vulnerabilty* dan *vulnerabilty assesment*. Sebagai metode yang digunakan dalam penelitian, akan digunakan untuk menganalisis *vulnerability* yang terdapat pada *website* ITP, literatur yang sudah didapatkan akan dipilih dan dicocokkan dengan apa aja yang akan dipakai di dalam penelitian. Sumber

dari literatur ini dapat berasal dari artikel, jurnal ilmiah mengenai *vulnerability assesment*, serta referensi-referensi yang berhubungan dengan penelitian.

4. Pengumpulan Data dan Informasi *Vulnerability Website ITP*

Tujuan dari pengumpulan data adalah untuk mendapatkan suatu informasi dari data-data yang dibutuhkan untuk penelitian agar mencapai tujuan yang diharapkan. Pada tahapan ini dijelaskan berupa kegiatan *scanning* dari *website* ITP dengan menggunakan *tools* acunetix web *vulnerability Scanner*. Dari data yang didapat akan dihimpun dalam bentuk tabulasi agar mudah dilakukan analisa. *Tools* ini akan bekerja juga sebagai *penetration testing* yang akan langsung diujikan dengan sistem. Sebagai contoh; untuk pengujian *SQL injection* maka acunetix akan bekerja meng-*inject* dengan berbagai macam kemungkinan untuk mendapatkan titik lemah dari sistem yang diuji. Pada *tools* acunetix WVS ini menghasilkan report dan keluaran berupa tingkatan atau *level alert*. selain itu acunetix juga memberikan *report* secara umum masalah yang didapat setelah dilakukan *scanning*.

5. Analisa Hasil *Scanning*

Pada tahapan ini, data yang didapatkan dari hasil *scanning* akan ditelaah celah keamanan yang ditemukan satu berdasarkan jenis celah keamanan. Dengan mengelompokkan jenis celah keamanan memudahkan untuk menganalisis. Hasil analisa akan di tunjang dengan beberapa literatur yang sudah dibaca.

a. Analisa Jenis Celah Kemanan

Analisa yang dilakukan pada tahapan ini, adalah mengelompokkan jenis celah keamanan, seperti; *SQL Injection*, *XSS*, *CSRF* dan lain-lain. Dengan adanya pengelompokan ini memudahkan untuk menganalisa apa yang harus dilakukan untuk menangani celah dengan tipe ini.

b. Analisa Sumber Celah Keamanan

Setelah mengelompokkan jenis celah, maka akan diturunkan *case by case* dari mana sumber celah tersebut. Kenapa hal ini dilakukan karena untuk memberikan solusi dan perbaikan yang tepat sasaran untuk perbaikan. Analisa sumber celah keamanan ini juga dapat dibagi menjadi 2 yaitu; bersumber dari kesalahan penulisan *syntax* program (*coding*); dan bersumber dari kekurangan atau kesalahan konfigurasi di sisi infrastruktur (*server*) seperti pengaturan hak akses direktori (*directory level acces control*), yang bisa menyebabkan *client* atau *attacker* bisa mengakses ke *root* folder.

c. Menentukan Saran Perbaikan

Setelah jenis dan sumber telah di temukan, maka tahapan ini adalah tahapan yang paling penting, karena walaupun satu *case* dengan *case* yang lain sama, namun perbaikan atau *treatment* yang dilakukan berbeda-beda. Hal ini karena beberapa faktor seperti penulisan *script* yang berulang atau tidak menggunakan fungsi yang dipakai bersama, ataupun perbaikan yang tidak dilakukan merataa karena belum dilakukan *scanning*, yang menyebabkan tumpang tindih pada perbaikan sistem.

6. Hasil dan Pembahasan

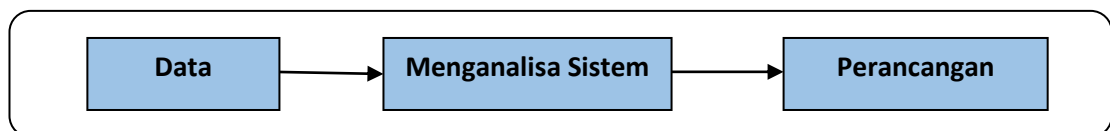
Pada tahapan ini data yang didapat dari proses *scanning* menggunakan acunetix web *vulnerability scanner* akan diuraikan dengan tujuan data tersebut dapat menjadi informasi atau *report* yang dapat dijadikan saran perbaikan untuk pengelola *website*. Analisa yang dilakukan adalah dengan menjelaskan jenis celah keamanan, sumber celah, solusi perbaikan dari celah keamanan tersebut. Selain dengan menggunakan metode tabulasi, tata cara penulisan *report* harus disampaikan dengan bahasa yang teknis, kenapa demikian karena hasil dari *report* ini akan diberikan kepada pengembang atau *developer*. Namun selain itu untuk memudahkan pimpinan dalam membaca laporan, maka juga perlu disiapkan kesimpulan dengan menggunakan tata bahasa yang mudah dipahami.

BAB IV

ANALISA DAN PERANCANGAN

4.1 Tahapan Analisa dan Perancangan

Berdasarkan kerangka kerja penelitian yang terdapat pada metodologi penelitian di Bab 3, bahwa tahapan kerja terdiri dari mengidentifikasi masalah penelitian, menentukan tujuan penelitian, mempelajari literatur mengenai *vulnerability assesment*, pengumpulan data, analisa data serta hasil dan pembahasan. Guna memudahkan dalam analisa dan perancangan penelitian maka dibuat bagan alir analisa dan perancangan seperti pada Gambar 4.1 dibawah ini.



Gambar 4.1 Bagan Alir Analisa dan Perancangan

4.2 Data

Data yang digunakan pada penelitian ini adalah data hasil *scanning* atau *vulnerability assesment* dari *website* Institut Teknologi Padang. Untuk mendapatkan data ini digunakan *tools* Acunetix WVS. Jenis data yang diperoleh dari *tools* Acunetix WVS adalah *report* dari proses *penetration test* yang berupa data celah keamanan yang terdapat pada *website* tersebut. Acunetix WVS pada penelitian ini memiliki peran penting karena dengan *tools* ini data penelitian didapatkan dan digunakan sebagai *benchmark* hasil analisa perbaikan yang dilakukan setelah dilakukan *vulnerability assesment*.



Gambar 4.2 Halaman Website Institut Teknologi Padang

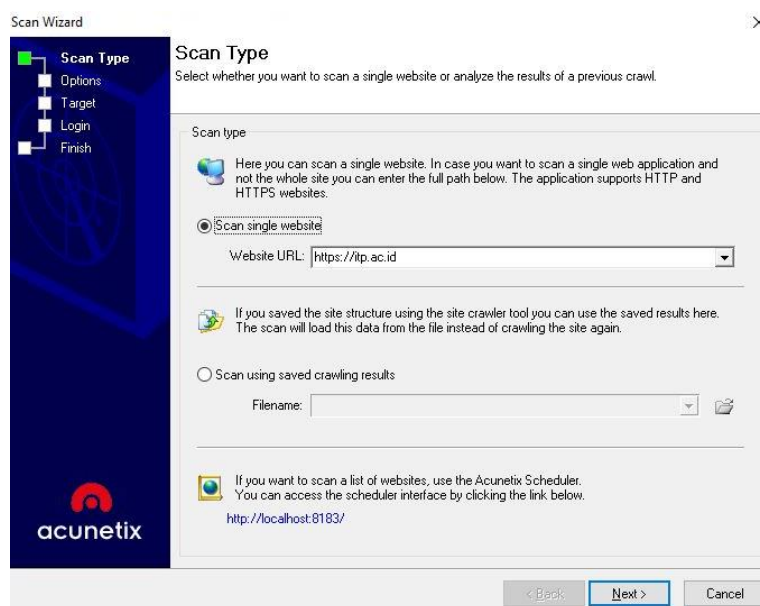
Versi dari Acunetix yang digunakan adalah versi 10.5. Acunetix mengelompokkan hasil VA menjadi 4, yaitu *High Risk Alert* (sangat rentan), *Medium Risk Alert Level 2* (disebabkan oleh kesalahan konfigurasi server dan kelemahan pengkodean situs), *Low Risk Alert Level 1* (berasal dari kurangnya enkripsi lalu lintas data atau keamanan direktori), *Informational Alert* (bersifat informasi yang dianggap bisa menjadi celah seperti IP address, alamat email dan lain lain). Proses mendapatkan data dengan menggunakan tools Acunetix WVS dilakukan dengan tahapan berikut:

1. Klik *file > new > new website scan* untuk memulai *scan wizard* atau klik *new scan* di sisi kiri atas menu bar Acunetix WVS.



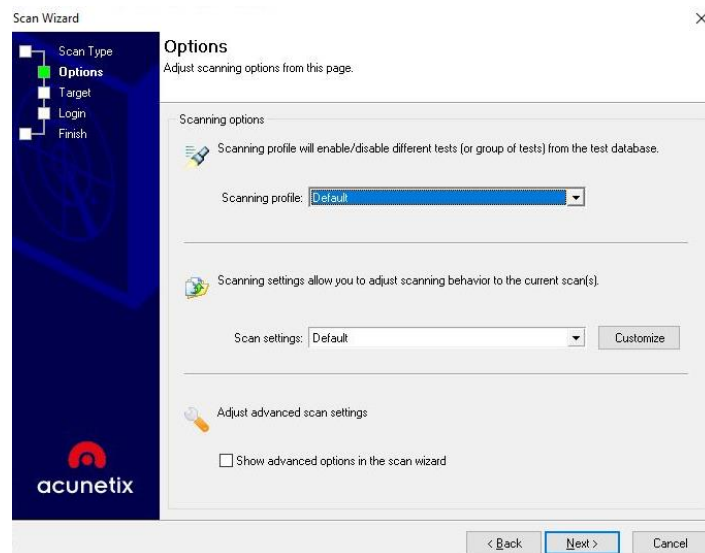
Gambar 4.3 New Website Scan

2. Tentukan jenis *scanning* yang akan digunakan, pada tahap ini diberikan dua opsi *scan* yaitu; *Scan Single Website* dan *Scan Using Saved Crawling Result*, dimana opsi pertama adalah melakukan *scanning* dengan *live* dari URL atau aplikasi yang sedang berjalan, sedangkan opsi ke dua adalah melakukan *scanning* dengan *file website* yang sudah di simpan di *local* komputer atau biasa disebut dengan hasil *crawling*. Pada penelitian ini kita gunakan metode *scanning live* dari aplikasi yang berjalan.



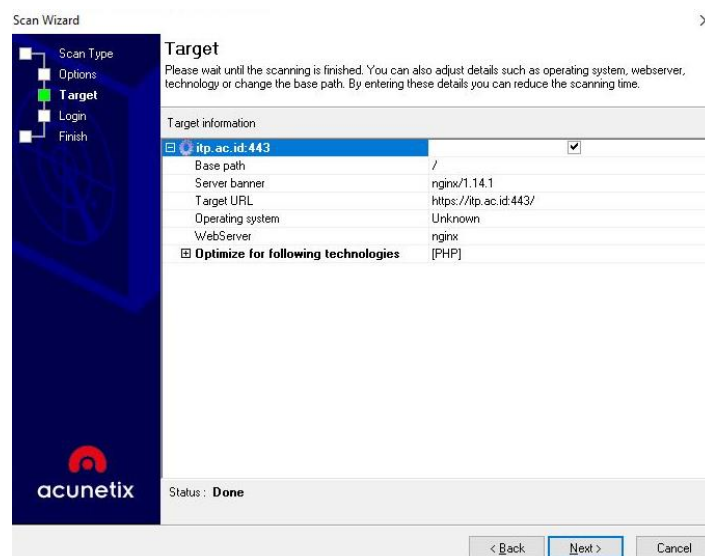
Gambar 4.4 Scan Type Website

3. Pada menu *option* diberikan jenis *scanning option*, ini digunakan untuk memilih jenis *scan* yang akan digunakan, apakah *default* yang berarti semua jenis *scan* dan *penetration* dilakukan atau hanya satu metoda saja seperti XSS atau SQL Injection dan lain sebagainya. Dalam penelitian ini digunakan *scanning profile default*.



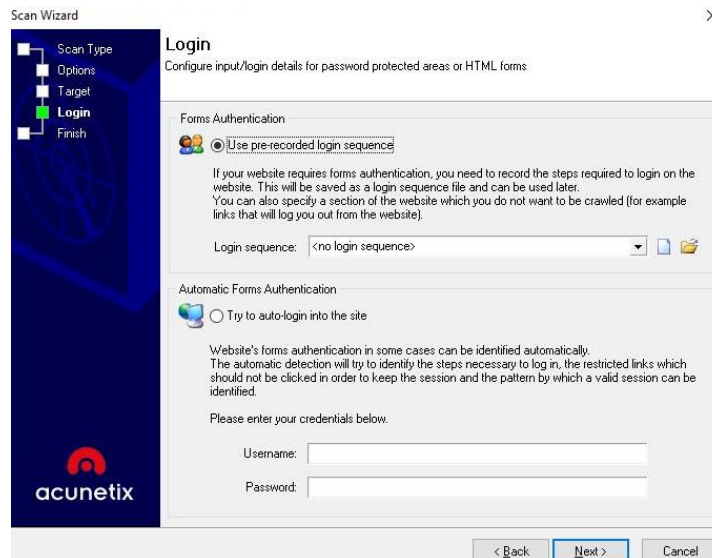
Gambar 4.5 Options Scanning Website

4. Pada tahapan ini, akan disajikan informasi dari target *scanning* yang telah di set di awal, informasi yang diberikan yaitu jenis *webserver* berserta versinya dan teknologi bahasa pemrograman yang dipakai yaitu PHP.



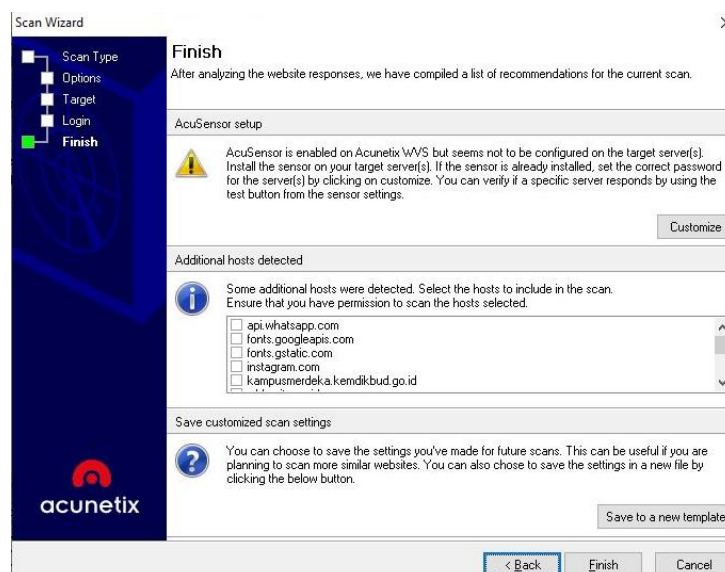
Gambar 4.6 Target Scanning Website

5. Tahap selanjutnya adalah untuk mengkonfirmasi apakah *website* yang akan di *scan* memerlukan *login* atau melewati *form login*. Jika tidak menggunakan *form login*, maka cukup pilih opsi yang pertama yaitu *use pre-recorded login sequence*. Jika menggunakan *login* pilih opsi kedua dan masukkan *username* dan *password*.



Gambar 4.7 Login Scanning Website

6. Tahapan terakhir yaitu konfirmasi apakah *setup* yang telah di inisialisasi sudah benar dan jika ada *additional host* atau URL lain yang berada pada target juga akan di *scan* maka centang untuk memproses, jika tidak langsung *finish* dan menunggu hasil *scan*.



Gambar 4.8 Tahapan Terakhir Initialize Scanning

Tabel 4.1 Data Penelitian

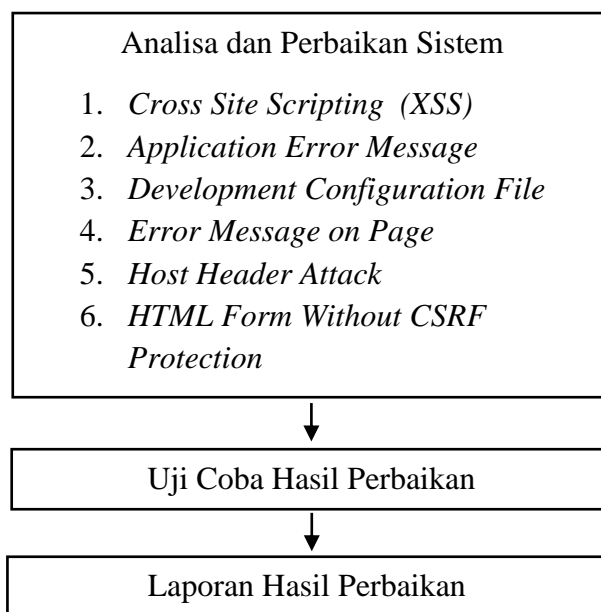
No	Vulnerability	Lokasi Bug
1	<i>Cross Site Scripting (High)</i>	/ /akreditas-sipil-s1 /beasiswa /berita /berita/arsip /berita/arsip/10 /berita/arsip/15 /berita/arsip/20 /berita/arsip/450 /berita/arsip/5 /berita/arsip/academic /berita/arsip/campus-activity /berita/arsip/career /berita/arsip/new-student-registration /berita/arsip/news /berita/arsip/opini /berita/arsip/publication /berita/detail/105 /berita/detail/154 /berita/detail/154/announcement-of-ordik-t.a-2014 /berita/detail/159 /berita/detail/163 /berita/detail/165 /berita/detail/217 /berita/detail/225 /berita/detail/226 /berita/detail/231 /berita/detail/231/informasi-beasiswa-yayasan-t.a-2015 /berita/detail/234 /berita/detail/271 /berita/detail/273 /berita/detail/276 /berita/detail/278 /berita/detail/286 /berita/detail/391 /berita/detail/407 /berita/detail/409 /berita/detail/411 /berita/detail/412 /berita/detail/426 /berita/detail/444 /berita/detail/446 /berita/detail/447 /berita/detail/448 /berita/detail/449 /berita/detail/451 /berita/detail/453 /berita/detail/454 /berita/detail/455 /berita/detail/457 /berita/detail/458

No	Vulnerability	Lokasi Bug
		/berita/detail/459 /berita/detail/460 /berita/detail/461 /berita/detail/462 /berita/detail/463 /berita/detail/464 /berita/detail/465 /berita/detail/466 /berita/detail/467 /berita/detail/468 /berita/detail/469 /berita/detail/470 /berita/detail/82 /berita/detail/89 /berita/detail/94 /berita/detail/99 /berita/page /dosen /download /education /faculty_of_vocation /home-fakultas-teknik /index.php /index.php/web /kurikulum-mesins1 /kurikulum-sipil-s1 /pendidikan /penelitian /profil /profile /search (3) /teknik-elektro-s1 /teknik-geodesi-s1 /teknik-informatika-s1 /teknik-lingkungan /teknik-mesin-d3 /teknik-mesin-s1 /teknik-sipil-d3 /teknik-sipil-s1 /teknologi-listrik-d3 /visi-misi-sipils1
2.	<i>Application Error Message (Medium)</i>	/berita/ (2) /berita/ komentar (2) /search (1)
3	<i>Development Configuration File</i>	/composer.json
4	<i>Error Message On Page</i>	/admin/datatable
5	<i>Host Header Attack</i>	/berita/detail/446/monev-coe-mbkm-itp-digelar-secara-hybrid /berita/detail/453/launching-sistem-informasi-mbkm-itp /berita/detail/457/komit--ini-inovasi-itp-untuk-

No	Vulnerability	Lokasi Bug
		keterbukaan-info... /berita/detail/460/visiting-professor-itp-and-umpedac /berita/detail/462/monev-pkk-m-tahun-2021 /berita/detail/463/monev-coe-mbkm-itp-bahas-kendala-kegiatan /berita/detail/464/rektor-itp-dilantik-28-pejabat-struktural-period... /berita/detail/465/lpj-tahun-2021--rektor-itp:capaian-proker-... /berita/detail/467/raker-tahun-2022--itp-mantapkan-pengemb... /berita/detail/468/ft-itp-teken-perjanjian-kerja-sama-dengan-f... /berita/detail/469/itp-sosialisasikan-aturan-baru-bkd-ikd- /berita/detail/470/sambangi-itp--smkn-5-padang-upgrade-kerja... /kerjasama /mahasiswa /pengabdian /search
6	HTML Form Without CSRF Protection	/search

4.3 Menganalisa Sistem

Pada tahap analisa dan perbaikan hasil *Assesment* awal yang ditemukan akan diuraikan *case by case* menjadi informasi acuan perbaikan yang akan dilakukan. Analisa yang dilakukan pada penelitian ini adalah menjelaskan apa celah kewanan tersebut, bagaimana celah keamanan ini bersumber, dan bagaimana penanganan dan perbaikan yang harus dilakukan agar celah tersebut dapat tertutupi. Terdapat beberapa tahapan proses yang dilakukan seperti pada *flowchart* Gambar 4.9 berikut.



Gambar 4.9 Flowchart Vulnerability Assesment

1. *Cross Site Scripting (XSS)*

Vulnerability ini merupakan serangan *code injection* yang dilakukan pada sisi klien. Serangan *cross site scripting* biasanya digunakan untuk mencuri *cookie*, penyebaran *malware*, *session hijacking* atau pembajakan *session*, dan membelokkan tujuan atau *malicious redirects* dengan mengeksekusi *script* berbahaya di *browser website* korban dengan memasukkan kode berbahaya pada halaman *website*. Serangan XSS ini dimungkinkan terdapat dalam *VBScript*, *activex*, *flash*, dan bahkan *CSS*. Serangan *cross site scripting (XSS)* terjadi apabila data memasuki aplikasi *web* melalui sumber yang tidak tepercaya dan data disertakan dalam konten dinamis yang dikirim ke pengguna *website* tanpa divalidasi untuk konten berbahaya. Berdasarkan data diatas maka pada bagian ini terdapat 94 *bug* (lihat Tabel 4.1). Adapun langkah-langkah penyelesaiannya sebagai berikut :

- a. Membuka *file include navbar.php*
- b. Menemukan *bug Cross Site Scripting (XSS)*
- c. Memperbaiki *bug*
- d. Tabel perbaikan
- e. Hasil

Berikut penjelasan dari langkah-langkah diatas:

a. Membuka *File*

Pada langkah ini akan di cari *file* yang menjadi sumber celah keamanan XSS. Dilihat dari temuan *bug*, lokasi *injection* yang dilakukan berada pada *head* HTML yang berisi *meta* data. Pada aplikasi ini *file head* HTML berada pada *file head.php*.

```

58 <meta property="og:image" itemprop="image" content="<?= $image_head; ?>" />
59 <meta property="og:image:url" itemprop="image" content="<?= $image_head; ?>" />
60 <meta property="og:image:type" content="image/png" />
61 <meta property="og:image:width" content="300">
62 <meta property="og:image:height" content="300">
63
64 <!-- No need to change anything here -->
65 <meta property="og:type" content="website" />
66
67 <!-- Website to visit when clicked in fb or WhatsApp-->
68
69 <meta property="og:url" content="https://<?= $_SERVER['SERVER_NAME']; $_SERVER['REQUEST_URI'] ?>">
70 <meta name="author" content="Institut Teknologi Padang, PUSTIKom">
71 <meta name="googlebot-news" content="index, follow, follow" />
72 <meta name="googlebot" content="index, follow, follow" />
73 <meta name="robots" content="index, follow" />
74 <meta name="robots" content="max-image-preview:large">
75 <meta name="language" content="id" />
76 <meta name="geo.country" content="id" />
77 <meta http-equiv="content-language" content="In-Id" />
78 <meta name="geo.placename" content="Indonesia" />
79 <meta property="og:locale" content="en_GB" />
80

```

Gambar 4.10 File head.php

b. Menemukan *Bug*

Berdasarkan panduan dari Acunetix WVS, *bug* terdapat pada bagian *head* HTML, tepat nya pada *meta* data properti untuk *thumbnail* WhatsApp.

```

31 <!-- Website to visit when clicked in fb or WhatsApp-->
32 <meta property="og:url" content="https://itp.ac.id/berita/arsip/?onmouseover='POKC(9759)'bad="">
33 <meta name="author" content="Institut Teknologi Padang, PUSTIKom">
34 <meta name="googlebot-news" content="index, follow, follow" />
35 <meta name="googlebot" content="index, follow, follow" />
36 <meta name="robots" content="index, follow" />
37

```

Gambar 4.11 Temuan Bug Pada File Head.php

c. Memperbaiki *Bug*

Untuk memperbaiki dan pengamanan dari serangan XSS dapat dilakukan dengan beberapa cara berikut ini:

1. Untuk keamanan inputan data maka mengatasi serangan ini dengan melakukan aktivasi XSS *filterting* dan bisa juga dengan memeberikan *regex* atau karakter yang diperbolehkan untuk diolah didalam *script* PHP. Dalam

kasus ini XSS ditanganui dengan cara input dari semua *method* dilakukan *filtering* dengan fungsi berikut.

```
$this->security->xss_clean(preg_replace("/^[^a-zA-Z0-9\ ]/", "", $VariableInput));
```

Gambar 4.12 Fungsi Filtering XSS

Pada *script* diatas digunakan fungsi `xss_clean` yang merupakan bawaan dari *framework* CodeIgniter untuk memfilter karakter yang di input. Sedangkan fungsi `preg_replace` digunakan untuk fungsi REGEX atau *regular expression* membatasi jenis karakter yang bisa diterima dari inputan. Pada kasus ini REGEX yang digunakan adalah hanya bisa menginputkan karakter a-z, A-Z dan 0-9.

2. Disisi konfigurasi aplikasi cukup dengan menghidupkan *config global xss filtering* dengan *true* pada *file* `config.php`.

```
$config['global xss filtering'] = true;
```

Gambar 4.13 Global XSS Filtering

3. Pada *header* halaman dimana lokasi yang di *inject* XSS, terdapat kesalahan *scripting* dimana untuk mengambil parameter *url* tidak perlu di gunakan *script* `$_SERVER['REQUEST_URI']` yang akan menangkap semua inputan *method* GET pada *url* yang menyebabkan semua inputan tersebut ter-*inject* ke HTML. Seharus nya untuk mendapatkan *url* aktif cukup dengan fungsi `current_url()` seperti Gambar 4.14 berikut.

```
content="https://<?= $_SERVER['SERVER_NAME'].$_SERVER['REQUEST_URI'] ?>">
content="<?= current url() ?>">
```

Gambar 4.14 Global XSS Filtering

d. Tabel Perbaikan

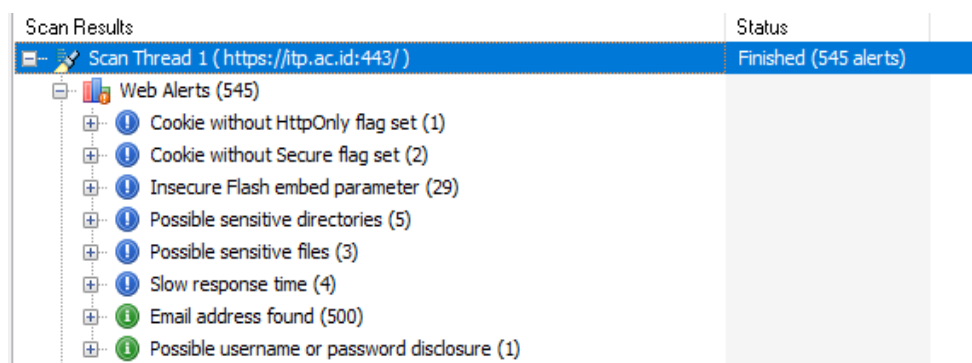
Berikut Tabel perbaikan pada *vulnerability* XSS.

Tabel 4.2 Perbaikan *Bug* XSS

No	Bug	Solusi
1.	XSS Injection Meta Head HTML	1. Input XSS Filtering 2. Fungsi Global XSS Filtering 3. Mengganti <i>script Request</i> URI dengan <i>current_url()</i>

e. Hasil

Setelah tahapan analisa dan perbaikan selesai maka akan dilakukan pengujian ulang pada sistem, apakah perbaikan yang dilakukan mencapai target dari penelitian ini. Ketika perbaikan sudah mencapai target maka akan diberikan laporan hasil dari perbaikan yang dilakukan. Tahapan pengujian ini pengujian dilakukan dengan *scanning* ulang dengan *tools* Acunetix WVS.



Gambar 4.15 Hasil Perbaikan XSS

Hasil dari *scanning* ulang menunjukkan untuk *vulnerability* jenis *Cross Site Scripting* (XSS) sudah berhasil diperbaiki, dengan bukti sudah tidak ada lagi *alert* dari Acunetix WVS.

4.3.1 Laporan Hasil Perbaikan

Berdasarkan perbaikan yang dilakukan, *bug* dari *vulnerability Cross Site Scripting* (XSS) berhasil dihilangkan. Untuk detail laporan perbaikan disajikan dalam bentuk tabel pada Tabel 4.3 berikut.

Tabel 4.3 Laporan Perbaikan

No	Vulnerability	Lokasi Bug	Status Perbaikan
1	Cross Site Scripting (High)	/	Berhasil
		/akreditas-sipil-s1	Berhasil
		/beasiswa	Berhasil
		/berita	Berhasil
		/berita/arsip	Berhasil
		/berita/arsip/10	Berhasil
		/berita/arsip/15	Berhasil
		/berita/arsip/20	Berhasil
		/berita/arsip/450	Berhasil
		/berita/arsip/5	Berhasil
		/berita/arsip/academic	Berhasil
		/berita/arsip/campus-activity	Berhasil
		/berita/arsip/career	Berhasil
		/berita/arsip/new-student-registration	Berhasil
		/berita/arsip/news	Berhasil
		/berita/arsip/opini	Berhasil
		/berita/arsip/publication	Berhasil
		/berita/detail/105	Berhasil
		/berita/detail/154	Berhasil
		/berita/detail/154/announcement-of-ordik-t.a-2014	Berhasil
		/berita/detail/159	Berhasil
		/berita/detail/163	Berhasil
		/berita/detail/165	Berhasil
		/berita/detail/217	Berhasil
		/berita/detail/225	Berhasil
		/berita/detail/226	Berhasil
		/berita/detail/231	Berhasil
		/berita/detail/231/informasi-beasiswa-yayasan-t.a-2015	Berhasil
		/berita/detail/234	Berhasil
		/berita/detail/271	Berhasil
		/berita/detail/273	Berhasil
		/berita/detail/276	Berhasil
		/berita/detail/278	Berhasil
		/berita/detail/286	Berhasil
		/berita/detail/391	Berhasil
		/berita/detail/407	Berhasil
		/berita/detail/409	Berhasil
		/berita/detail/411	Berhasil
		/berita/detail/412	Berhasil
		/berita/detail/426	Berhasil
		/berita/detail/444	Berhasil
		/berita/detail/446	Berhasil
		/berita/detail/447	Berhasil
		/berita/detail/448	Berhasil
		/berita/detail/449	Berhasil
		/berita/detail/451	Berhasil
		/berita/detail/453	Berhasil
		/berita/detail/454	Berhasil
		/berita/detail/455	Berhasil

No	Vulnerability	Lokasi Bug	Status Perbaikan
		/berita/detail/457	Berhasil
		/berita/detail/458	Berhasil
		/berita/detail/459	Berhasil
		/berita/detail/460	Berhasil
		/berita/detail/461	Berhasil
		/berita/detail/462	Berhasil
		/berita/detail/463	Berhasil
		/berita/detail/464	Berhasil
		/berita/detail/465	Berhasil
		/berita/detail/466	Berhasil
		/berita/detail/467	Berhasil
		/berita/detail/468	Berhasil
		/berita/detail/469	Berhasil
		/berita/detail/470	Berhasil
		/berita/detail/82	Berhasil
		/berita/detail/89	Berhasil
		/berita/detail/94	Berhasil
		/berita/detail/99	Berhasil
		/berita/page	Berhasil
		/dosen	Berhasil
		/download	Berhasil
		/education	Berhasil
		/faculty_of_vocation	Berhasil
		/home-fakultas-teknik	Berhasil
		/index.php	Berhasil
		/index.php/web	Berhasil
		/kurikulum-mesins1	Berhasil
		/kurikulum-sipil-s1	Berhasil
		/pendidikan	Berhasil
		/penelitian	Berhasil
		/profil	Berhasil
		/profile	Berhasil
		/search (3)	Berhasil
		/teknik-elektro-s1	Berhasil
		/teknik-geodesi-s1	Berhasil
		/teknik-informatika-s1	Berhasil
		/teknik-lingkungan	Berhasil
		/teknik-mesin-d3	Berhasil
		/teknik-mesin-s1	Berhasil
		/teknik-sipil-d3	Berhasil
		/teknik-sipil-s1	Berhasil
		/teknologi-listrik-d3	Berhasil
		/visi-misi-sipils1	Berhasil

Adapun untuk menghitung nilai persentasi keberhasilan digunakan rumus *average* atau rata-rata seperti berikut.

$$\text{NRK} = (\text{Jumlah data berhasil diperbaiki} / \text{Jumlah data awal}) \times 100\%$$

$$\text{NRK} = (94/94) \times 100\% = \mathbf{100\%}$$

Keterangan :

NRK : Nilai rata-rata *bug* yang berhasil diperbaiki pada kasus

4.4 Perancangan

Adapun rancangan program yang akan diimplementasikan sebagai *bank* data perbaikan adalah sebagai berikut.

4.4.1 Desain Output

1. Halaman Utama

Rancangan halaman utama aplikasi akan disajikan fitur pencarian terhadap data *vulnerability* yang akan dicari. Pada Gambar 4.16, *user* akan menginputkan *keyword* seperti XSS, SQL *injection* atau lainnya. Di halaman ini juga *user* dapat melihat langsung seluruh data tanpa harus memasukkan *keyword* dengan meng-*click* tombol *show all* dan akan diarahkan ke halaman tabel data seperti Gambar 4.16.



Gambar 4.16 Desain Halaman Utama

2. Halaman Tabel *Bank* Data

Halaman tabel *Bank* data pada Gambar 4.17 menampilkan data-data *vulnerability* dan solusi yang diberikan, pada kolom detail terdapat tombol info yang berguna untuk melihat detail dan solusi yang diberikan dari *case* atau kasus yang dipilih. Tombol ini jika di-*click* akan menampilkan sebuah *pop-up*, bukan halaman utuh. Untuk lebih detail dapat dilihat pada Gambar 4.18 berikut.

Knowlegde Transfer Application

Vulnerability Assesment for Web Application

List Data

Show 10 entries Search:

No ↑↓	Vulnerability ↑↓	Bugs ↑↓	Detail
1.	XSS	Server Request URI	i
2.	Application Error Message	Error Message Handler	i
3.	Development Configuration File	File Development Only Owner Can Access	i
4.	Error Message on Page	Error Message Handler	i
5.	Host Header Attack	Header Application Injected and Changed to another site	i
6.	HTML Form Without CSRF Protectio	Handling CSRF Protection on search form	i

Showing 1 to 6 of 6 entries

Previous 1 Next

Knowlegde Transfer Application - Afif Zirwan @ 2022

Gambar 4.17 Tabel Data Pencarian

3. Halaman Info Detail

Halaman ini akan menampilkan detail dari *case* yang dipilih oleh *user* seperti apa jenis *vulnerability*, sumber *bug* dan bagaimana cara memperbaiki dari *bug* tersebut seperti Gambar 4.18 berikut.

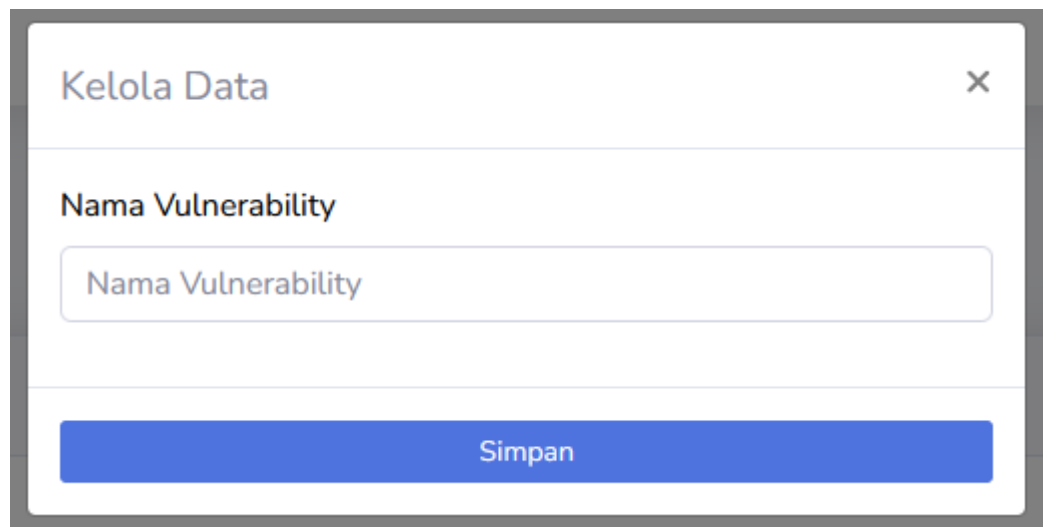


Gambar 4. 18 Desain Halaman Detail

4.4.2 Desain Input

1. Input Data Master *Vulnerability*

Pada halaman input ini digunakan untuk menambagkan master data jenis *vulnerability* yang ditemukan. Data ini berguna sebagai relasi data pada data *bug* yang nanti pada proses penginputan nya akan memilih jenis dari *vulnerability* yang terjadi.

The image shows a web form titled "Kelola Data" with a close button (X) in the top right corner. Below the title, there is a label "Nama Vulnerability" followed by a text input field containing the placeholder text "Nama Vulnerability". At the bottom of the form, there is a blue button labeled "Simpan".

Gambar 4. 19 Input Data Master *Vulnerability*

2. Input Data *Bug's*

Desgin input yang dirancang pada memiliki 3 inputan pada sebuah *form penambahan* data. Inputan pertama diberikan *select box vulnerabilities* yang berguna untuk memilih jenis dari *vulnerabilty* yang ditemukan. Selanjutnya inputan *bugs* digunakan inputan dengan jenis *text area* untuk mendeskripsikan sedikit terkait informasi yang akan ditambahkan. Berbeda dengan *detail and solution* ini digunakan *text editor* agar *user* dapat mengelola postingan seperti menambah gambar, mengelola paragraf dan lain lain.

Gambar 4.20 Input Data *Bug's*

4.4.3 Desain *Database*

Aplikasi ini akan dibangun menggunakan *database* MySQL, dengan memiliki 2 tabel karena aplikasi ini masih tergolong aplikasi sederhana. Tabel yang digunakan adalah tabel master *vulnerability*, tabel *case* seperti pada Tabel 4.4 dan 4.5 berikut.

1. Tabel *Vulnerability*

Tabel ini berfungsi untuk menyimpan data master *vulnerability*. Pada *primary key* terdapat kolom *VulnerabilityID* dengan tipe data *integer*. Untuk kolom *UCreate* digunakan untuk menyimpan *log user* yang menambahkan data. *DCreate* digunakan untuk menyimpan kapan data ini ditambahkan, untuk *UEdited* digunakan untuk menyimpan *user* yang terakhir kali merubah data

dan yang terakhir Dedicated untuk menyimpan tanggal terakhir dirubah. Empat *field* ini juga akan ada pada tabel *bug*.

Tabel 4.4 Rancangan Tabel Vulnerability

Nama *database* : db_knowledge
 Nama tabel : Vulnerability
 Field Key : VulnerabilityID

No	Name	Type	Size	Description
1	VulnerabilityID	Integer	11	ID, <i>Primary Key</i>
2	Nama	Varchar	50	Nama <i>vulnerability</i>
3	DCreate	Datetime		Tanggal dibuat
4	UCreate	Varchar	50	<i>User</i> yang membuat
5	DEdited	Datetime		Tanggal dirubah
6	UEdited	Varchar	50	<i>User</i> yang merubah

2. Tabel Case

Tabel ini berfungsi untuk menyimpan *case* yang ditemukan dari data penelitian ini termasuk kedalamnya juga bagaimana memperbaikinya. Untuk relasi tabel, tabel *case* berelasi dengan tabel *vulnerability* pada kolom VulnerabilityID. Untuk menyimpan detail perbaikan akan ditampung pada tabel *details*.

Tabel 4.5 Rancangan Tabel Case

Nama *database* : db_knowledge
 Nama tabel : Vulnerability
 Field Key : VulnerabilityID

No	Name	Type	Size	Keterangan
1	CaseID	Integer	11	ID, <i>Primary Key</i>
2	VulnerabilityID	Integer	11	<i>Foreign Key</i>
3	Bugs	Text		Nama <i>bug</i>

No	Name	Type	Size	Keterangan
4	Details	Text		Keterangan detail
5	DCreate	Datetime		Tanggal dibuat
6	UCreate	Varchar	50	<i>User</i> yang membuat
7	DEdited	Datetime		Tanggal dirubah
8	UEdited	Varchar	50	<i>User</i> yang merubah

BAB V

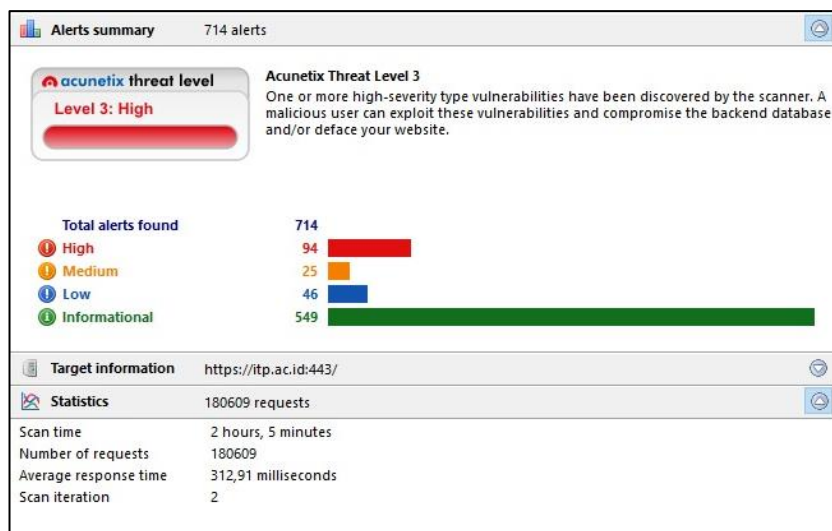
IMPLEMENTASI DAN HASIL

5.1 Implementasi

Setelah melakukan tahapan analisis perancangan, untuk tahap selanjutnya adalah mengimplementasikan hasil yang telah dianalisis dan dirancang sebelumnya. Tahapan-tahapan implementasi tersebut berupa analisa dan perbaikan serta implementasi aplikasi *knowledge transfer*.

5.2 Analisa dan Perbaikan

Dari hasil *Scanning* yang dilakukan, *website* ITP mendapatkan hasil *score* pada level 3, yang berarti *website* ini tidak dalam kondisi aman. Durasi *scan* yaitu 2 jam 5 menit dengan iterasi *scan* atau perulangan *scan* dilakukan sebanyak 2 kali. Pada penelitian ini terdapat 714 *alert* atau celah yang ditemukan yang terdiri dari 94 pada level *High*, 25 pada level *Medium*, 46 pada level *Low* dan 549 pada level *Informational*. Pada penelitian ini, target yang dicapai adalah perbaikan sampai menjadi level 1. Oleh karena itu pada penjelasan perbaikan hanya di fokuskan pada *alert high* dan *medium*.



Gambar 5.1 Hasil Scanning Awal Website

Sesuai langkah kerja penelitian, dari hasil *scan* yang dilakukan, maka dilakukan pengelompokan jenis celah keamanan dan dijelaskan apa dan bagaimana celah itu terjadi, Mengacu pada batasan penelitian ini yaitu memberikan *report* dan perbaikan pada aplikasi yang di uji coba, maka pada sub bab berikut dijelaskan apa saja yang dilakukan untuk perbaikan aplikasi hingga *threat level* menjadi level 1. Berikut penjelasan terkait *alert* atau celah yang ditemukan pada *scanning website* dan perbaikan yang dilakukan.

1. Cross Site Scripting (XSS)

Vulnerability ini merupakan serangan *code injection* yang dilakukan pada sisi klien. Serangan *cross site scripting* biasanya digunakan untuk mencuri *cookie*, penyebaran *malware*, *session hijacking* atau pembajakan *session*, dan membelokkan tujuan atau *malicious redirects* dengan mengeksekusi *script* berbahaya di *browser website* korban dengan memasukkan kode berbahaya pada halaman *website*. Serangan XSS ini dimungkinkan terdapat dalam *VBScript*, *activeX*, *flash*, dan bahkan *CSS*. Serangan *cross site scripting* (XSS) terjadi apabila data memasuki aplikasi web melalui sumber yang tidak tepercaya dan data disertakan dalam konten dinamis yang dikirim ke pengguna *website* tanpa divalidasi untuk konten berbahaya. Berdasarkan data diatas maka pada bagian ini terdapat 94 *bug* (lihat Tabel 4.1). Adapun langkah-langkah penyelesaiannya sebagai berikut :

- a. Membuka file *include* navbar.php
- b. Menemukan bug *Cross Site Scripting* (XSS)
- c. Memperbaiki bug
- d. Tabel perbaikan
- e. Hasil

Berikut penjelasan dari langkah-langkah diatas:

a. Membuka *File*

Pada langkah ini akan di cari *file* yang menjadi sumber celah keamanan XSS. Dilihat dari temuan *bug*, lokasi *injection* yang dilakukan berada pada *head* HTML yang berisi *meta* data. Pada aplikasi ini *file head* berada pada *file head.php*.

```

60 <meta property="og:image:type" content="image/png" />
61 <meta property="og:image:width" content="300">
62 <meta property="og:image:height" content="300">
63
64 <!-- No need to change anything here -->
65 <meta property="og:type" content="website" />
66
67 <!-- Website to visit when clicked in fb or WhatsApp-->
68
69 <meta property="og:url" content="https://<?=$_SERVER['SERVER_NAME'].$_SERVER['REQUEST_URI'] ?>">
70 <meta name="author" content="Institut Teknologi Padang, PUSTIKom">
71 <meta name="googlebot-news" content="index, follow, follow" />
72 <meta name="googlebot" content="index, follow, follow" />
73 <meta name="robots" content="index, follow" />
74 <meta name="robots" content="max-image-preview:large">
75 <meta name="language" content="id" />
76 <meta name="geo.country" content="id" />
77 <meta http-equiv="content-language" content="In-Id" />
78 <meta name="geo.placename" content="Indonesia" />
79 <meta property="og:locale" content="en_GB" />
80

```

Gambar 5.2 File head.php

b. Menemukan *Bug*

Berdasarkan panduan dari Acuneti WVS, *bug* terdapat pada bagian *head* HTML, tepat nya pada *meta* data properti untuk *thumbnail* WhatsApp.

```

31 <!-- Website to visit when clicked in fb or WhatsApp -->
32 <meta property="og:url" content="https://itp.ac.id/berita/arsip/?onmouseover='POKC(9759)'bad="">
33 <meta name="author" content="Institut Teknologi Padang, PUSTIKom">
34 <meta name="googlebot-news" content="index, follow, follow" />
35 <meta name="googlebot" content="index, follow, follow" />

```

Gambar 5.3 Temuan Bug Pada File Head.php

c. Memperbaiki *Bug*

Untuk memperbaiki dan pengamanan dari serangan XSS dapat dilakukan dengan beberapa cara berikut ini:

1. Untuk keamanan inputan data maka mengatasi serangan ini dengan melakukan aktivasi XSS *filtering* dan bisa juga dengan memberikan regex atau karakter yang diperbolehkan untuk diolah didalam *script* PHP. Dalam kasus ini XSS ditanganui dengan cara input dari semua *method* dilakukan *filtering* dengan fungsi berikut.

```
$this->security->xss_clean(preg_replace("/[^a-zA-Z0-9\ ]/", "", $VariableInput));
```

Gambar 5.4 Fungsi Filtering XSS

Pada *script* diatas digunakan fungsi `xss_clean` yang merupakan bawaan dari *framework* CodeIgniter untuk memfilter karakter yang di input. Sedangkan fungsi `preg_replace` digunakan untuk fungsi REGEX atau *regular expression* membatasi jenis karakter yang bisa diterima dari inputan. Pada kasus ini REGEX yang digunakan adalah hanya bisa menginputkan karakter a-z, A-Z dan 0-9. Disisi konfigurasi aplikasi cukup dengan menghidupkan *config global xss filtering* dengan *true* pada *file* `config.php`.

```
$config['global xss filtering'] = true;
```

Gambar 5.5 Global XSS Filtering

Pada *header* halaman dimana lokasi yang di *inject* XSS, terdapat kesalahan *scripting* dimana untuk mengambil parameter *url* tidak perlu di gunakan *script* `$_SERVER['REQUEST_URI']` yang akan menangkap semua inputan *method* GET pada *url* yang menyebabkan semua inputan tersebut ter-*inject* ke HTML. Seharus nya untuk mendapatkan *url* aktif cukup dengan fungsi `current_url()` seperti Gambar 4.14 berikut.

```
content="https://<?=$_SERVER['SERVER_NAME'].$_SERVER['REQUEST_URI'] ?>">
content="<?= current url() ?>">
```

Gambar 5.6 Global XSS Filtering

d. Tabel Perbaikan

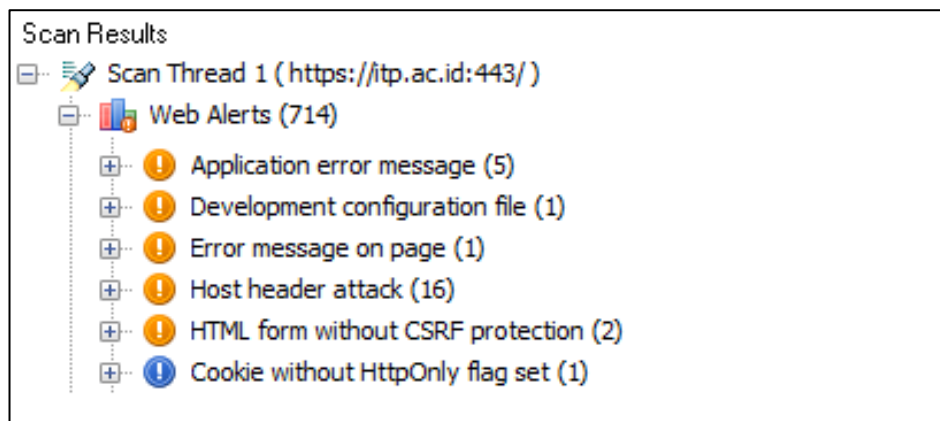
Berikut Tabel perbaikan pada *vulnerability* XSS.

Tabel 5.1 Perbaikan Bug XSS

No	Bug	Solusi
1.	XSS Injection Meta Head HTML	1. Input XSS Filtering 2. Fungsi Global XSS Filtering 3. Mengganti <i>script Request URI</i> dengan <i>current_url()</i>

e. Hasil

Hasil dari *scanning* ulang menunjukkan untuk *vulnerability* jenis *Cross Site Scripting* (XSS) sudah berhasil diperbaiki, dengan bukti sudah tidak ada lagi *alert* dari Acunetix WVS.



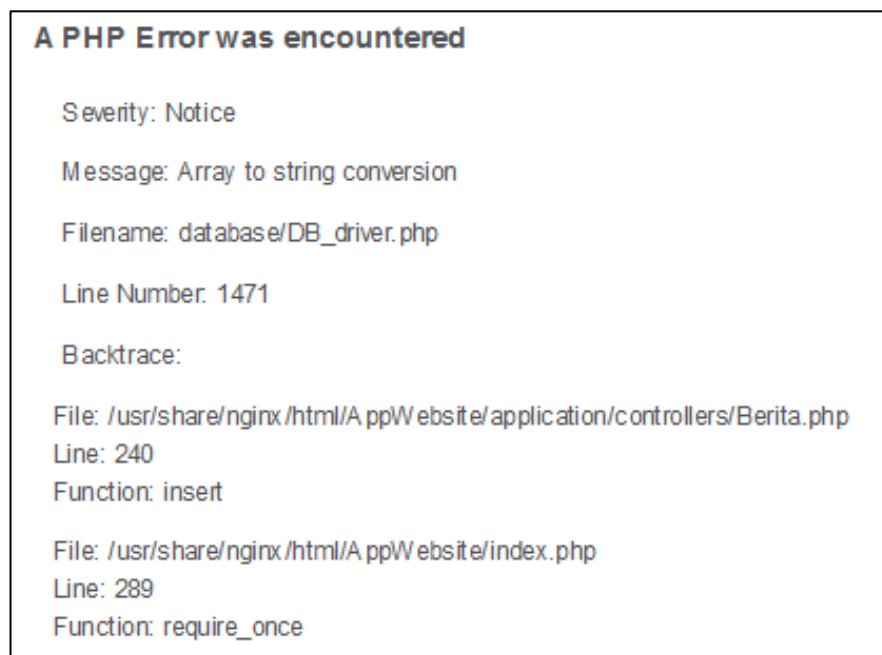
Gambar 5.7 Hasil Perbaikan XSS

2. Application Error Messages

Application Error Messages merupakan halaman yang berisi pesan kesalahan atau peringatan. Kesalahan aplikasi atau pesan peringatan dapat mengekspos informasi sensitif tentang cara kerja internal aplikasi kepada penyerang, informasi ini dapat digunakan untuk meluncurkan serangan lebih lanjut.

Mengatasi serangan ini dengan memastikan *report error* atau yang sering di panggil sebagai *error reporting* dapat di set menjadi 0 yang dimana nanti *error script* tidak akan ditampilkan ke *user*. Adapun langkah-langkah penyelesaiannya sebagai berikut:

- a. Membuka *file* index.php
- b. Menemukan *bug*
- c. Memperbaiki *bug*
- d. Tabel perbaikan
- e. Hasil



Gambar 5.8 Application Error Messages

Berikut penjelasan dari langkah-langkah diatas:

- a. Membuka *File*

Untuk menemukan *file* index.php maka perlu masuk kedalam direktori aplikasi pada *root folder* > index.php. pada *file* ini akan ditemukan konfigurasi untuk mengatur fungsi *error reporting*.

```

54  * NOTE: If you change these, also change the error_reporting() code below
55  */
56  define('ENVIRONMENT', isset($_SERVER['CI_ENV']) ? $_SERVER['CI_ENV'] : 'development');
57
58  /*
59  -----
60  * ERROR REPORTING
61  -----
62  *
63  * Different environments will require different levels of error reporting.
64  * By default development will show errors but testing and live will hide them.
65  */
66  switch (ENVIRONMENT)
67  {
68      case 'development':
69          error_reporting(-1);
70          ini_set('display_errors', 1);
71          break;
72
73      case 'testing':
74      case 'production':
75          ini_set('display_errors', 0);
76          if (version_compare(PHP_VERSION, '5.3', '>='))
77          {
78              error_reporting(E_ALL & ~E_NOTICE & ~E_DEPRECATED & ~E_STRICT & ~E_USER_NOTICE & ~E_USER_DEPRECATED);
79          }
80          else
81          {
82              error_reporting(E_ALL & ~E_NOTICE & ~E_STRICT & ~E_USER_NOTICE);
83          }
84          break;
85
86      default:
87          header('HTTP/1.1 503 Service Unavailable.', TRUE, 503);
88          echo 'The application environment is not set correctly.';
89          exit(1); // EXIT_ERROR
90  }

```

Gambar 5.9 File index.php

b. Menemukan *Bug*

Bug yang terdapat pada kasus ini adalah kesalahan dalam mengatur fungsi *error reporting*, oleh karena itu maka dicari posisi *script* yang mengatur fungsi tersebut seperti pada Gambar 5.10 berikut.

```

54  * NOTE: If you change these, also change the error_reporting() code below
55  */
56  define('ENVIRONMENT', isset($_SERVER['CI_ENV']) ? $_SERVER['CI_ENV'] : 'development');
57
58  /*
59  -----
60  * ERROR REPORTING
61  -----
62  *
63  * Different environments will require different levels of error reporting.
64  * By default development will show errors but testing and live will hide them.
65  */
66  switch (ENVIRONMENT)
67  {
68      case 'development':
69          error_reporting(-1);
70          ini_set('display_errors', 1);
71          break;
72
73      case 'testing':
74      case 'production':
75          ini_set('display_errors', 0);
76          if (version_compare(PHP_VERSION, '5.3', '>='))
77          {
78              error_reporting(E_ALL & ~E_NOTICE & ~E_DEPRECATED & ~E_STRICT & ~E_USER_NOTICE & ~E_USER_DEPRECATED);
79          }
80          else
81          {
82              error_reporting(E_ALL & ~E_NOTICE & ~E_STRICT & ~E_USER_NOTICE);
83          }
84          break;
85
86      default:
87          header('HTTP/1.1 503 Service Unavailable.', TRUE, 503);
88          echo 'The application environment is not set correctly.';
89          exit(1); // EXIT_ERROR
90  }

```

Gambar 5.10 Pengaturan *Error Reporting*

c. Memperbaiki *Bug*

Untuk menangani kasus ini dilakukan perbaikan pada *query* yang terdapat *error*, dan untuk pencegahannya, fungsi dari *error_reporting* dijakikan 0 atau pada kasus ini settingan pada konfigurasi *environment* menjadi *production* seperti *script* pada Gambar 5.11 dibawah ini

```

54 * NOTE: If you change these, also change the error_reporting() code below
55 */
56 define('ENVIRONMENT', isset($_SERVER['CI_ENV']) ? $_SERVER['CI_ENV'] : 'production');
57
58 /*

```

Gambar 5.10 Konfigurasi *Environment* Aplikasi

d. Tabel Perbaikan

Berikut Tabel perbaikan pada *vulnerability Application Error Messages*.

Tabel 5.2 Perbaikan *Bug Application Error Messages*

No	Bug	Solusi
1.	Kesalahan pengaturan <i>enviromtment</i> aplikasi	Mengatur <i>enviromtment</i> aplikasi menjadi <i>production</i> agar fungsi <i>error reporting</i> menjadi 0

e. Hasil

Hasil dari *scanning* ulang menunjukkan untuk *vulnerability* jenis *Application Error Messages* sudah berhasil diperbaiki, dengan bukti sudah tidak ada lagi *alert* dari Acunetix WVS.



Gambar 5.11 Hasil Perbaikan *Application Error Message*

3. *Development Configuration File*

Development configuration file merupakan *file* yang berisi konfigurasi dari *environment* aplikasi. Biasanya pada *file* ini berisi konfigurasi yang bersifat penting seperti koneksi dan lain lain. Pada Acunetix WVS, *file* yang berisi *keyword* “*description*” dianggap sebagai *file* yang rentan jika dapat diakses oleh *attacker*. Adapun langkah-langkah penyelesaiannya sebagai berikut :

- a. Menemukan *file* composer.json
- b. Menemukan bug
- c. Memperbaiki bug
- e. Tabel perbaikan
- f. Hasil

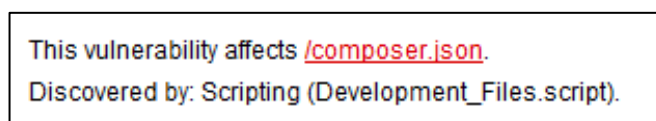


Gambar 5.12 *Development Configuration File*

Berikut penjelasan dari langkah-langkah diatas:

- a. Menemukan *File* composer.json

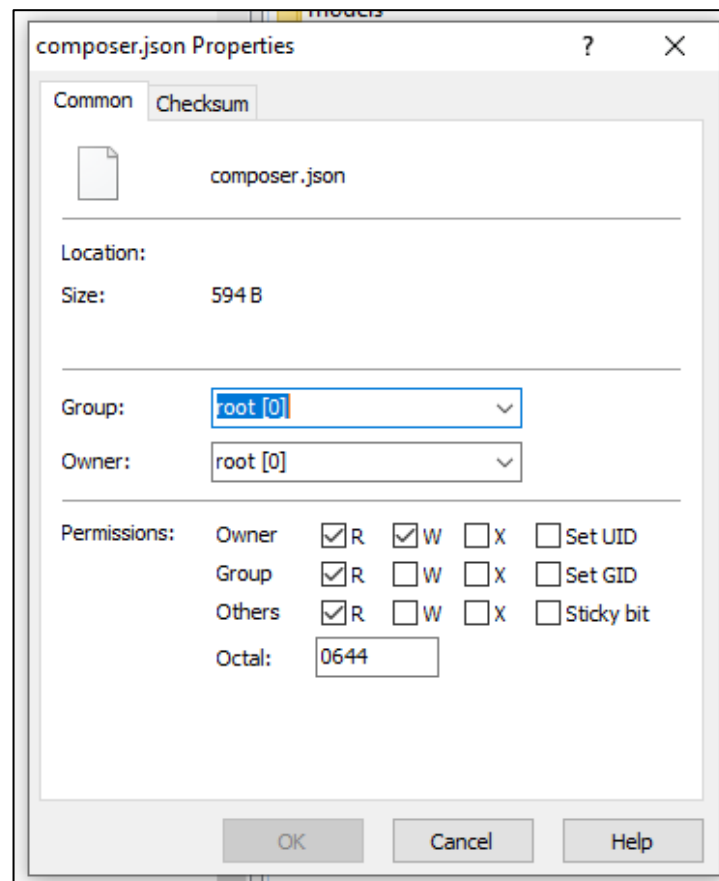
Berdasarkan petunjuk dari Acunetix WVS, *file* composer.json berada pada *root folder* dari aplikasi seperti Gambar 5.13 berikut.



Gambar 5.13 Lokasi *File* composer.json

b. Menemukan bug

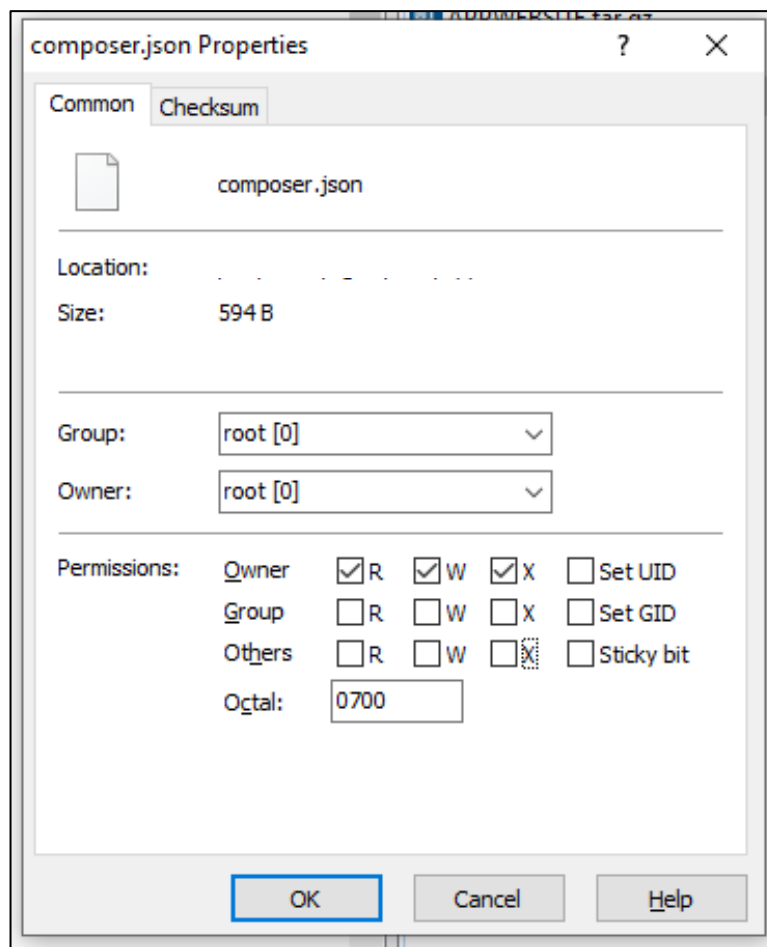
Untuk melihat *bug* yang terjadi pada *file* ini dapat dengan melihat *properties* dari *file* ini dengan meng-klik kanan *file* > *propeties*. Disini akan terlihat *permission file* memiliki *octal* 644 yang dimana *group* dan *others* masih bsa melihat *file* ini.



Gambar 5.14 Permission *File* composer.json

c. Memperbaiki *bug*

Pada kasus ini ditemukan *file* konfigurasi yaitu *composer.json* yang dapat diakses oleh *user* secara gamblang. Untuk menyelesaikan permasalahan ini dapat diberikan settingan hak akses *file* dengan oktal 700 yang berfungsi agar *file* ini dapat diakses ditulis dan dieksekusi hanya oleh *owner* saja.



Gambar 5.15 Pengaturan *Permission File*

d. Tabel Perbaikan

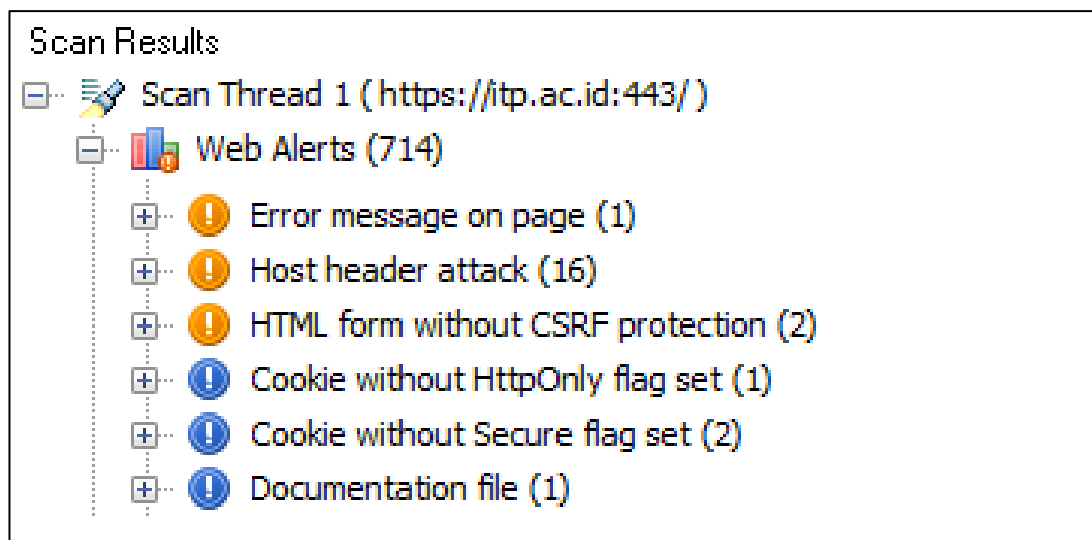
Berikut Tabel perbaikan pada *vulnerability Development Configuration File*.

Tabel 5.3 Perbaikan *Bug Development Configuration File*

No	Bug	Solusi
1.	<i>Permission File</i> composer.json	Merubah <i>permission file</i> dari 644 menjadi 700

e. Hasil

Hasil dari *scanning* ulang menunjukkan untuk *vulnerability* jenis *Development Configuration File* sudah berhasil diperbaiki, dengan bukti sudah tidak ada lagi *alert* dari Acunetix WVS



Gambar 5.16 Hasil Perbaikan *Development Configuration File*

4. *Error Message on Page*

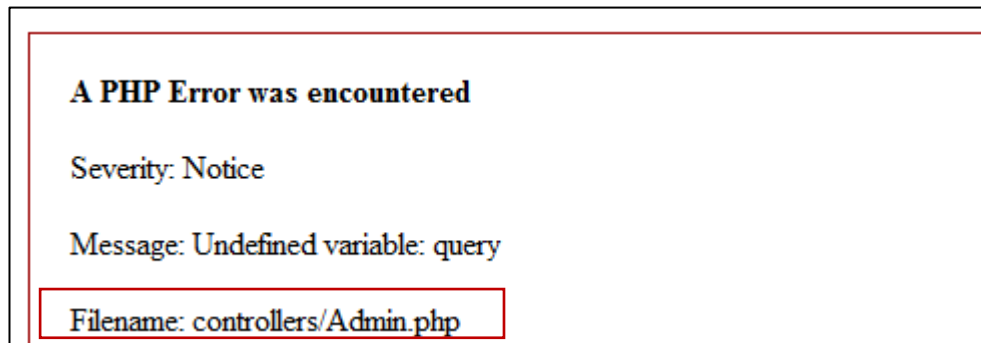
Error message on page merupakan pesan *error* yang muncul pada kesalahan sistem, baik itu pada sisi *scripting* atau *query database*. Muncul nya *error* ini merupakan celah bagi *attacker* untuk menemukan celah. Biasanya *attacker* Menemukan kesalahan atau *injection* pada *query database* dengan tujuan menampilkan *script query* apa yang di gunakan. Dengan munculnya *error* ini *attacker* akan lebih mudah men-*inject* SQL atau yang lebih sering disebut SQL *Injection*. Pada kasus ini, *controller* admin dari fungsi *datatable* tidak memiliki fungsi *index*, yang menyebabkan munculnya pesan *error*. Seharusnya ketika *controller* ini diakses tanpa fungsi, harusnya akan diarahkan ke fungsi *index*. Adapun langkah-langkah penyelesaiannya sebagai berikut :

- a. Membuka *file controller* admin.php
- b. Menemukan *bug*
- c. Memperbaiki *bug*
- d. Tabel perbaikan
- e. Hasil

Berikut penjelasan dari langkah-langkah diatas:

a. Membuka *File*

Untuk menemukan *file* index.php maka perlu masuk kedalam direktori aplikasi pada *root folder* > *controller* > Admin.php. pada *file* akan ditambahkan fungsi index.



Gambar 5.17 Lokasi File Controller Admin

b. Menemukan *Bug*

Bug yang terdapat pada kasus ini adalah tidak adanya fungsi *index* pada *controller* ini yang menyebabkan jika *file* ini diakses secara langsung akan menampilkan pesan *error* karena akan menjalankan fungsi yang pertama yaitu *datatable()*.

```

16 class Admin extends CI_Controller
17 {
18
19     function __construct()
20     {
21         parent::__construct();
22         $this->load->model('Models');
23         $this->load->helper('global');
24         $this->load->library('library');
25         $this->load->model('Konten');
26     }
27
28
29     public function datatable()
30     {
31
32         $this->load->model('Datatable');

```

Gambar 5.18 File Controller Admin

c. Memperbaiki *Bug*

Untuk menangani kasus ini dilakukan penambahan fungsi *controller* index dan diarahkan ke halaman 404 seperti Gambar 5.14 berikut dan untukantisipasi kemungkinan akan terjadinya muncul pesan *error* pada pengaturan *enviromtment* aplikasi disetting menjadi *production* agar *error reporting* tidak ditampilkan.

```

28      public function index()
29      {
30          show_404();
31      }

```

Gambar 5.19 Penambahan *Controller Index*

d. Tabel Perbaikan

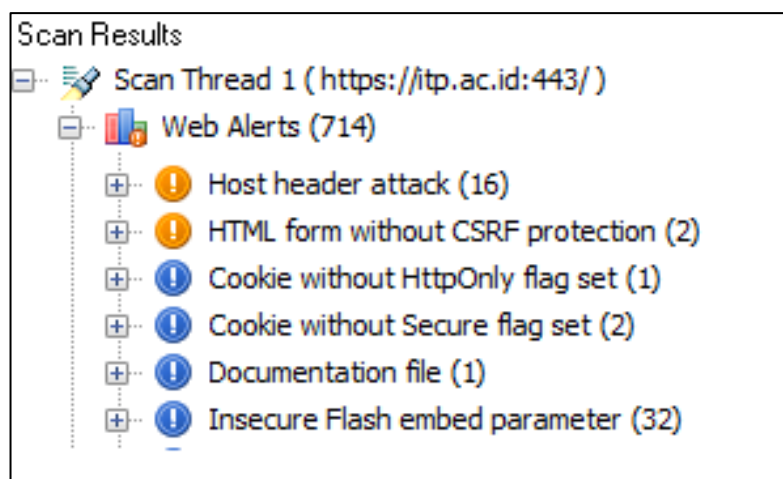
Berikut Tabel perbaikan pada *vulnerability Application Error Messages*.

Tabel 5.4 Perbaikan *Bug Error Mesaage on Page*

No	Bug	Solusi
1.	<ol style="list-style-type: none"> Kesalahan pengaturan <i>enviromtment</i> aplikasi Tidak ada <i>index</i> pada fungsi <i>controller</i> 	<ol style="list-style-type: none"> Mengatur <i>enviromtment</i> aplikasi menjadi <i>production</i> agar fungsi <i>error reporting</i> menjadi 0 Menambagkan fungsi <i>index</i> pada <i>controler</i> admin.

e. Hasil

Hasil dari *scanning* ulang menunjukkan untuk *vulnerability* jenis *Application Error Messages* sudah berhasil diperbaiki, dengan bukti sudah tidak ada lagi *alert* dari Acunetix WVS



Gambar 5.20 Hasil Perbaikan *Error Message On Page*

5. *Host Header Attack*

Host header merupakan yang menentukan situs *web* atau aplikasi *web* mana yang harus memproses permintaan HTTP yang masuk. *Server web* menggunakan nilai *header* untuk mengirim permintaan ke situs *web* atau aplikasi *web* yang ditentukan. Setiap aplikasi *web* yang dihosting di alamat IP yang sama biasanya disebut sebagai *host virtual*. Dampak dari *host header* menyebabkan penyerang dapat memanipulasi *header host* seperti yang terlihat oleh aplikasi web dan menyebabkan aplikasi merespon dengan cara yang tidak terduga. Adapun langkah-langkah penyelesaiannya sebagai berikut :

- a. Membuka *file controller* admin.php
- b. Menemukan *bug*
- c. Memperbaiki *bug*
- d. Tabel perbaikan
- e. Hasil

Berikut penjelasan dari langkah-langkah diatas:

- a. Membuka *file*

Dilihat dari temuan *bug*, lokasi *injection* yang dilakukan berada pada *head* HTML yang berisi *meta* data. Pada aplikasi ini *file head* berada pada *file head.php*.

```

1 <!-- type name -->
2 <html lang="eng">
3
4
5 <head>
6   <meta charset="utf-8">
7   <meta name="viewport" content="width=device-width, initial-scale=1">
8   <meta name="keywords" content="Universitas di Padang, Perguruan Tinggi Terbaik d
9   <meta name="description" content="Institut Teknologi Padang (ITP) adalah salah s
10
11 <meta name="msapplication-TileImage" content="https://itp.ac.id/assets/images/fa
12
13 <!-- fb & Whatsapp -->
14
15 <!-- Site Name, Title, and Description to be displayed -->
16 <meta property="og:site_name" content="Institut Teknologi Padang">
17 <meta property="og:title" content="Institut Teknologi Padang">
18 <meta property="og:description" content="Institut Teknologi Padang (ITP) adalah
19
20 <!-- Image to display -->
21 <!-- Replace «example.com/image01.jpg» with your own -->
22 <meta property="og:image" itemprop="image" content="https://itp.ac.id/assets/ima
23 <meta property="og:image:url" itemprop="image" content="https://itp.ac.id/assets
24 <meta property="og:image:type" content="image/png" />
25 <meta property="og:image:width" content="300">
26 <meta property="og:image:height" content="300">
27
28 <!-- No need to change anything here -->
29 <meta property="og:type" content="website" />

```

Gambar 5.21 File head.php

b. Menemukan *Bug*

Bug yang ditemukan pada kasus ini adalah, ketika fungsi dari *base url* yang seharusnya berisikan <https://itp.ac.id>, dapat diganti oleh *attacker* dan diganti dengan *ste url* lain. Seharusnya ketika nilai dari *base url* tidak sama dengan yang diatur maka harus ada *handler* yang berjalan.

```

<!-- Custom styles for this template -->
<link href="https://evilhostRkhAhstZ.com/assets/administrator/css/sb-admin-2.min.css" rel="stylesheet" />
<link href="https://evilhostRkhAhstZ.com/assets/administrator/css/loading.css" rel="stylesheet" />
<link href="https://evilhostRkhAhstZ.com/assets/administrator/css/extend.css" rel="stylesheet" />

```

Gambar 5.22 Host Header Attack

c. Memperbaiki *Bug*

Mengatasi serangan *host header*, *aplication* web harus menggunakan `SERVER_NAME` bukan *header host* dan harus membuat *vhost dummy* yang menangkap semua permintaan dengan pembaca *host* yang tidak dikenal. Pada Kasus ini dibutuhkan konfigurasi pada sisi *webserver* dan pada sisi *script*. Berikut konfigurasi yang dilakukan.

```

server_name itp.ac.id www.itp.ac.id; # Server name disesuaikan

if ( $host !~* ^(itp.ac.id|www.itp.ac.id)$ ) {
    return 444;
}
if ( $http_host !~* ^(itp.ac.id|www.itp.ac.id)$ ) {
    return 444;
}

```

Gambar 5.23 Konfigurasi Header Pada NGINX

Pada sisi aplikasi, juga diberikan *script* yang sama , tetapi *return* yang di berikan adalah *redirect* ke 404 seperti Gambar 5.19 berikut.

```

if (base_url() != "https://itp.ac.id/") {
    show_404();
}
$domains = ['itp.ac.id'];
if (!in_array($_SERVER['SERVER_NAME'], $domains)) {
    show_404();
}

```

Gambar 5.24 Konfigurasi Header Pada Aplikasi

d. Tabel Perbaikan

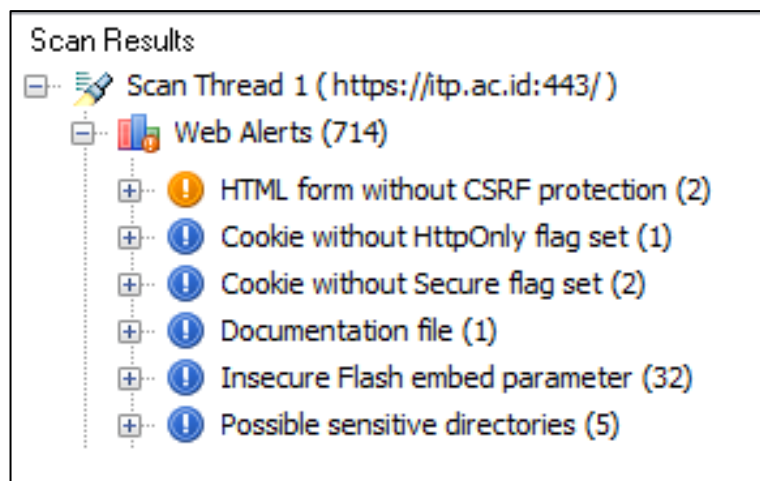
Berikut Tabel perbaikan pada *vulnerability Host Header Attack*.

Tabel 5.5 Perbaikan Bug Host Header Attack

No	Bug	Solusi
1.	Tidak fungsi yang memvalidasi <i>server name</i> atau <i>base url</i> tidak sesuai	<ol style="list-style-type: none"> 1. Mendefinisikan <i>header</i> pada <i>nginx.conf</i> beserta <i>handler</i> 2. Mendefinisikan <i>header</i> pada <i>header.php</i> beserta <i>handler</i>

e. Hasil

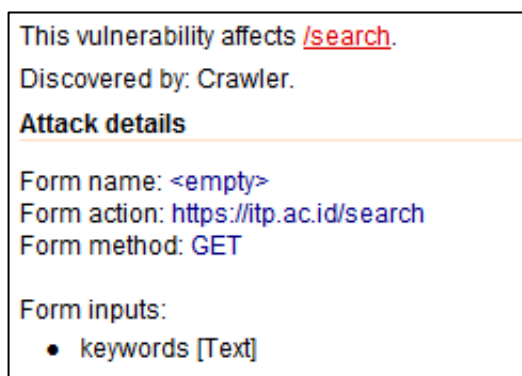
Hasil dari *scanning* ulang menunjukkan untuk *vulnerability* jenis *Host Header Attack* sudah berhasil diperbaiki, dengan bukti sudah tidak ada lagi *alert* dari Acunetix WVS.



Gambar 5.25 Hasil Perbaikan *Host Header Attack*

6. HTML Form Without CSRF Protection

HTML form without CSRF protection merupakan Pemalsuan permintaan lintas situs yang dikenal sebagai serangan satu klik atau CSRF atau XSRF yang jenis eksploitasinya berbahaya dari situs web di mana perintah yang tidak sah ditransmisikan dari pengguna yang dipercaya situs web. Penyerang dapat memaksa pengguna aplikasi web untuk melakukan tindakan yang dipilih penyerang. Tidak adanya pengamanan CSRF dapat memberikan celah *attacker* untuk melakukan *passing* data dari *form* lain dengan tujuan yang sama dengan *form* target. Hal ini membuat *attacker* akan leluasa untuk melakukan *injection* atau *pentest*. Kesalahan memilih metode pengiriman data juga menjadi masalah pada CSRF, pada kasus ini metode yang dipakai adalah GET sedangkan CSRF adalah *form* berbasis POST *handler*.



Gambar 5.26 HTML Form Without CSRF Protection


```

<div class="container mt-3">
  <div class="row">
    <div class="col-lg-12 mt-5">
      <form action="<?php echo base_url('search')?>">
        <div class="search-wrapper active">
          <div class="input-holder">
            <input type="text" name="keywords" class="search-input">
            <button class="search-icon"><span></span></button>
          </div>
          <!-- <span class="close" onclick="searchToggle(this)"></span>
        </div>
      </form>
    </div>
  </div>
</div>

```

Gambar 5.28 File v_search.php

b. Menemukan Bug

Bug yang terdapat pada kedua kasus ini adalah metode *form* yang digunakan adalah *method* GET yang dimana *form* tersebut tidak bisa diamankan dengan dari CSRF.

```

</button>
<form action="<?php echo base_url('search') ?>" >
  <label for="keywords" class="visuallyhidden">Search</label>
  <input type="text" name="keywords" class="search-input">
  <button class="search-icon"><span></span></button>
</form>
</div>
</div>

```

Gambar 5.29 File v_search.php

c. Memperbaiki Bug

Untuk menyelesaikan permasalahan ini dilakukan pengaktifan CSRF *protection* pada file *config.php* dan Kedua *form* tersebut diganti dengan *method* POST dengan penulisan *tag form* yang telah di standarisasi oleh fungsi CSRF *protection* agar *form* tersebut memiliki token validasi ketika di lakukan submit.

```

460 $config['csrf_protection'] = true;
461 $config['csrf_token_name'] = 'Token';
462 $config['csrf_cookie_name'] = 'CToken';
463 $config['csrf_expire'] = 7200;
464 $config['csrf_regenerate'] = true;

```

Gambar 5.30 Aktivasi Fungsi CSRF Protection

```

<div class="container mt-3">
  <div class="row">
    <div class="col-lg-12 mt-5">
      <?php echo form_open(base_url('search'), ' name="search" method="POST"') ?>
      <div class="search-wrapper active">
        <div class="input-holder">
          <input type="text" name="keywords" class="search-input" value="<?
          <button class="search-icon"><span></span></button>
        </div>
        <!-- <span class="close" onclick="searchToggle(this, event);"></span>
      </div>
      <?php echo form_close(); ?>
    </div>
  </div>
</div>

```

Gambar 5.31 Mengganti Penulisan Tag Form File v_home.php

```

<div class="col-lg-12 mt-5">
  <?php echo form_open(base_url('search'), ' name="search" method="POST"') ?>
  <div class="search-wrapper active">
    <div class="input-holder">
      <input type="text" name="keywords" class="search-input" value="<?= $_SES
      <button class="search-icon"><span></span></button>
    </div>
    <!-- <span class="close" onclick="searchToggle(this, event);"></span> -->
  </div>
  <?php form_close(); ?>
</div>

```

Gambar 5.32 Mengganti Penulisan Tag Form File v_search.php

d. Tabel Perbaikan

Berikut Tabel perbaikan pada *vulnerability HTML form without CSRF protection*.

Tabel 5.6 Perbaikan Bug HTML Without CSRF Protection

No	Bug	Solusi
1.	Form pencarian menggunakan <i>method</i> GET sehingga tidak ada pengamanan terhadap CSRF	<ol style="list-style-type: none"> 1. Mengaktifkan fungsi CSRF <i>protection</i> pada file <i>config.php</i> 2. Merubah cara penulisan <i>tag form</i> dengan standar yang sudah memiliki CSRF <i>protection</i>

e. Hasil

Hasil dari *scanning* ulang menunjukkan untuk *vulnerability* jenis *vulnerability HTML form without CSRF protection*.

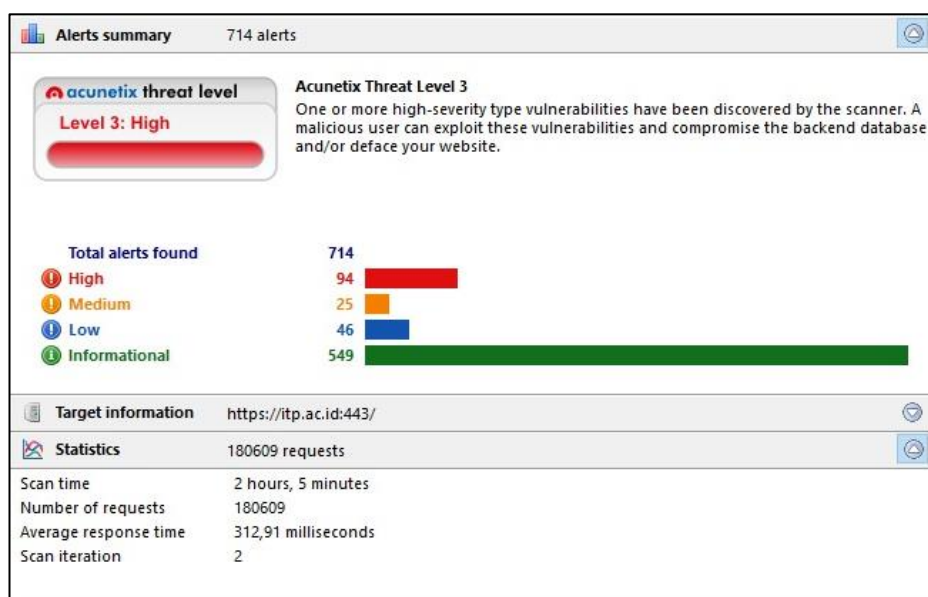
sudah berhasil diperbaiki, dengan bukti sudah tidak ada lagi *alert* dari Acunetix WVS.

Scan Results	Status
<ul style="list-style-type: none"> Scan Thread 1 (https://itp.ac.id:443/) Web Alerts (545) <ul style="list-style-type: none"> Cookie without HttpOnly flag set (1) Cookie without Secure flag set (2) Insecure Flash embed parameter (29) Possible sensitive directories (5) Possible sensitive files (3) Slow response time (4) Email address found (500) Possible username or password disclosure (1) 	Finished (545 alerts)

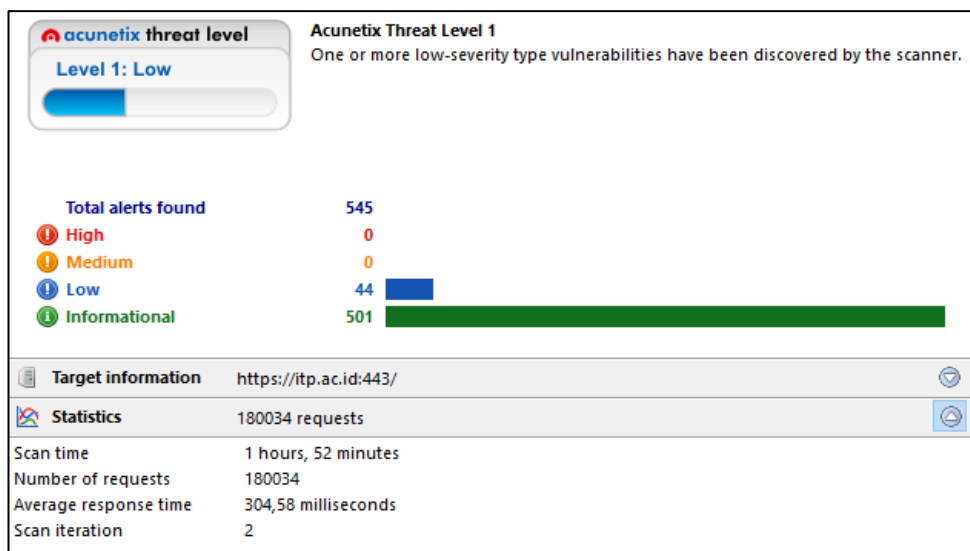
Gambar 5.33 Hasil Perbaikan HTML form Without CSRF Protection

5.3 Hasil

Setelah kegiatan perbaikan yang dilakukan, maka dilakukan *scanning* ulang yang berguna memastikan apakah perbaikan yang dimaksud sudah berfungsi dan menutupi celah keamanan yang terdeteksi pada *scanning* awal. Pada *scanning* ulang ini dilakukan selama 1 jam 52 menit dengan iterasi *scan* sebanyak 2 kali. Pada Gambar 5.35 dilihat hasil proses *scanning* ulang tidak ditemukan lagi *vulnerability* pada kategori *high* dan *medium*.



Gambar 5.34 Hasil Scanning Sebelum Perbaikan



Gambar 5.35 Hasil Scanning Setelah Perbaikan

Hasil yang diperoleh adalah *website* ITP sudah mendapatkan peringkat 1 atau *low threat level* yang dimana ini sudah dikategorikan aman berdasarkan Acunetix WVS karena pada *level 1 vulnerability* yang ditemukan bersifat rendah dan bersifat informasional saja.

Tabel 5.7 Perbandingan Data Setelah Perbaikan

No	Vulnerability	Jumlah Case	
		Sebelum Perbaikan	Setelah Perbaikan
1.	<i>Cross Site Scripting (XSS)</i>	94	0
2.	<i>Application Error Message</i>	5	0
3.	<i>Development Configuration File</i>	1	0
4.	<i>Error Message on Page</i>	1	0
5.	<i>Host Header Attack</i>	16	0
6.	<i>HTML Form Without CSRF Protection</i>	2	0
Jumlah Case		119	0

No	Bug/Alert/Port	Deskripsi	Perbaikan Bug/Alert/Port	Hasil
	22. /berita/detail/163		RI'] yang akan menangkap semua inputan <i>method</i> GET pada <i>url</i> yang menyebabkan semua inputan tersebut ter- <i>inject</i> ke HTML.	Berhasil
	23. /berita/detail/165			Berhasil
	24. /berita/detail/217			Berhasil
	25. /berita/detail/225			Berhasil
	26. /berita/detail/226			Berhasil
	27. /berita/detail/231			Berhasil
	28. /berita/detail/231/informasi-beasiswa-yayasan-t.a-2015			Berhasil
	29. /berita/detail/234			Berhasil
	30. /berita/detail/271			Berhasil
	31. /berita/detail/273			Berhasil
	32. /berita/detail/276			Berhasil
	33. /berita/detail/278			Berhasil
	34. /berita/detail/286			Berhasil
	35. /berita/detail/391			Berhasil
	36. /berita/detail/407			Berhasil
	37. /berita/detail/409			Berhasil
	38. /berita/detail/411			Berhasil
	39. /berita/detail/412			Berhasil
	40. /berita/detail/426			Berhasil
	41. /berita/detail/444			Berhasil
	42. /berita/detail/446			Berhasil
	43. /berita/detail/447			Berhasil
	44. /berita/detail/448			Berhasil
	45. /berita/detail/449			Berhasil
	46. /berita/detail/451			Berhasil
	47. /berita/detail/453			Berhasil

No	Bug/Alert/Port	Deskripsi	Perbaikan Bug/Alert/Port	Hasil
	75. /index.php/web 76. /kurikulum-mesins1 77. /kurikulum-sipil-s1 78. /pendidikan 79. /penelitian 80. /profil 81. /profile 82. /search (3) 83. /teknik-elektro-s1 84. /teknik-geodesi-s1 85. /teknik-informatika-s1 86. /teknik-lingkungan 87. /teknik-mesin-d3 88. /teknik-mesin-s1 89. /teknik-sipil-d3 90. /teknik-sipil-s1 91. /teknologi-listrik-d3 92. /visi-misi-sipils1			Berhasil Berhasil Berhasil Berhasil Berhasil Berhasil Berhasil Berhasil Berhasil Berhasil Berhasil Berhasil Berhasil Berhasil Berhasil Berhasil Berhasil Berhasil Berhasil Berhasil
2	<i>Application Error Message</i> 1. /berita/ (2) 2. /berita/ komentar (2) 3. /search (1)	<i>Application Error Messages</i> merupakan halaman yang berisi pesan kesalahan atau peringatan. Kesalahan aplikasi atau pesan peringatan dapat mengekspos informasi sensitif tentang cara kerja internal aplikasi kepada penyerang, informasi ini dapat digunakan untuk meluncurkan serangan lebih lanjut.	Mengatur envirointment aplikasi menjadi production agar fungsi error reporting menjadi 0	Berhasil Berhasil Berhasil

No	Bug/Alert/Port	Deskripsi	Perbaikan Bug/Alert/Port	Hasil
3	Development Configuration File 1. /composer.json	<i>Development configuration file</i> merupakan <i>file</i> yang berisi konfigurasi dari <i>environment</i> aplikasi. Biasanya pada <i>file</i> ini berisi konfigurasi yang bersifat penting seperti koneksi dan lain lain. Pada Acunetix WVS, <i>file</i> yang berisi <i>keyword</i> “ <i>description</i> ” dianggap sebagai <i>file</i> yang rentan jika dapat diakses oleh <i>attacker</i> .	Merubah permission <i>file</i> dari 644 menjadi 700	Berhasil
4	Error Message on Page 1. /berita/ (2) 2. /berita/ komentar (2) 3. /search (1)	<i>Error message on page</i> merupakan pesan <i>error</i> yang muncul pada kesalahan sistem, baik itu pada sisi <i>scripting</i> atau <i>query database</i> . Munculnya <i>error</i> ini merupakan celah bagi <i>attacker</i> untuk menemukan celah.	1. Mengatur <i>enviromtment</i> aplikasi menjadi <i>production</i> agar fungsi <i>error reporting</i> menjadi 0 2. Menambagkan fungsi <i>index</i> pada <i>controler admin</i> .	Berhasil Berhasil Berhasil
5	Host Header Attack 1. /berita/detail/446/monev-coe-mbkm-itp-digelar-secara-hybrid 2. /berita/detail/453/launching-	<i>Host header</i> merupakan yang menentukan situs <i>web</i> atau aplikasi <i>web</i> mana yang harus memproses permintaan HTTP yang masuk. <i>Server web</i> menggunakan nilai <i>header</i> untuk	1. Mendefinisikan <i>header</i> pada <i>nginx.conf</i> beserta <i>handler</i> 2. Mendefinisikan <i>header</i> pada <i>header.php</i> beserta <i>handler</i>	Berhasil Berhasil Berhasil

No	Bug/Alert/Port	Deskripsi	Perbaikan Bug/Alert/Port	Hasil
	sistem-informasi-mbkm-itp	mengirim permintaan ke situs <i>web</i> atau aplikasi <i>web</i> yang ditentukan. Setiap aplikasi <i>web</i> yang dihosting di alamat IP yang sama biasanya disebut sebagai <i>host virtual</i> .		Berhasil
3.	/berita/detail/457/komit--ini-inovasi-itp-untuk-keterbukaan-info...			Berhasil
4.	/berita/detail/460/visiting-professor-itp-and-umpedac			Berhasil
5.	/berita/detail/462/monev-pkk-m-tahun-2021			Berhasil
6.	/berita/detail/463/monev-coe-mbkm-itp-bahas-kendala-kegiatan			Berhasil
7.	/berita/detail/464/rektor-itp-dilantik-28-pejabat-struktural-period...			Berhasil
8.	/berita/detail/465/lpj-tahun-2021--rektor-itp:capaian-proker-...			Berhasil
9.	/berita/detail/467/raker-tahun-2022--itp-mantapkan-pengemb...			Berhasil
10.	/berita/detail/468/ft-itp-teken-perjanjian-kerja-sama-dengan-f...			Berhasil
11.	/berita/detail/469/itp-sosialisasikan-aturan-baru-bkd-ikd-			Berhasil
12.	/berita/detail/470/sambangi-			Berhasil

No	Bug/Alert/Port	Deskripsi	Perbaikan Bug/Alert/Port	Hasil
	itp--smkn-5-padang-upgrade-kerja... 13. /kerjasama 14. /mahasiswa 15. /pengabdian 16. /search			Berhasil Berhasil
6	HTML Form Without CSRF Protection 1. / 2. /search	HTML <i>form without CSRF protection</i> merupakan Pemalsuan permintaan lintas situs yang dikenal sebagai serangan satu klik atau CSRF atau XSRF yang jenis eksploitasinya berbahaya dari situs web di mana perintah yang tidak sah ditransmisikan dari pengguna yang dipercaya situs web. Penyerang dapat memaksa pengguna aplikasi web untuk melakukan tindakan yang dipilih penyerang.	1. Mengaktifkan fungsi CSRF protection pada file config.php 2. Merubah cara penulisan tag form dengan standar yang sudah memiliki CSRF protection	Berhasil Berhasil

5.3.1 Implementasi Sistem

Untuk implementasi sistem, data yang di inputkan kedalam sistem *knowledge transfer* adalah rekapan data dari analisa dan perbaikan yang dilakukan. Pada penelitian ini di inputkan enam jenis *vulnerability* beserta solusi perbaikan kedalam sistem.

Knowledge Transfer Application
Vulnerability Assessment for Web Application

List Data

Show 10 entries Search:

No	Vulnerability	Bugs	Detail
1.	XSS	Server Request URI	i
2.	Application Error Message	Error Message Handler	i
3.	Development Configuration File	File Development Only Owner Can Access	i
4.	Error Message on Page	Error Message Handler	i
5.	Host Header Attack	Header Application Injected and Changed to another site	i
6.	HTML Form Without CSRF Protectio	Handling CSRF Protection on search form	i

Showing 1 to 6 of 6 entries Previous 1 Next

Knowledge Transfer Application - Afif Zirwan @ 2022

Gambar 5.36 Aplikasi *Knowledge Transfer*

Bugs Detail

Vulnerability ini merupakan serangan *code injection* yang dilakukan pada sisi klien. Serangan *cross site scripting* biasanya digunakan untuk mencuri *cookie*, penyebaran *malware*, *session hijacking* atau pembajakan *session*, dan pemblokkan tujuan atau *malicious redirects* dengan mengeksekusi *script* berbahaya di *browser website* korban dengan memasukkan kode berbahaya pada halaman *website*. Serangan XSS ini dimungkinkan terdapat dalam *VBScript*, *activeX*, *flash*, dan bahkan *CSS*. Serangan *cross site scripting* (XSS) terjadi apabila data memasuki aplikasi web melalui sumber yang tidak tepercaya dan data disertakan dalam konten dinamis yang dikirim ke pengguna *website* tanpa divalidasi untuk konten berbahaya.

```

31 <!-- Website to visit when clicked in fb or WhatsApp-->
32 <meta property="og:url" content="https://itp.ac.id/berita/arsip/?onmouseover='POHC(9759)'bad=''"
33 <meta name="author" content="Institut Teknologi Padang, IUSTIKom">
34 <meta name="googlebot-news" content="index, follow, follow" />
35 <meta name="googlebot" content="index, follow, follow" />
36 <meta name="robots" content="index, follow" />
37

```

Solusi Perbaikan :

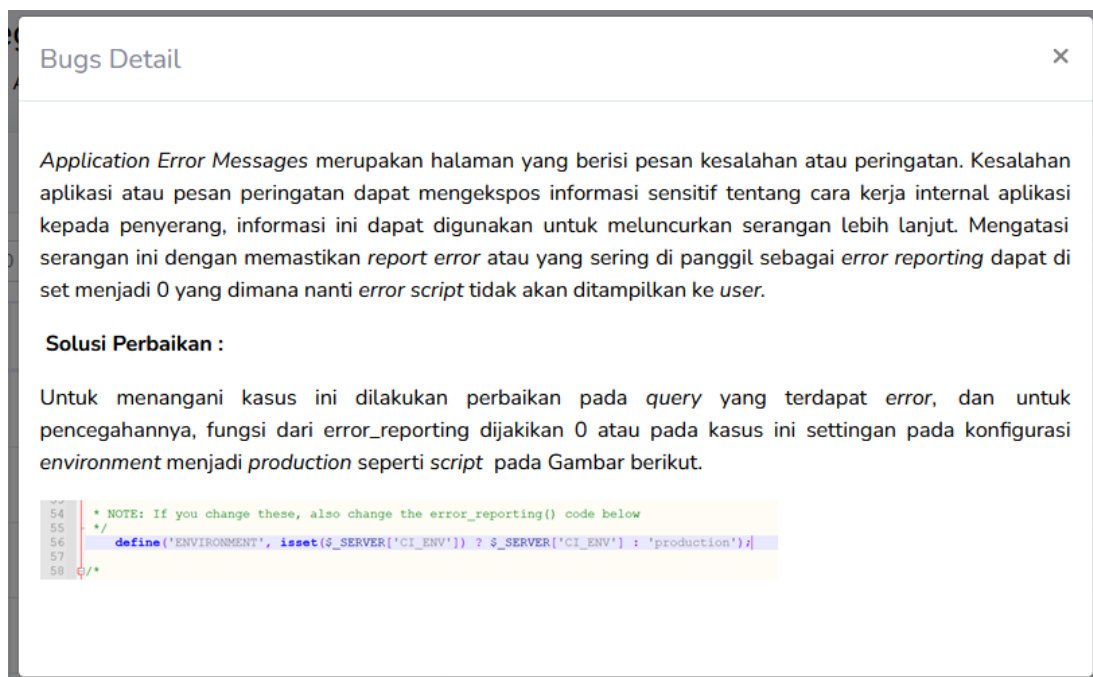
1. Untuk keamanan inputan data maka mengatasi serangan ini dengan melakukan aktivasi XSS *filtering* dan bisa juga dengan memberikan *regex* atau karakter yang diperbolehkan untuk diolah didalam *script PHP*. Dalam kasus ini XSS ditanganui dengan cara input dari semua *method* dilakukan *filtering* dengan fungsi berikut

```

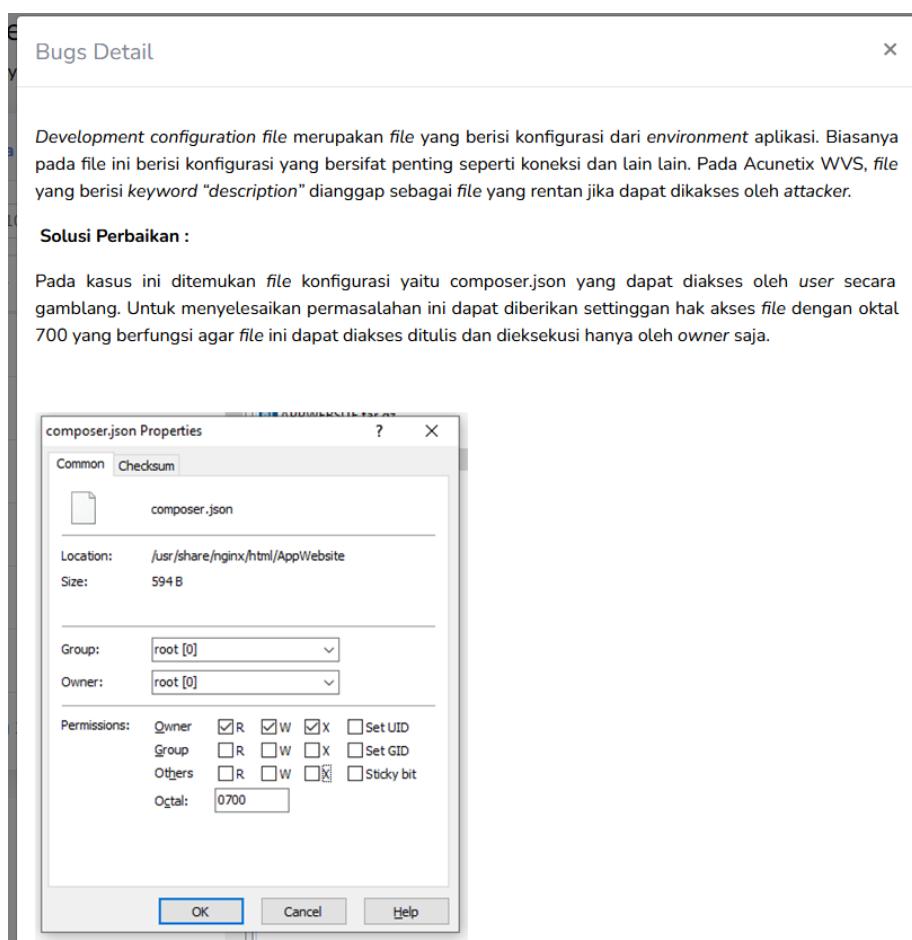
$this->security->xss_clean(preg_replace("/[^\a-zA-Z0-9\ ]/", "", $VariableInput));

```

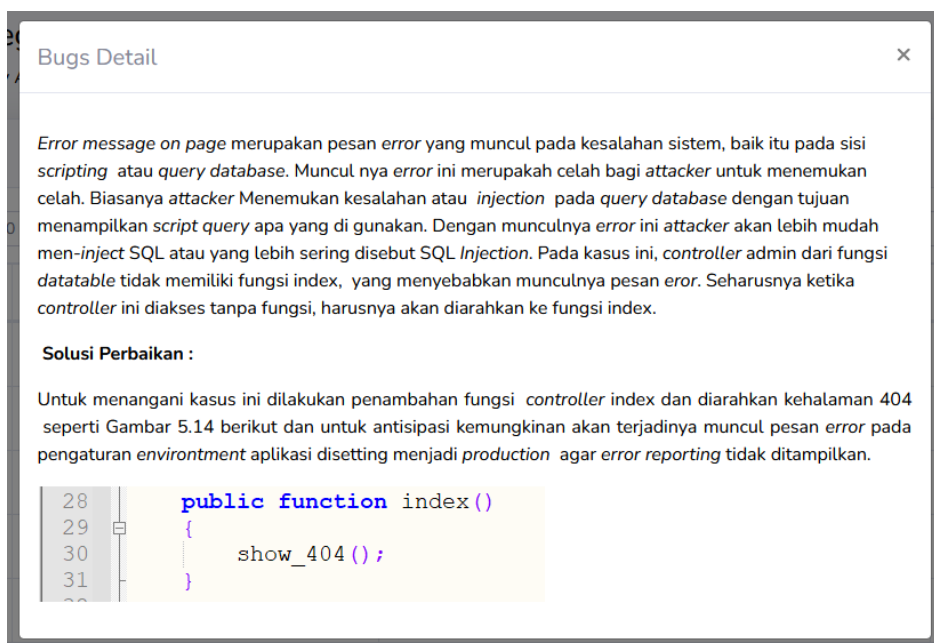
Gambar 5.37 Detail Informasi Perbaikan XSS



Gambar 5.38 Detail Informasi Perbaikan *Application Error Message*



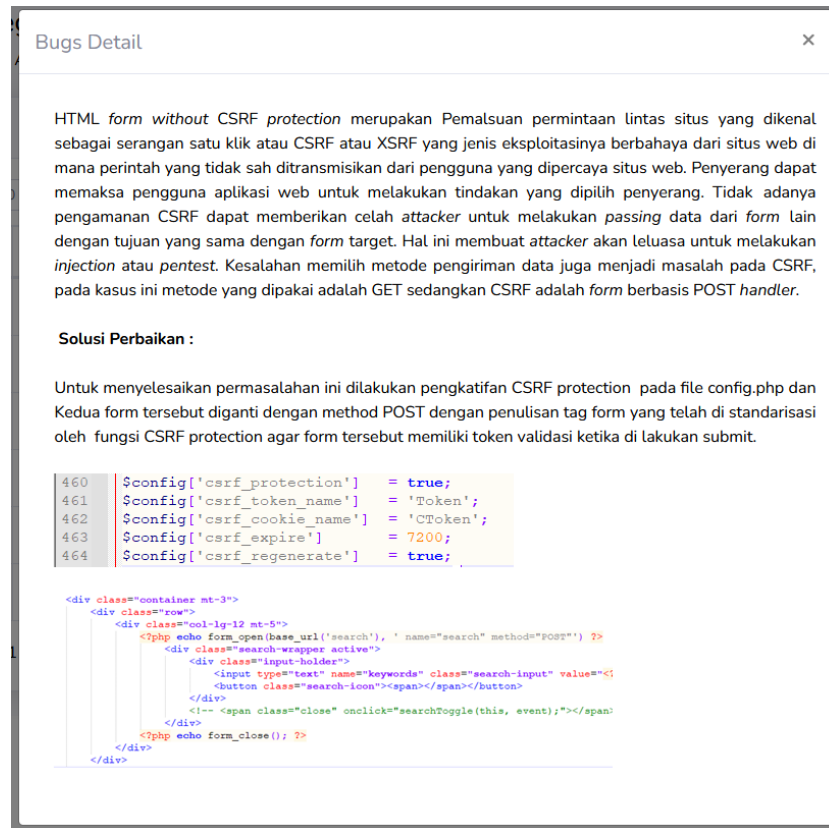
Gambar 5.39 Detail Informasi Perbaikan *Development Configuration File*



Gambar 5.40 Detail Informasi Perbaikan *Error Message In Page*



Gambar 5.41 Detail Informari Perbaikan *Host Header Attack*



Gambar 5.42 Detail Informasi Perbaikan HTML Without CSRF Protection

BAB VI

KESIMPULAN

6.1 Kesimpulan

Berdasarkan pengujian dan analisa *vulnerability* pada *website* ITP, *tools* Acunetix WVS dapat membantu dalam proses *assessment* dan perbaikan yang kemudian dapat disimpulkan bahwa:

1. Hasil dari *vulnerability assesment* awal menggunakan *tools* Acunetix WVS, *website* ITP mendapatkan *threat* pada *level* 3 yang termasuk kategori *high* dengan 2 kali iterasi *scanning*, ditemukan 119 *alert* atau celah yang terdiri yang terdiri dari 94 pada *level high* dan 25 pada *level medium*.
2. Berdasarkan analisa, perbaikan dan pengujian yang dilakukan pada penelitian ini terhadap *website* ITP, menghasilkan *threat level* sudah pada *level* 1, dimana pada *level high* jumlah celah sudah menjadi 0 dan *medium* juga sudah menjadi 0, yang dapat disimpulkan *website* ITP sudah tergolong aman dari celah keamanan.
3. Menggunakan *tools* Acunetix WVS dapat membantu *assesment* dan perbaikan untuk mengurangi *vulnerability* yang ditemukan.

6.2 Saran

Adapun saran dari penelitian ini yang dapat dijadikan pengembangan pada penelitian selanjutnya adalah:

1. Untuk mengetahui celah keamanan yang terdapat pada suatu *website* ataupun aplikasi berbasis *web* bukan hanya dengan *tools* Acunetix WVS. Oleh karena itu untuk kedepannya penelitian juga dapat diarahkan dengan *tools pentest* yang lain.

2. Dalam perbaikan dan penanganan celah yang ditemukan mungkin saja memiliki metode yang lebih baik, hal ini juga tidak menutup kemungkinan untuk dijadikan penelitian lanjutan berdasarkan *case* yang ditemukan pada penelitian ini.
3. Aplikasi *knowledge transfer* yang digunakan untuk menyimpan data perbaikan dapat dikembangkan lebih baik dari segi *interface* maupun *backend* agar lebih optimal.

DAFTAR PUSTAKA

- Ade Bastian, Harun Sujadi, & Latiful Abror. (2020). ANALISIS KEAMANAN APLIKASI DATA POKOK PENDIDIKAN (DAPODIK) MENGGUNAKAN PENETRATION TESTING DAN SQL INJECTION. *INFOTECH Journal*, 6, 65–70. <https://doi.org/10.31949/infotech.v6i2.848>
- Al Fajar, F. (2020). Analisis Keamanan Aplikasi Web Prodi Teknik Informatika Uika Menggunakan Acunetix Web Vulnerability. *Inova-Tif*, 3(2), 110. <https://doi.org/10.32832/inova-tif.v3i2.4127>
- Bustami, A., & Bahri, S. (2020). Ancaman, Serangan dan Tindakan Perlindungan pada Keamanan Jaringan atau Sistem Informasi: Systematic Review. In *Jurnal Pendidikan dan Aplikasi Industri (UNISTEK)* (Vol. 7, Issue 2). <https://doi.org/10.33592/unistek.v7i2.645>
- Efyz Zam. (2011). *Buku Sakti Hacker* (1st ed.). Mediakita.
- Goel, J. N., & Mehtre, B. M. (2015). Vulnerability Assessment & Penetration Testing as a Cyber Defence Technology. *Procedia Computer Science*, 57, 710–715. <https://doi.org/10.1016/j.procs.2015.07.458>
- Gupta, U., Raina, S., Verma, P., Singh, P., & Aggarwal, M. M. (2020). Web Penetration Testing. *International Journal for Research in Applied Science and Engineering Technology*, 8(5), 56–60.
- Irawadi Alwi, E., & Umar, F. (2020). Analisis Keamanan Website Menggunakan Teknik Footprinting dan Vulnerability Scanning. In *Informatics Journal* (Vol. 5, Issue 2). <https://doi.org/https://doi.org/10.19184/isj.v5i2.18941>
- Mayasari, R., Ridha, A. A., Juardi, D., & Baihaqi, K. A. (2020). Analisis Vulnerability pada Website Universitas Singaperbangsa Karawang menggunakan Acunetix Vulnerability. In *SYSTEMATICS* (Vol. 2, Issue 1). <https://doi.org/10.35706/sys.v2i1.3450>
- Orisa, M., & Ardita, M. (2021). VULNERABILITY ASSESMENT UNTUK MENINGKATKAN KUALITAS KEMANAN WEB. In *Jurnal MNEMONIC*

- (Vol. 4, Issue 1). <https://doi.org/10.36040/mnemonic.v4i1.3213>
- Pohan, Y. A. (2021). Meningkatkan Keamanan Webserver Aplikasi Pelaporan Pajak Daerah Menggunakan Metode Penetration Testing Execution Standar. *Jurnal Sistim Informasi Dan Teknologi*, 3, 1–6. <https://doi.org/10.37034/jsisfotek.v3i1.36>
- Priandoyo, A. (2006). Vulnerability Assessment untuk Meningkatkan Kesadaran Pentingnya Keamanan Informasi. *Jurnal Teknik Informatika Dan Sistem Informasi*, 1(2), 73–83.
- Retna Mulya, B. W., & Tarigan, A. (2018). Pemeringkatan Risiko Keamanan Sistem Jaringan Komputer Politeknik Kota Malang Menggunakan Cvss Dan Fmea. *ILKOM Jurnal Ilmiah*, 10(2), 190–200. <https://doi.org/10.33096/ilkom.v10i2.311.190-200>
- Riadi, I., Herman, & Ifani, A. Z. (2021). Optimasi Keamanan Web Server terhadap Serangan Broken Authentication Menggunakan Teknologi Blockchain. *JISKA (Jurnal Informatika Sunan Kalijaga)*, 6(3), 139–148. <https://doi.org/10.14421/jiska.2021.6.3.139-148>
- Riadi, I., Yudhana, A., & W, Y. (2020). Analisis Keamanan Website Open Journal System Menggunakan Metode Vulnerability Assessment. *Jurnal Teknologi Informasi Dan Ilmu Komputer*, 7(4), 853. <https://doi.org/10.25126/jtiik.2020701928>
- Syarifudin, I. (2018). Pentesting dan Analisis Keamanan Web Paud Dikmas. *Pentesting Dan Analisis Keamanan Web Paud Dikmas*, April, 2. <https://doi.org/10.5281/zenodo.1211847>
- T, G. S., & Sasikala D. (2019). Vulnerability Assessment of Web Applications using Penetration Testing. *International Journal of Recent Technology and Engineering*, 8(4), 1552–1556. <https://doi.org/10.35940/ijrte.b2133.118419>
- Ula, M. (2019). Evaluasi Kinerja Software Web Penetration Testing. *TECHSI - Jurnal Teknik Informatika*, 11(3), 336. <https://doi.org/10.29103/techsi.v11i3.1996>
- Wibowo, F., & Purwo Wicaksono, A. (2019). Uji Vulnerability pada Website Jurnal Ilmiah Universitas Muhammadiyah Purwokerto Menggunakan OpenVAS dan Acunetix WVS. *JURNAL INFORMATIKA*, 6(2), 212–218. <https://doi.org/10.31294/ji.v6i2.5925>