Evaluasi Kinerja Software *Web Penetration Testing*

Munirul Ula
Program Studi Sistem Informasi
Universitas Malikussaleh

Abstrak

Abstrak— Website sudah menjadi bagian penting dalam setiap aspek kehidupan kita sehari-hari. Dari belanja online hingga bersosialisasi, semuanya tersedia dalam satu klik melalui gatget. Setiap website adalah unik dengan caranya sendiri, mulai dari coding hingga eksekusi, tetapi secara umum di setiap website terdapat celah keamanan yang memudahkan tersusupi oleh para hacker. Oleh karena itu perlu dilakukan scanning celah keamanan yang ada pada sebuah website. Dalam artikel ini, berbagai macam program pendeteksi celah keamanan aplikasi website telah diperiksa dan dievaluasi secara terperinci untuk mengetahui program scanner mana yang paling cocok digunakan untuk mendeteksi kelemahan keamanan sebuah website. Program-program scanner keamanan tersebut memberikan informasi tentang cara melakukan berbagai skenario serangan terhadap website sampel. Artikel ini juga menunjukkan kelebihan dan kekurangan kinerja dari program yang diuji.

Kata Kunci : Aplikasi website *(WebApp)*, Pendeteksi Keamanan, Celah Keamanan, *Scanner* keamanan website

1. Pendahuluan

Internet sudah menjadi bagian penting dalam setiap aspek kehidupan kita sehari-hari. Dari belanja online hingga bersosialisasi semuanya tersedia dalam satu klik melalui gatget. Setiap website adalah unik dengan caranya sendiri, mulai dari *coding* hingga eksekusi, tetapi secara umum di setiap *website* adalah *bug*. *Bug* ini memudahkan peretas untuk masuk kedalam

DOI: https://doi.org/10.29103/techsi.v11i3.1996

server dan *system website*. Dalam artikel ini melalui pengujian penetrasi pada *website* menggunakan *software* penguji penetrasi yang berbeda kita dapat menemukan berbagai *bug* ini. Pengujian penetrasi akan membantu pengembang web dalam membangun aplikasi *website* yang kuat dan aman. Ini sangat penting untuk website mana pun karena bug memberi keuntungan bagi peretas untuk lebih mengeksploitasi aplikasi *website.*

Aplikasi website sulit untuk diamankan karena secara alami terbuka untuk umum, termasuk hacker. Selain itu, input ke aplikasi website berasal dari permintaan *HTTP*. Input ini sulit untuk diproses dengan benar. Validasi input yang tidak benar atau tidak ada akan menghasilkan Celah keamanan di aplikasi website. *Network vulnerability scanners, firewall* jaringan, dan penggunaan *Secure Socket Layer (SSL)* tidak sepenuhnya mengamankan *website. Gartner Group* memperkirakan bahwa lebih dari 70% serangan di *website* atau aplikasi *website* perusahaan terjadi di lapisan aplikasi, bukan lapisan jaringan atau *server*.

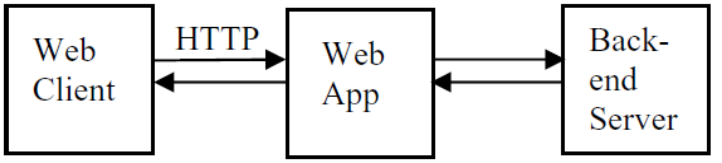
Aplikasi *website scanner* membantu kita mengurangi jumlah Celah Keamanan dalam aplikasi *website.* Singkatnya, Aplikasi *website scanner* mendeteksi halaman aplikasi *website* dan mencari Celah Keamanan dengan mensimulasikan serangan pada aplikasi tersebut. *BroAplikasi website scanner* dapat menemukan banyak Celah keamanan dan tidak dapat memberi jaminan bahwa aplikasi yang berdiri sendiri akan aman. APLIKASI *WEBSITE SCANNER* diimplementasikan pada tahap akhir siklus pengembangan perangkat lunak sebuah *website*. Keamanan harus dirancang dan terstruktur. Berbagai *best practice* harus digunakan pada saat pengembangan sebuah *website.*

2. Tinjauan Pustaka

2.1 Anatomi Aplikasi *Website*

Konsorsium Keamanan Aplikasi *website* mendefinisikan aplikasi website sebagai "aplikasi perangkat lunak yang dijalankan oleh server web yang menanggapi permintaan halaman web dinamis melalui *HTTP*".

DOI: https://doi.org/10.29103/techsi.v11i3.1996



Gambar 1. Diagram kerja aplikasi website biasa

Aplikasi *website* menyertakan *script* pada server web yang berinteraksi dengan basis data atau konten dinamis lainnya. Menggunakan infrastruktur Internet, aplikasi website memungkinkan penyedia layanan dan klien untuk berbagi dan mengarahkan informasi dengan tidak terikat kepada platform. Teknologi yang biasa digunakan untuk membuat aplikasi website adalah *PHP, Active Server Pages (ASP), Perl, Common Gateway Interface (CGI), Java Server Pages (JSP), JavaScript, ASP.NET, VBScript* dan lain-lain. Beberapa kategori luas dari teknologi aplikasi website adalah protokol komunikasi, format, bahasa script sisi-server dan sisi klien, *add-on browser add on,* dan *API server web.*

2.2 Celah Keamanan Aplikasi *Website*

Aplikasi website umumnya berinteraksi dengan pengguna melalui elemen formulir seperti tombol, kotak teks, dll. Dan variabel *GET* atau *POST*. Pemrosesan item data yang tidak benar dalam permintaan *HTTP* menyebabkan Celah keamanan paling kritis di aplikasi *website. SSL* tidak mencegah Celah keamanan karena menyediakan transfer data yang dianggap aman dan tidak memeriksa permintaan *HTTP*. Aplikasi website adalah titik arah untuk database yang menyimpan data dan aset aplikasi penting. Beberapa ancaman utama pada lapisan server basis data adalah injeksi *SQL,* pencurian akses server, dan serangan peretas kata sandi. Sebagian besar Celah Keamanan injeksi *SQL* disebabkan oleh validasi input yang lemah. Dan sebagian besar sistem aplikasi website menyimpan informasi sensitif dalam basis data atau dalam sistem file. Pengembang sering membuat kesalahan dalam teknik enkripsi untuk melindungi informasi ini. web menggunakan mekanisme terpisah untuk mempertahankan status sesi. Sesi adalah serangkaian interaksi antara pengguna dan web selama satu kunjungan ke website. Secara umum, manajemen sesi

DOI: https://doi.org/10.29103/techsi.v11i3.1996

dilakukan menggunakan string unik yang disebut *Session ID* yang dikirim ke server web dengan setiap permintaan. Sebagian besar script web mengelola sesi melalui variabel *GET* dan / atau *cookie.* Jika penyerang dapat menebak atau mencuri *ID sesi*, ia dapat mengubah sesi pengguna lain

3. Kerangka Kerja Dan *Scanner* Keamanan Website

Scanner keamanan website adalah program otomatis yang menganalisis web untuk Celah keamanan. Selain mencari Celah keamanan spesifik untuk aplikasi *website*, program juga mencari kesalahan dalam pengkodean perangkat lunak, seperti string input ilegal dan *buffer overflow*. *Scanner* keamanan website mencari aplikasi dengan menjelajahi halaman web dan melakukan tes penetrasi. Lebih sederhana, ini mensimulasikan serangan aplikasi website dan menganalisisnya secara efektif. Hasil dari penelusuran ini akan mendeteksi potensi bahaya dan kemudian mengevaluasi respons aplikasi web tersebut. Scanner tersebut melakukan berbagai jenis serangan. Serangan praktis, umumnya dikenal sebagai *fuzzing,* mengirimkan entri acak ke aplikasi dalam berbagai ukuran.

Pengujian penetrasi adalah pendekatan tes dengan metode kotak hitam. Keterbatasan dari pendekatan ini adalah ia tidak memeriksa kode sumber, jadi kecil kemungkinannya untuk mendeteksi celah keamanan pintu belakang yang ada. Namun, sangat cocok untuk mendeteksi masalah validasi input. Selain itu, kode sisi klien (*JavaScript*, dll.) Tersedia untuk penguji penetrasi dan dapat memberikan informasi penting tentang operasi internal aplikasi website.

Bagian ini membahas Celah Keamanan yang harus dideteksi oleh Scanner keamanan website. Celah Keamanan ini akan membentuk dasar dari persyaratan yang dinyatakan secara resmi untuk fitur wajib untuk *Scanner* keamanan website. Proyek Keamanan Aplikasi *website Terbuka (OWASP)* menerbitkan daftar Celah Keamanan aplikasi website yang paling kritis. Upaya ini dan lainnya termasuk dalam *Common Weakness Enumeration (CWE)*.

Sebahagian besar celah keamanan aplikasi website disebabkan oleh kelemahan validasi input. Penyebab kelemahan lain adalah

DOI: https://doi.org/10.29103/techsi.v11i3.1996

penggunaan mekanisme otentikasi yang lemah, kelemahan logika, pengungkapan konten dan informasi rahasia yang tidak disengaja, dan kelemahan pengkodean tingkat rendah (seperti *buffer overflow*). Kombinasi titik lemah ini yang sering menyebabkan kelemahan pada *website.*

Beberapa serangan dan Celah Keamanan berbasis web umum diberikan di bawah ini:

1. Celah Keamanan *Cross Site Scripting (XSS);* Celah Keamanan ini terjadi ketika penyerang mengirim data berbahaya ke aplikasi website. Contoh data tersebut termasuk script sisi klien dan *hyperlink* ke situs tempat penyerang berada. Jika aplikasi mengumpulkan data tanpa verifikasi yang tepat dan secara dinamis menampilkannya di halaman web yang dihasilkan, itu akan menampilkan data berbahaya di browser add on pengguna yang sah. Akibatnya, penyerang dapat memproses atau mencuri kredensial pengguna yang sah, dengan berkedok sebagai pengguna yang sah, atau mengeksekusi script berbahaya pada mesin pengguna.
2. Celah Keamanan injeksi; tipe serangan ini termasuk injeksi data, injeksi perintah, injeksi sumber daya, dan serangan injeksi SQL. SQL Injection bisa terjadi karena tidak difilter dengan benar input pengguna aplikasi website dan menempatkannya langsung ke pernyataan SQL. Hal ini memungkinkan data dalam database ditampilkan atau dimodifikasi. Objek injeksi lain yang mungkin adalah *script* yang dapat dieksekusi paksa untuk melakukan hal-hal yang berbahaya.
3. *Cookie Poisoning* adalah teknik yang digunakan terutama untuk mengakses peniruan dan pelanggaran privasi melalui manipulasi cookie sesi yang melindungi identitas klien. Dengan melakukan ini, penyerang dapat meniru klien yang valid dan dengan demikian mendapatkan informasi dan melakukan tindakan atas nama korban.
4. Otorisasi, otentikasi, dan Celah Keamanan pada access control dapat memungkinkan hacker mengendalikan aplikasi atau server back-end. Jenis serangan ini meliputi; manajemen kata sandi yang lemah, penggunaan metode enkripsi yang lemah, penggunaan elevasi hak istimewa, penggunaan makro yang tidak aman, kesalahan otentikasi, dan kesalahan kriptografi.

DOI: https://doi.org/10.29103/techsi.v11i3.1996

5. Kesalahan penanganan informasi dan pelaporan dapat mengungkapkan informasi yang memungkinkan hacker untuk memprediksi informasi sensitif. Ini termasuk *catch NullPointerException, blok catch* kosong, dan deklarasi "*throAplikasi website scanner*".

Beberapa Celah keamanan lainnya:

1. Penolakan Layanan *(DoS)*
2. Manipulasi Jalur
3. Otentikasi Rusak
4. Access control Rusak
5. Paparan Data Sensitif
6. Kesalahan Konfigurasi Keamanan
7. Menggunakan Komponen dengan Celah Keamanan yang sudah diketahui orang

4. Analisa Program Pendeteksi Celah Keamanan Website

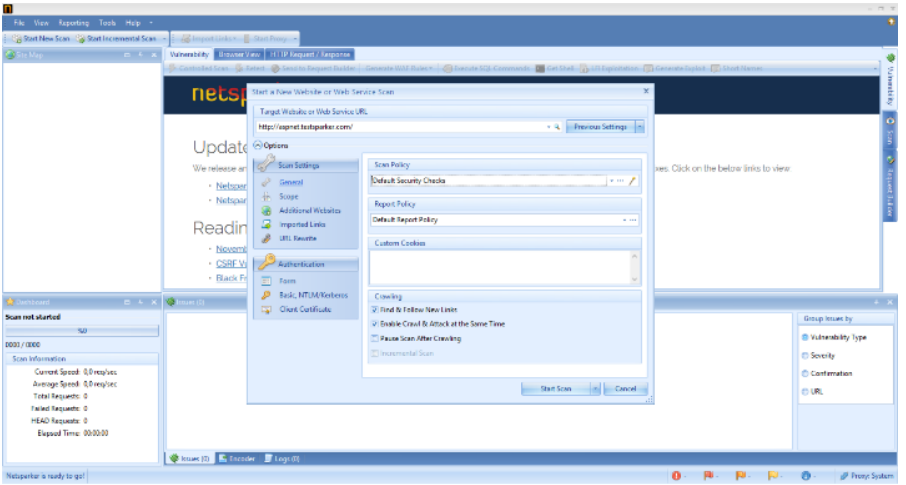
Pada bagian ini, enam program uji komersial dan open source diperiksa. Ini adalah *Netsparker, Acunetix, Vega, OWASP ZAP, Wapiti* dan *IronWASP.* Untuk menguji setiap program, tes dilakukan pada alamat *http://aspnet.testsparker.com/* yang disiapkan oleh tim Netsparker.

A. Netsparker

Netsparker adalah program pengujian keamanan web. Ini menganalisis dan melaporkan celah keamanan di tingkat aplikasi web mana pun. Dengan menggunakan *Netsparker,* selain mendeteksi celah keamanan ini, ia juga dapat mengekstrak data dari sistem penyerang, atau menyediakan akses penuh ke sistem. Namun, banyak fault positif yang harus dihindari. *Netsparker* memberikan dukungan penuh untuk aplikasi berbasis *AJAX* dan *JavaScript. Netsparker* juga menyediakan dukungan penuh untuk *HTML5. Netsparker* memiliki dua versi, yaitu versi desktop dan cloud. Versi cloud memungkinkan kita untuk mendeteksi ribuan website atau aplikasi berbasis web secara bersamaan.

Berikut adalah skenario deteksi celah keamanan web dengan *Netsparker*, tes dilakukan dengan aplikasi desktop Netsparker versi

4.8, *URL* yang diuji dalam skenario adalah *http://aspnet.testsparker.com /*



Gambar 2. Input URL dan konfigurasi pengujian dengan program Netsparker.

Mendeteksi dengan *Netsparker* bersifat komprehensif dan telah mengidentifikasi 47 celah keamanan di situs belajar keamanan *website http://aspnet.testsparker.com.* Pendeteksian memakan waktu sekitar 2 jam. Laporan celah keamanan sangat rinci dan sangat memuaskan. Kelemahan dari pendeteksian komprehensif ini adalah waktu pendeteksiannya cukup lama. *Netsparker* adalah salah satu program pengujian keamanan web kualitas terbaik yang dapat digunakan dengan antarmuka yang ramah pengguna, dan pendeteksian komprehensif.

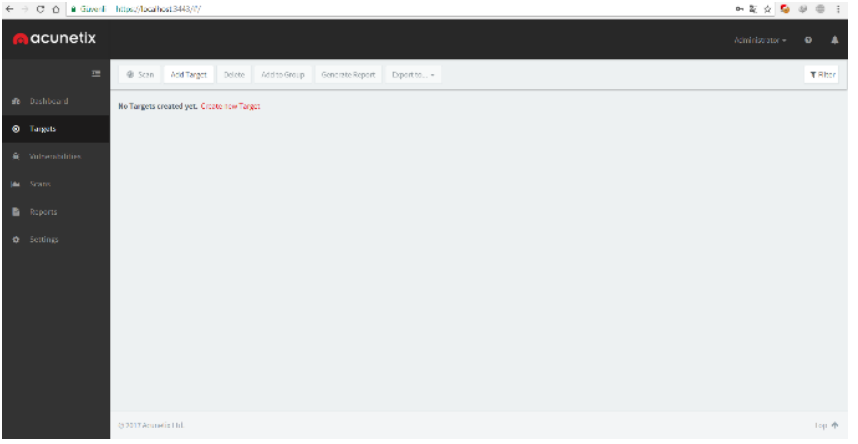
B. Acunetix

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.*

DOI: https://doi.org/10.29103/techsi.v11i3.1996

Acunetix juga memiliki deteksi celah keamanan *WordPress* yang paling canggih dan berbagai laporan manajemen dan hukum, termasuk *ISO* 27001 dan *PCI*.

Berikut skenario Tes yang dilakukan, tes ini dilakukan dengan Acunetix v11 pada Windows Aplikasi website scanner 10 (x64).



Gambar 3. Tampilan pendeteksi celah keamanan *web Acunetix*.

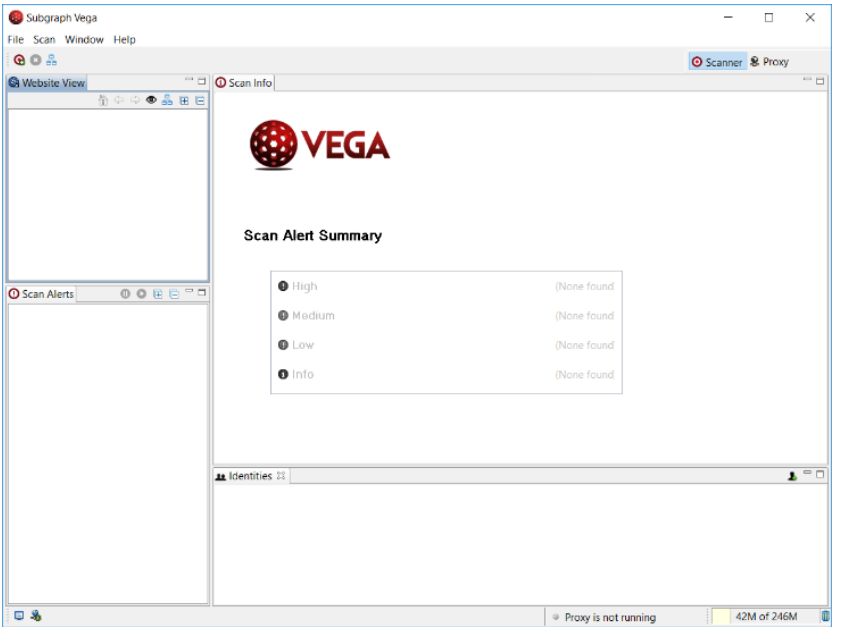
Acunetix memiliki antarmuka yang sangat sederhana dan ramah pengguna. Kelengkapan pada bagian pelaporan memberikan nilai tambah yang besar dibandingkan pesaingnya. Dalam pengaturan konfigurasi pra-pendeteksian, ia menawarkan opsi yang cukup banyak. Dalam penelitian ini, sebagai hasil pendeteksian, 163 Celah Keamanan telah terdeteksi di *website*

C. Vega

Vega adalah program pengujian keamanan web gratis dan *open source* yang digunakan untuk menemukan celah keamanan dalam aplikasi *website. GUI Vega* ditulis dengan *Java. Vega* memiliki dua fungsi, yaitu sebagai *Scanner* dan *Proxy*. Modul serangan *Vega* dibuat dengan *JavaScript* yang *open source*, sehingga modul-modul ini dapat diperluas dengan *JavaScript API* atau ditulis ulang oleh pengguna.

Skenario Tes yang dilakukan dengan Vega v1 pada Windows Aplikasi website scanner 10 (x64) terlihat pada gambar 4 berikut :

DOI: https://doi.org/10.29103/techsi.v11i3.1996



Gambar 4. GUI Pendeteksi Celah Keamanan Web Vega

Vega memberikan keuntungan kepada para pesaingnya dengan open source dan rasio kinerja. Ini dianggap sebagai alternatif yang sangat baik untuk penggunaan individu dalam sistem dengan tingkat keparahan yang lebih rendah. Dalam skenario ini, pengujian aplikasi website yang diberikan dalam skenario selesai dalam 45 menit dan menemukan total 151 Celah keamanan. Selain itu, modul dapat diatur atau ditulis ulang memberikan keuntungan besar bagi pengguna.

D. OWASP Zed Attack Proxy (ZAP)

OWASP adalah organisasi nirlaba di seluruh dunia yang berfokus pada peningkatan keamanan perangkat lunak. *OWASP* berada dalam posisi unik untuk memberikan informasi praktis dan tidak memihak tentang keamanan perangkat lunak, kepada individu, perusahaan, dan lembaga pemerintah. *OWASP* juga merupakan komunitas online yang menyediakan artikel, metode, dokumentasi, dan program secara bebas di bidang keamanan

DOI: https://doi.org/10.29103/techsi.v11i3.1996

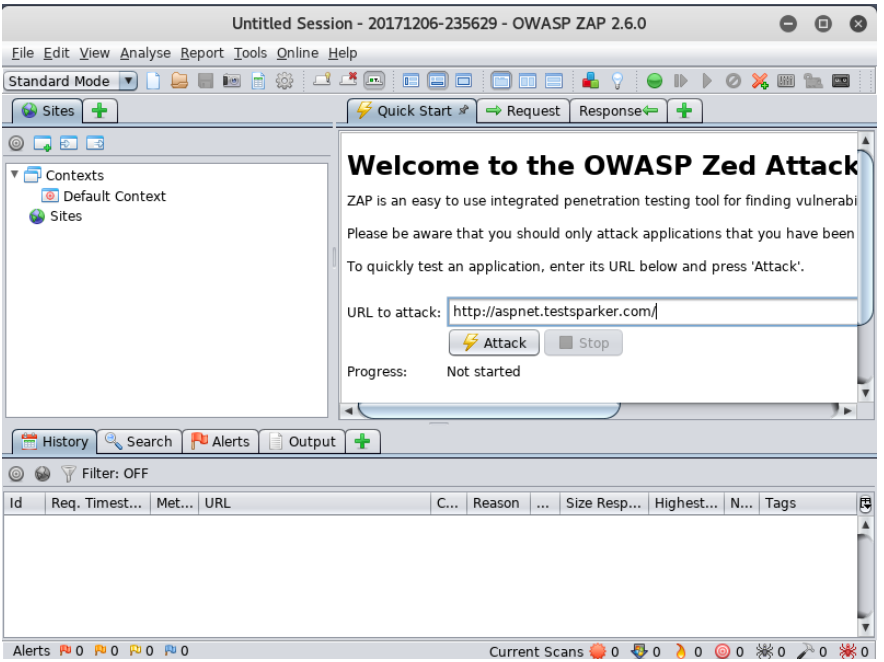
aplikasi *website. ZAP* adalah program pengujian penetrasi terintegrasi pada *OWSAP* yang mudah digunakan, gratis dan *open source* untuk mendeteksi celah keamanan dalam aplikasi *website. ZAP* juga dikenal sebagai proyek unggulan *OWASP. ZAP* juga menyediakan seperangkat program yang memungkinkan Anda untuk mendeteksi Celah Keamanan secara manual selain pendeteksi otomatis. *ZAP* juga dapat menghasilkan laporan dalam format *HTML* dan *XML*.

Skenario Tes dalam kasus ini dilakukan dengan aplikasi *OWASP ZAP 2.6.0* di Kali Linux 2019.

ZAP menguntungkan karena struktur berbasis open source. Tapi itu tidak mendeteksi semua Celah keamanan yang diketahui dalam sistem. Program ini dianggap lebih cocok untuk orang yang baru melakukan pengujian penetrasi. Dalam skenario ini, pendeteksian selesai dalam waktu sekitar 1,5 jam dan mendeteksi total 7 Celah Keamanan. Salah satu keuntungan terbesar ZAP adalah adanya sejumlah program yang memungkinkan penguji menemukan celah keamanan secara manual selain pendeteksi otomatis.

Berikut skenario Tes dengan aplikasi *OWASP ZAP 2.6.0* di Kali Linux 2019.

DOI: https://doi.org/10.29103/techsi.v11i3.1996



Gambar 5. OWASP ZAP GUI

E. Wapiti

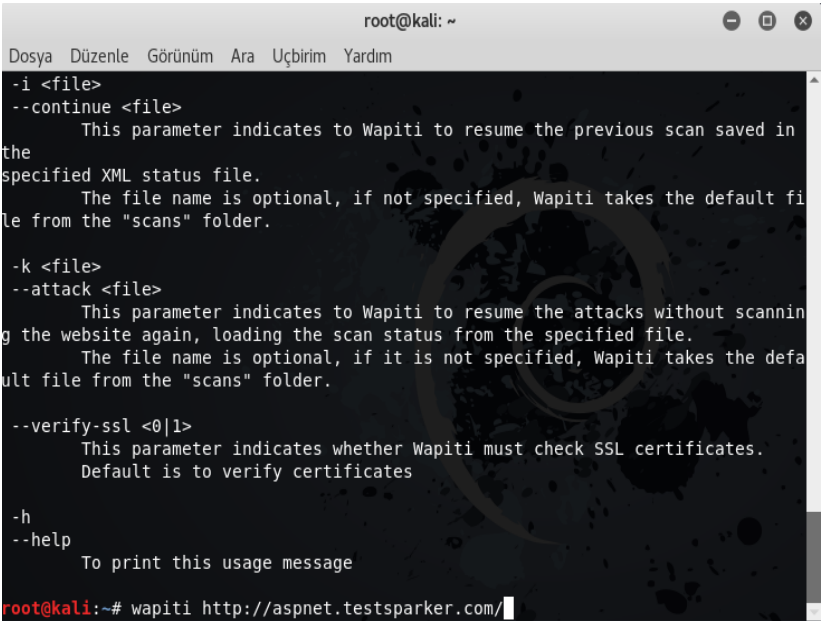
Wapiti adalah program pengujian keamanan web gratis yang digunakan untuk mendeteksi Celah Keamanan dalam aplikasi website. Ia melakukan tes *Black Box*, yaitu tidak menganalisis kode sumber aplikasi, tetapi mendeteksi halaman web aplikasi website yang akan diuji dan melihat *script* dan *formulir* yang dapat menyuntikkan data. Setelah mendapat daftar *URL, formulir*, dan inputnya, *Wapiti* bertindak seperti seorang fuzzer, menyuntikkan muatan untuk melihat apakah *script* yang rentan disusupi.

Wapiti dapat mendeteksi celah keamanan berikut:

1. Pengungkapan file
2. Injeksi Basis Data (Suntikan *PHP / JSP / ASP* dan Suntikan *XPath*)
3. Injeksi *XSS*
4. Deteksi Eksekusi Perintah *(eval (), system (), passtru () ...)*
5. Injeksi *CRLF*
6. Injeksi *XXE*
7. *SSRF*

DOI: https://doi.org/10.29103/techsi.v11i3.1996

8. *Shellshock* (alias *Bash bug*)



Gambar 6. GUI Wapiti

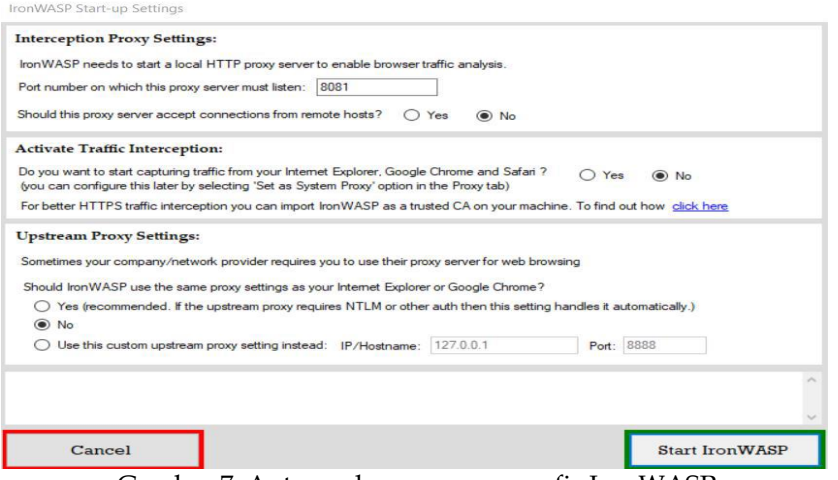
Berikut skenario tes yang dilakukan pada Kali Linux 2019, karena operasi aplikasi berbasis terminal, menjadi sulit untuk digunakan, terutama dalam hal kemudahan penggunaan, yang merupakan pesaingnya dengan *GUI*. Kemampuan untuk mewarnai posting di terminal meningkatkan keterbacaan. *Wapiti* menyelesaikan pendeteksian dalam waktu yang sangat lama sekitar 13 jam dan menemukan total 129 Celah Keamanan. Untuk memastikan bahwa pendeteksian tidak memakan waktu lama, Wapiti harus dijalankan dari modul menggunakan parameter khusus untuk pendeteksian. Ini adalah fitur yang bagus untuk menghasilkan laporan dalam format *HTML* dan *XML*. Hasil deteksi yang dibuat dengan Wapiti dapat ditransfer ke Metasploit dan laporan yang dihasilkan dalam format *XML* dapat ditambahkan ke basis data *Metasploit*. Fakta bahwa aplikasi hanya melakukan pengujian *Black Box*, yang tidak memungkinkan kita untuk menganalisis kode sumber aplikasi menyebabkannya kalah dibandingkan para pesaingnya.

DOI: https://doi.org/10.29103/techsi.v11i3.1996

F. IronWASP

Platform Pengujian keamanan *IronWASP* adalah program pengujian keamanan *web open source* dan gratis yang digunakan untuk menemukan celah keamanan dalam aplikasi *website. IronWASP* dapat mengidentifikasi lebih dari 25 Celah Keamanan web. Ini adalah program berbasis *GUI* yang ditulis dalam *Python* dan *Ruby*. *IronWASP* dapat mendeteksi positif palsu dan negatif palsu. *IronWASP* menghasilkan laporan dalam format *HTML* dan *RTF*. *IronWASP* dapat diperluas melalui *plug-in* atau modul yang ditulis dengan *Python, Ruby, C* # atau *VB.NET*.

Berikut skenario tes yang dilakukan dengan *IronWASP 2015* pada *Windows 10*.



Gambar 7. Antarmuka pengguna grafis IronWASP

IronWASP adalah program *open source* dan menguntungkan dengan tingkat kinerjanya. Program *IronWASP* berbasis *GUI* dapat menghasilkan laporan dalam format *HTML* dan *RTF*. *IronWASP* mudah digunakan, ini membuatnya menjadi program yang nyaman untuk pentester pemula. Ini adalah alternatif yang baik untuk para profesional yang memiliki banyak opsi pengaturan. Dalam skenario pengujian ini, *IronWASP* menyelesaikan pendeteksian dalam waktu yang sangat lama 7 jam dan mendeteksi total 213 celah keamanan. Kerugian terbesar adalah waktu pendeteksian yang

DOI: https://doi.org/10.29103/techsi.v11i3.1996

lama. Selain itu, karena struktur modular program ini, modul dapat diatur atau ditulis ulang untuk memberikan keuntungan besar bagi pengguna.

5. Evaluasi Dan Pembahasan

Sejalan dengan meluasnya penggunaan aplikasi *website,* kebutuhan akan pengujian semakin meningkat. Selain spesifikasi fungsinya, perlu diketahui apakah program tersebut menyediakan fitur rencana uji dan kasus uji untuk memeriksa pendeteksi celah keamanan. Rencana pengujian merinci bagaimana suatu aplikasi website tersebut diuji, bagaimana hasil tes ditafsirkan, dan bagaimana laporan yang dihasilkan. Saat ini, program pendeteksi celah keamanan menghasilkan laporan dalam berbagai format. Format pelaporan yang umum akan memudahkan untuk mengotomatiskan perbandingan dengan berbagai program lain. Kami mengukur kesesuaian program menurut berbagai kondisi pengujian. Penting untuk mengetahui bagaimana penyerang menggunakan celah keamanan saat memilih kasus uji ini.

Dalam aplikasi website normal, pengguna mengirim permintaan ke aplikasi website dan menerima respons. Tetapi seorang penyerang mengirimkan permintaan yang tidak terduga ke aplikasi berharap untuk mengeksploitasi celah keamanan yang ada. Tujuan penyerang adalah untuk melanggar kebijakan keamanan aplikasi *website* tersebut. Seorang penyerang mendapatkan adanya celah keamanan dengan memeriksa respons aplikasi website atau memperhatikan perubahan perilaku aplikasi tersebut. *Scanner-scanner* keamanan *website* berfungsi dengan mensimulasikan tindakan penyerang.

Sampel aplikasi *website* dengan celah keamanan diperlukan untuk menguji kinerja *scanner* keamanan *website* yang akan digunakan. Paling tidak harus ada satu program pengujian untuk setiap kelas celah keamanan. Kasus uji kecil dengan satu celah keamanan dapat digunakan untuk secara tepat menguji kemampuan program untuk mendeteksi celah keamanan tertentu. Penting juga untuk menguji kemampuan program untuk mendeteksi celah keamanan dalam aplikasi *website* yang dibuat menggunakan teknologi web yang berbeda.

DOI: https://doi.org/10.29103/techsi.v11i3.1996

Paket tes dasar dapat mencakup aplikasi yang memiliki celah keamanan. Misalnya, jika suatu aplikasi tidak memverifikasi entri apa pun, ada banyak cara untuk mengeksploitasi celah keamanan, dan sebagian besar program dapat menemukannya kelemahan ini. Namun, untuk menguji pendeteksi secara menyeluruh, kami membutuhkan program yang mampu mendeteksi celah keamanan dengan sensitif.

Contoh lain adalah dengan serangan injeksi *SQL*. Penyerang biasanya mengirimkan permintaan yang menyebabkan aplikasi membuat kueri *SQL* yang dapat menyebabkan respon tak terduga dari sebuah website. Kemudian, penyerang memeriksa pesan kesalahan yang ditampilkan pada web tersebut. Pendekatan mitigasi yang khas adalah untuk mencegah aplikasi menampilkan pesan kesalahan basis data apa pun. Perbandingan data yang diperoleh dari skenario yang diterapkan dan program uji dirangkum dalam Tabel 1.

Tabel 1. Perbandingan data yang diperoleh dari skenario yang diterapkan dan program uji

Program	**Waktu Scanning**	**Celah keamanan**	**Laporan**	**Tes Manual**	**GUI**	**Harga**	**Pengguna**
Netsparker	2 jam	47	Ada	Yes	Yes	5000 $	Institusi
Acunetix	2.5 jam	163	Ada	Ada	Ada	4500 $	Institusi
Vega	45 menit	151	Ada	Tidak	Ada	-	Individual
OWASP ZAP	1.5 jam	7	Ada	Ada	Ada	-	Individual
Wapiti	13 jam	129	Ada	Ada	Tidak	-	Individual
IronWASP	7 jam	213	Ada	Ada	Ada	-	Individual

6. Kesimpulan

Dalam artikel ini, berbagai macam program pendeteksi celah keamanan aplikasi website telah diperiksa dan dievaluasi secara

DOI: https://doi.org/10.29103/techsi.v11i3.1996

terperinci untuk mengetahui program scanner mana yang paling cocok digunakan untuk mendeteksi kelemahan keamanan sebuah *website.* Program uji tersebut memberikan informasi tentang cara melakukan berbagai skenario serangan terhadap *website* sampel. Artikel ini juga menunjukkan kelebihan dan kekurangan kinerja dari program yang diuji.

Daftar Pustaka

Acunetix, Web Site Security, Diakses pada 20 Agustus 2019: dari https://www.acunetix.com/.

Common Weakness Enumeration: CWE, MITRE, Diakses pada 20 Agustus 2019: dari http://cve.mitre.org/cwe/.

Fortune Turkey, Diakses tanggal 20 Agustus 2019 dari: http://www.fortuneturkey.com/fotohaber/dunyanin-en-degerli-markalari-2017-42674.

Jeremiah Grossman, 2005, The Five Myths of Web Application Security, WhiteHat Security, Inc,

Jody Melbourne ve David Jorm, 2003. Penetration Test for Web Applications, SecurityFocus,

G. McGraw, 2006, Software Security: Building Security, Addison-Wesley Software Security Series,

IronWASP, Application Security Scanner, Diakses pada 20 Agustus 2019: dari https://www.utest.com/tools/ironwasp.

Netsparker, Web Application Security Scanner, Diakses pada 20 Agustus 2019: dari https://www.netsparker.com/.

OWASP ZAP, Zed Attack Proxy, Diakses pada 20 Agustus 2019: dari https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project.

DOI: https://doi.org/10.29103/techsi.v11i3.1996

OWASP, "OWASP Top 10 Most Critical Web Application Security Risks", [Online]. Diakses pada 20 August 2019: http://www.owasp.org/index.php/OWASP_Top_Ten_Project.

Prescatore, John, Gartner, Computerworl, 25 Şubat 2005, [Online]. Diakses pada 20 August 2019 dari http://www.computerworld.com/printthis/2005/0,4814,99981,00.html.

Vega, Vulnerability Scanner Diakses pada 20 Agustus 2019: dari, https://subgraph.com/vega/index.fr.html.

Wapiti, Web Vulnerability Scanner, Diakses pada 20 Agustus 2019: dari http://wapiti.sourceforge.net/.

Web Application Security Consortium Glossary, [Online]. Diakases pada 20 August 2019: dari: http://www.webappsec.org/projects/glossary/.

Web Application Security Consortium, "Threat Classification", [Online]. Diakses pada 20 Agustus 2019: dari http://www.webappsec.org/projects/threat/.

DOI: https://doi.org/10.29103/techsi.v11i3.1996