**STANDAR PENGUJIAN KEAMANAN APLIKASI BERBASIS WEB**

**{{nama organisasi}}**

**{{logo organisasi}}**

|  |  |
| --- | --- |
| No Dokumen | : 0.3 (Draft) |
| Revisi | : 00 |
| Tanggal Terbit | : |

|  |  |  |
| --- | --- | --- |
| Disusun  oleh : | Diketetahui  oleh: | Disetujui / Disahkan  oleh : |
|  |  |  |
| **NN**  NN | **NN**  NN | **NN**  NN |

**Riwayat Dokumen**

|  |  |  |
| --- | --- | --- |
| **Versi** | **Tanggal** | **Perubahan** |
| 0.1 (Draft) | 28/02/2023 | Rilis draft awal dengan hanya menyalin sepenuhnya konten dari **PEDOMAN TATA KELOLA KEAMANAN APLIKASI BERBASIS WEB - BSSN** |
| 0.2 (Draft) | 01/03/2023 | Penambahan standar Security Design Review (SDR) |
| 0.3 (Draft) | 03/03/2023 | Penambahan aktivitas-aktivitas pengujian keamanan pada tahapan Secure SDLC |

1. **TUJUAN**
   1. Instansi dapat mengidentifikasi kelemahan keamanan suatu aplikasi, sistem komputer, atau suatu jaringan.
   2. Jika celah kelemahan ditemukan dan dapat dibuktikan dengan beberapa Analisis resikonya, maka instansi dapat segera melakukan perbaikan sistem.
2. **RUANG LINGKUP**
   1. Vulnerability Assessment
   2. Penetration Testing
3. **ISTILAH DAN DEFINISI**

|  |  |  |  |
| --- | --- | --- | --- |
| 1. | Penetration testing | : | Suatu kegiatan pengujian keamanan sebuah sistem, aplikasi, atau jaringan untuk mengetahui keamanan yang terdapat pada sistem atau aplikasi mempunyai celah keamanan sehingga dapat segera diperbaiki dengan melakukan patch.. Hal ini dilakukan agar keamanan yang terdapat pada suatu sistem atau aplikasi yang diuji menjadi semakin kuat. Selain melakukan pengujian, juga mendokumentasi tingkat keamanan dari sistem atau aplikasi yang diuji untuk selanjutnya  dibuatkan laporan kepada lembaga/pimpinan |
| 2. | Instansi | : | Kementerian/Lembaga, Instansi Pusat dan Daerah |
| 3. | *Intranet* | : | Jaringan pribadi yang menggunakan protokol komunikasi internet  protocol, dapat terhubung ke internet (tidak selalu), dan hanya dapat digunakan dalam lingkungan terbatas. |
| 4. | *Password* | : | Kata sandi yang digunakan bersamaan dengan username (sign  on/sign in/log-on/log-in) oleh pemilik yang sah sebelum melakukan koneksi/akses ke sistem komputer. |
| 5. | *Patch* | : | Rutin program atau sekumpulan kecil instruksi yang biasanya dibuat sebagai solusi sementara untuk mengatasi atau memperbaiki permasalahan (bugs) pada program komputer dan sering dibuat dalam bentuk ’object code’ yang disisipkan ke dalam program yang akan  dieksekusi. |
| *6.* | *Directory Traversal* |  | Jenis eksploitasi HTTP yang digunakan oleh penyerang untuk mendapatkan akses tidak sah ke file dan direktori terbatas. Selain itu serangan lintasan direktori menggunakan perangkat lunak server web untuk mengeksploitasi mekanisme keamanan yang tidak memadai dan mengakses file dan direktori yang disimpan di luar folder root Aplikasi  Berbasis Web. |

|  |  |  |
| --- | --- | --- |
| *7.* | *Vulnerability Assessment* | Adalah proses identiﬁkasi dan kuantiﬁkasi kerentanan keamananan pada suatu lingkungan keamanan sistem infomasi. Dapat diartikan juga sebagai suatu evaluasi mendalam terhadap keamanan sistem  infomasi yang aktif digunakan. |
| *8.* | Logic Bomb | Merupakan salah satu program jahat yang ditempelkan pada computer agar mameriksa suatu kumpulan kondisi di system. Ketikaa kondisi- kondisi yang dimaksud ditemui, logic mengeksekusi suatu fungsi yang menghasilkan aksi-aksi tak diotorisasi. Logic bomb menempel pada suatu program resmi yang diset meledak ketika kondisi-kondisi tertentu dipenuhi. Contoh kondisi-kondisi untuk memicu logic bomb adalah ada atau tidaknya file tertentu, hari tertentu dari minggu atau tanggal, atau pemakai menjalankan aplikasi tertentu. Begitu terpicu, bomb mengubah atau menghapus data atau seluruh file, menyebabkan  mesin berhenti, atau menyebabkan kerusakan lain. |

1. **REFERENSI**

1. OWASP Testing Guides Release 4.0

1. **STANDAR**
2. **Penilaian Kerentanan Sistem (*Vulnerability Assessment*)**
   1. Kegiatan ini dilakukan untuk menghadapi, menghindari, membagi, dan mengurangi kerentanan sistem sampai batas toleransi risiko yang dapat diterima.
   2. Instansi dalam periode tertentu harus melakukan tahapan penilaian kerentanan sistem berupa:\

. Penyusunan Katalog asset sistem informasi dan sumber daya lainnya mencakup seluruh sistem yang digunakan pada proses bisnis instansi

* + 1. Menentukan nilai dan tingkat kepentingan terhadap aset. Hasil dalam bentuk daftar aset berdasarkan urutan yang dinilai paling tinggi.
    2. Identiﬁkasi kerentanan keamanan dan potensi ancaman (dengan dampak terburuk) yang dapat terjadi pada setiap aset. Hasil dalam bentuk daftar ancaman berdasarkan tingkat dampak yang dinilai paling tinggi
    3. Mitigasi atau eliminasi ancaman dilakukan (atau direncanakan) terhadap ancaman yang memilik tingkat dampak dan aset bernilai paling tinggi.

1. **Pengujian Aplikasi Berbasis Web (*Penetration Test*)**
   1. Penentuan Tujuan (*Identifying Objectives*)

Dalam tahapan ini, instansi yang meminta pengujian harus menyampaikan:

* tujuan dan ruang lingkup pengujian;
* skenario pengujian;
* batasan pengujian;
* teknik dan tools yang digunakan dalam pengujian
  1. Pengumpulan/Pemindaian Informasi (*Gathering & Scanning Information*)

Mengumpulkan informasi sebanyak mungkin dari suatu Aplikasi Berbasis Web melalui mesin pencari atau melalui pendekatan secara aktif dengan perangkat yang dirancang untuk memetakan / mengumpulkan informasi dari sebuah Aplikasi Berbasis Web. Sebuah aplikasi sebagai alat teknis untuk mengumpulkan berbagai data lanjutan pada target yang telah kita tentukan. Pada tahapan ini data yang dicari lebih umum, yaitu mengenai sistem yang dimiliki.

* 1. *Application Threat Modeling*

Melakukan aktivitas Threat Modeling sesuai standar yang hasilnya akan memberikan gambaran umum desain arsitektur aplikasi beserta kemungkinan *threat* yang akan terjadi beserta mitigasinya.

* 1. *Infrastructure Penetration Testing*

Melakukan Infrastructure Penetration Testing menggunakan tool InsightVM, nikto, dan nuclei untuk kemudian dilakukan eksploitasi.

* 1. *Web Penetration Testing*

Melakukan Web Penetration Testing menggunakan tool Burp Suite Professional dan OWASP Zap Proxy mulai dari melakukan *web* *vulnerability scanning* hingga melakukan eksploitasi terhadap kerentanan yang ditemukan

* 1. *Reporting*

Mendokumentasikan seluruh hasil temuan kerentanan, menghitung skor tingkat keparahan kerentanan menggunakan CVSS, lalu meng-*export* seluruh hasil daftar kerentanan ke aplikasi OWASP Defect-dojo untuk dilakukan *vulnerability tracking & triaging* (Vulnerability Management).

* 1. Objek pengujian penetrasi meliputi:
* Layanan yang menggunakan koneksi internet (Aplikasi Berbasis Web, VPN endpoint, infrastruktur e-mail, extranet, dan lain-lain).
* Sistem internal atau servis yang terdapat di dalam jaringan (Active Directory, Exchange, dan lain-lain).
* Aplikasi mobile (iOs & Android ), web, dan desktop.
* Jaringan internal.
  1. JIka penetration test menggunakan pihak ketiga, sebelum melakukan penetration testing, ada disepakati kontrak antara pentester dengan instansi yang aplikasi atau sistemnya akan diuji, sehingga dalam konteks hukum bahwa kegiatan pentest merupakan kegiatan yang legal.
  2. Jenis – jenis pentest yang dapat dilakukan diantaranya:
* *Black Box Testing*

Pengujian dilakukan berdasarkan detail aplikasi, seperti tampilan aplikasi, fungsi-fungsi yang terdapat pada aplikasi, serta penyesuaian alur fungsi pada aplikasi dengan bisnis yang diinginkan oleh pelanggan. Pengujian ini dilakukan tanpa melihat dan menguji source code program yang ada pada aplikasi.

* *White Box Testing*

Pengujian dilakukan berdasarkan detail prosedur serta alur logika dari sebuah kode program. Pada metoda ini, tester akan melihat keseluruhan source code sebuah program untuk menemukan bugs dari kode program tersebut.

* *Grey Box Testing*

Pengujian merupakan kombinasi dari *Black Box* dan *White Box*, dimana pentester melakukan pengujian aplikasi berdasarkan spesifikasi namun menggunakan cara kerja dari dalam aplikasi tersebut alias source code program.

* 1. Dari jenis – jenis pentest diatas, item – item yang diuji secara umum adalah:

1. Pemindaian kerentanan/Pengujian server yang mengidentifikasi / melaporkan:
   * Perangkat lunak tidak resmi
   * Port server yang dibuka secara tidak tepat, protokol dan layanan yang diaktifkan
   * Layanan yang salah konfigurasi atau diaktifkan secara tidak tepat, misalnya, ftp dan telnet
   * Kredensial yang tersimpan tidak sesuai terkait dengan pekerjaan batch, skrip, atau file teks biasa
   * Akun lokal yang tidak sesuai dengan aturan kata sandi yang tidak kadaluwarsa
   * Metoda dan level enkripsi yang digunakan tidak memadai
   * Kemampuan untuk mendapatkan akses tidak sah ke kunci enkripsi
   * Kata sandi server yang lemah (seperti dapat ditentukan melalui pemecahan kata sandi)
   * Patch yang hilang
2. Pemindaian kerentanan/Pengujian Aplikasi Aplikasi Berbasis Webyang mengidentifikasi / melaporkan:
   * Panggilan atau tanggapan API tidak aman
   * Antarmuka aplikasi lintas tidak aman
   * Pengodean kode kustom yang tidak aman dalam produk COTS atau kode yang menggunakan API COTS
   * Pengodean tidak aman dan fungsionalitas fungsi aplikasi sensitif atau antarmuka akses istimewa seperti layar administrator aplikasi
   * Kemampuan untuk mengeksekusi perintah atau menyuntikkan kode (mis. Perintah OS, injeksi SQL, Cross-site Scripting, injeksi LDAP)
   * Kontrol manajemen sesi yang tidak memadai
   * Kemampuan untuk melakukan serangan jalur URL path
   * Kemampuan untuk menyebabkan kondisi overflow (mis. Parameter overflow dan buffer overflow)
   * Kemampuan untuk melakukan serangan encoding karakter
   * Kemampuan untuk berkompromi dengan aplikasi dengan memberikan nilai input yang tidak sesuai (mis. Pengujian fuzz)
   * Kelemahan keamanan dalam Layanan Aplikasi Berbasis Web(berbasis REST dan SOAP) yang digunakan oleh aplikasi
3. Analisis kode statis untuk mengidentifikasi / melaporkan:
   * Keberadaan Bom Logika (*Logic Bomb*) atau *Backdoors*
   * Fitur debug yang diaktifkan
   * Kredensial disimpan secara tidak benar dalam kode
4. Pemindaian/Pengujian *Middleware* untuk mengidentifikasi / melaporkan:
   * Pengaturan konfigurasi yang tidak memadai
   * Direktori tidak resmi yang dapat diakses atau ditampilkan, mis. Pencacahan direktori
   * File aplikasi sisi server yang tidak sah yang dapat diakses untuk diunduh atau diperiksa oleh klien (mis. Melihat konten file php, jsp, atau asp)
   * Informasi produk yang tidak perlu ditampilkan (mis. Modul yang dipasang)
   * Akun atau fitur yang seharusnya tidak perlu diaktifkan
   * Patch yang hilang
   * Kata sandi yang masih disetting default
5. Pemindaian/Pengujian Database untuk mengidentifikasi / melaporkan:
   * Pengaturan konfigurasi yang tidak memadai
   * Akun atau fitur yang tidak perlu diaktifkan
   * Hak istimewa yang berlebihan diberikan untuk objek basis data atau ke file OS basis data
   * Akun lokal yang tidak memadai dengan kata sandi yang tidak kedaluwarsa
   * Kredensial disimpan secara tidak benar dalam pekerjaan batch atau skrip
   * Pemisahan tugas yang tidak memadai
   * Adanya utilitas khusus atau fitur debug yang diaktifkan di lingkungan produksi
   * Ketidakcukupan metoda enkripsi dan level yang digunakan
   * Kemampuan untuk mendapatkan akses tidak sah ke kunci enkripsi
   * Kata sandi basis data yang lemah (mis. melalui pemecahan kata sandi)
   * Replikasi database melalui saluran yang tidak aman
   * Kemampuan untuk membaca, memodifikasi, menyalin, atau menghapus data konfigurasi, log, dan informasi kontrol akses
   * Kecukupan kontrol untuk semua titik masuk dan keluar dari suatu aplikasi
   * *Patch* yang hilang
   * Kata sandi yang masih disetting default
6. Pemindaian/Pengujian Aplikasi Aplikasi Berbasis Web untuk mengidentifikasi / melaporkan:
   * Konfigurasi yang tidak memadai
   * Akun atau fitur yang seharusnya tidak perlu diaktifkan
   * Hak Akses yang berlebihan
   * Kemampuan untuk memotong jalur akses aplikasi yang normal
   * Kendali manajemen sesi yang tidak memadai
   * Pemisahan infrastruktur yang tidak memadai
   * Kata sandi aplikasi yang lemah (mis. Melalui pemecahan kata sandi)
   * Pemisahan aset yang tidak memadai berdasarkan tujuan dan lingkungan
   * Akses istimewa tidak melalui jalur administratif
   * Kemampuan untuk meningkatkan hak istimewa
   * Kemampuan untuk menyusup data
   * Kemampuan untuk menyimpan konten berbahaya
   * Kemampuan untuk mendapatkan akses tidak sah ke data dan ke file produk yang diinstal
   * Kemampuan untuk mendapatkan akses tidak sah ke kunci enkripsi
   * Kemampuan untuk mendapatkan akses tanpa izin ke antarmuka dan alat administratif
   * Kemampuan untuk membaca, memodifikasi, menyalin, atau menghapus data konfigurasi, log, dan informasi kontrol akses
   * Utilitas yang diistimewakan atau mengaktifkan fitur debugging di lingkungan produksi
   * Metoda dan level enkripsi yang digunakan tidak memadai
   * Kemampuan untuk mendapatkan akses tidak sah ke kunci enkripsi
   * Kerentanan terhadap serangan umum seperti DDoS, dan replay sesi
   * Akses yang tidak terotorisasi atau fungsionalitas aplikasi yang tidak dibatasi berdasarkan akses yang diberikan sesuai dengan peran pengguna
   * Kecukupan kendali untuk semua titik masuk dan keluar dari suatu aplikasi
   * Kata sandi yang masih disetting default
   * Kecukupan dokumentasi aplikasi
7. **Tools Pendukung Pengujian**
   1. Dalam melakukan pengujian selain metoda yang digunakan maka diperlukan tools pendukung yang beredar di pasaran, mulai dari aplikasi dalam bentuk satuan atau dalam bentuk sistem yang utuh. Tools yang disediakan juga ada yang bersifat berbayar atau lisensi.
   2. Dengan semakin berkembangnya teknologi alat bantu pengujian, maka instansi tidak harus selalu menggunakan alat/software yang terdapat dalam daftar ini.

Berikut ini beberapa tools komersial dan distro yang gratis (terbuka) untuk digunakan dalam membantu pengujian keamanan.

# Tabel 3.1 Tools Untuk Melakukan Pengujian Keamanan Web

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Owner** | **Licence** | **Platforms** |
| [**Abbey Scan**](https://misterscanner.com/) | MisterScanner | Free | SaaS |
| [**Acunetix WVS**](http://www.acunetix.com/) | Acunetix | Commercial/Free (Limited Capability) | Windows |
| [**Application Security on Cloud**](https://www.ibm.com/us-en/marketplace/application-security-on-cloud) | IBM | Commercial | SaaS |
| [**AppScan**](http://www-03.ibm.com/software/products/en/appscan-standard) | IBM | Commercial | Windows |
| [**App Scanner**](https://www.trustwave.com/Products/Application-Security/App-Scanner-Family/App-Scanner-Enterprise/) | Trustwave | Commercial | Windows |
| [**AppSpider**](http://www.rapid7.com/products/appspider/) | Rapid7 | Commercial | Windows |
| [**AppTrana Aplikasi Berbasis**](https://apptrana.indusface.com/basic/)[**WebSecurity Scan**](https://apptrana.indusface.com/basic/) | AppTrana | Free | SaaS |
| [**Arachni**](http://www.arachni-scanner.com/) | Arachni | Free for most use cases | Most platforms supported |
| [**AVDS**](https://www.scanmyserver.com/) | Beyond Security | Commercial/Free (Limited Capability) | SaaS |
| [**BlueClosure BC Detect**](https://www.blueclosure.com/) | BlueClosure | Commercial, 2 weeks trial | Most platforms supported |
| [**BREACHLOCK Dynamic**](https://www.breachlock.com/dynamic-application-security-testing/)[**Application Security Testing**](https://www.breachlock.com/dynamic-application-security-testing/) | Breachlock | Commercial | SaaS |
| [**Burp Suite**](http://www.portswigger.net/) | PortSwiger | Commercial/Free (Limited Capability) | Most platforms supported |
| [**Contrast**](https://contrastsecurity.com/) | Contrast Security | Commercial/Free (Full featured for 1 App) | SaaS or On-Premises |
| [**Detectify**](https://detectify.com/) | Detectify | Commercial | SaaS |
| [**Digifort- Inspect**](http://www.digifort.se/en/scanner) | Digifort | Commercial | SaaS |
| [**edgescan**](https://www.edgescan.com/) | edgescan | Commercial | SaaS |
| [**GamaScan**](http://www.gamasec.com/Gamascan.aspx) | GamaSec | Commercial | Windows |
| [**Grabber**](http://rgaucher.info/beta/grabber/) | Romain Gaucher | Open Source | Python 2.4, BeautifulSoup and PyXML |
| [**Gravityscan**](https://gravityscan.com/) | Defiant, Inc. | Commercial/Free (Limited Capability) | SaaS |
| [**Grendel-Scan**](http://sourceforge.net/p/grendel/code/ci/c59780bfd41bdf34cc13b27bc3ce694fd3cb7456/tree/) | David Byrne | Open Source | Windows, Linux and Macintosh |

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Owner** | **Licence** | **Platforms** |
| [**GoLismero**](http://www.golismero.com/) | GoLismero Team | GPLv2.0 | Windows, Linux and Macintosh |
| [**IKare**](http://www.ikare-monitoring.com/) | ITrust | Commercial | N/A |
| [**ImmuniWeb**](https://www.htbridge.com/immuniweb/) | High-Tech Bridge | Commercial/Free (Limited Capability) | SaaS |
| [**InsightVM**](https://www.rapid7.com/products/insightvm/) |  | Commercial with Free Trial | SaaS |
| [**Indusface Web Application**](https://www.indusface.com/index.php/products/web-application-scanning)[**Scanning**](https://www.indusface.com/index.php/products/web-application-scanning) | Indusface | Commercial/Free Trial | SaaS |
| [**N-Stealth**](http://www.nstalker.com/) | N-Stalker | Commercial | Windows |
| [**Nessus**](https://www.tenable.com/products/tenable-io/web-application-scanning/) | Tenable | Commercial | Windows |
| [**Netsparker**](http://www.mavitunasecurity.com/) | MavitunaSecurity | Commercial | Windows |
| [**Nexpose**](http://www.rapid7.com/products/nexpose-community-edition.jsp) | Rapid7 | Commercial/Free (Limited Capability) | Windows/Linux |
| [**Nikto**](http://www.cirt.net/nikto2) | CIRT | Open Source | Unix/Linux |
| [**Probely**](https://probely.com/) | Probely | Commercial/Free (Limited Capability) | SaaS |
| [**Proxy.app**](http://www.websecurify.com/desktop/proxy.html) | Websecurify | Commercial | Macintosh |
| [**QualysGuard**](http://www.qualys.com/products/qg_suite/was/) | Qualys | Commercial | N/A |
| [**Retina**](http://www.beyondtrust.com/Products/RetinaNetworkSecurityScanner/) | BeyondTrust | Commercial | Windows |
| [**Securus**](http://www.orvant.com/) | Orvant, Inc | Commercial | N/A |
| [**Sentinel**](http://www.whitehatsec.com/home/services/services.html) | WhiteHat Security | Commercial | N/A |
| [**SOATest**](http://www.parasoft.com/products/article.jsp?articleId=3169&redname=webtesting&referred=webtesting) | Parasoft | Commercial | Windows/Linux/Solaris |
| [**Tinfoil Security**](https://www.tinfoilsecurity.com/) | Tinfoil Security, Inc. | Commercial/Free (Limited Capability) | SaaS or On-Premises |
| [**Trustkeeper Scanner**](https://www.trustwave.com/external-vulnerability-scanning.php) | Trustwave SpiderLabs | Commercial | SaaS |
| [**Vega**](https://subgraph.com/vega/) | Subgraph | Open Source | Windows, Linux and Macintosh |
| [**Vex**](https://www.ubsecure.jp/en/about_ubsecure) | UBsecure | Commercial | Windows |

|  |  |  |  |
| --- | --- | --- | --- |
| [**Wapiti**](http://wapiti.sourceforge.net/) | Informática Gesfor | Open Source | Windows, Unix/Linux and Macintosh |
| [**Web Security Scanner**](https://www.defensecode.com/webscanner.php) | DefenseCode | Commercial | On-Premises |
| [**WebApp360**](http://www.tripwire.com/it-security-software/enterprise-vulnerability-management/web-application-vulnerability-scanning/) | TripWire | Commercial | Windows |
| [**WebCookies**](https://webcookies.org/) | WebCookies | Free | SaaS |
| [**WebInspect**](https://www.microfocus.com/en-us/products/webinspect-dynamic-analysis-dast) | Micro Focus | Commercial | Windows |
| [**WebReaver**](http://www.websecurify.com/desktop/webreaver.html) | Websecurify | Commercial | Macintosh |
| [**WebScanService**](http://www.german-websecurity.com/en/products/webscanservice/product-details/overview/) | German Web Security | Commercial | N/A |
| [**Websecurify Suite**](https://suite.websecurify.com/) | Websecurify | Commercial/Free (Limited Capability) | Windows, Linux, Macintosh |
| [**Wikto**](http://www.sensepost.com/research/wikto/) | Sensepost | Open Source | Windows |
| [**w3af**](http://www.w3af.org/) | w3af.org | GPLv2.0 | Linux and Mac |
| [**Zed Attack Proxy**](https://www.owasp.org/index.php/OWASP_Zed_Attack_Proxy_Project) | OWASP | Open Source | Windows Unix/Linux and Macintosh |

1. **Item Pengujian berbasis OWASP**

Beberapa pengujian dasar berbasis OWASP sebagai berikut:

# Tabel 3.2 Item Pengujian Berbasis OWASP

|  |  |  |
| --- | --- | --- |
| **Item Pengujian** | **Tujuan** | **Teknik** |
| Testing for Credentials Transported over an Encrypted Channel (OTG- AUTHN-001) | Memverifikasi bahwa otentikasi data pengguna ditransfer melalui saluran terenkripsi | Analisis paket header |
| Testing for default credentials (OTG- AUTHN-002) | Memverifikasi adakah penggunaan *default password* (*weak password*) | *Brute force password* |
| Testing for Weak lock out mechanism (OTG-AUTHN-003) | Memverifikasi apakah ada mekanisme penguncian akun | Mencoba *invalid login*  beberapa kali |
| Testing for bypassing authentication schema (OTG-AUTHN-004) | Memverifikasi *direct page request*  tanpa proses login | *Direct page request* |
| Test remember password functionality (OTG-AUTHN-005) | Memverifikasi bahwa *password* tidak disimpan dalam bentuk teks, tetapi hash | Analisis *cookies* |
| Testing for Browser cache weakness (OTG-AUTHN-006) | Memeriksa apakah aplikasi menginstruksikan browser untuk tidak ingat data sensitif. | Analisis browser *history* dan browser *cache* |
| Testing Directory traversal/file include (OTG-AUTHZ-001) | Menguji apakah aplikasi tahan terhadap *malicious string* | Memasukkan malicious string pada address bar |
| Testing for bypassing authorization schema (OTG-AUTHZ-002) | Percobaan akses ke dalam fungsi administrasi tanpa *login* | Manipulasi *http request header* |
| Testing for Privilege Escalation (OTG-AUTHZ-003) | Mencoba mendapatkan akses admin dari *user* | Manipulasi *http request header* |
| Testing for Insecure Direct Object References (OTG-AUTHZ-004) | Mencoba mengakses file tanpa login | Scanning |
| Testing for Bypassing Session Management Schema (OTG-SESS- 001) | Memeriksa apakah token pada cookies dan sesi lainnya dibuat dalam cara yang aman dan *unpredictable* | Analisis *cookies* |
| Testing for Cookies attributes (OTG- SESS-002) | Memeriksa apakah *cookies* menyimpan informasi masa *expired* di hard *drive client* | Analisis *Cookies* |

|  |  |  |
| --- | --- | --- |
| **Item Pengujian** | **Tujuan** | **Teknik** |
| Testing for Session Fixation (OTG- SESS-003) | Memeriksa apakah session yang diberikan pada client selalu diperbarui setelah proses authentifikasi | Analisis *Cookies* |
| Testing for Exposed Session Variables (OTG-SESS-004) | Memeriksa pada cookies apakah *Cookie, SessionID, Hidden Field* jelas terlihat | Analisis *Cookies* |
| Testing for Cross Site Request Forgery (CSRF) (OTG-SESS-005) | Memeriksa POST dan GET request pada halaman login apakah terimplementasi dengan baik | Analisis *Cookies* |

1. **Threat Modeling**

Threat Modeling adalah aktivitas yang dilakukan oleh Analis Keamanan TI untuk memodelkan ancaman keamanan yang mungkin terjadi pada aplikasi atau sistem yang dibangun. Ini bertujuan mendapatkan visibilitas ancaman *(threat)* yang relevan dan mengeliminasi yang tidak relevan terhadap sistem. Hasil dari kegiatan ini adalah daftar ancaman, tingkat keparahan, dan juga mitigasinya. Daftar ancaman ini nantinya menjadi dasar dalam pembuatan *attack scenario* ataupun *test case* pada tahap pengujian keamanan *(penetration testing).* Threat Modeling menggunakan metodologi STRIDE dari Microsoft.

Standar tools untuk melakukan threat modeling adalah menggunakan:

* + - * 1. Draw.io – dengan library threat modeling

Tool ini untuk membuat threat modeling umum *(high level)* dari aplikasi dan dependency nya

* + - * 1. OWASP Threat Dragon – kontekstual threat modeling

Tool ini untuk membuat *contextual threat modeling* yaitu versi detail dari spesifik bagian dari aplikasi yang ingin dilakukan threat modeling, seperti fungsionalitas login pada aplikasi.

1. **Secure Design Review (SDR)**

Beberapa pengujian dasar berbasis OWASP sebagai berikut:

1. Pelajari desain dan dokumen pendukung untuk memperoleh pemahaman dasar tentang proyek.

* Pertama, baca dokumen untuk memperoleh pemahaman umum tentang desain.
* Selanjutnya, gunakan "topi keamanan" dan tinjau kembali dokumen dengan mindset yang sadar ancaman.
* Buat catatan, tangkap ide dan observasi Anda untuk referensi di masa depan.
* Identifikasi masalah potensial untuk ditinjau nanti, tetapi pada tahap ini terlalu dini

1. Tanyakan tentang desain dan tanyakan pertanyaan penjelasan tentang ancaman dasar.

* Pastikan dokumen desain jelas dan lengkap.
* Jika ada kelalaian atau koreksi yang diperlukan, bantu perbaikinya di dokumen.
* Pahami desain cukup untuk dapat berbicara, tetapi tidak harus di tingkat ahli.
* Tanyakan kepada anggota tim apa yang mereka khawatirkan yang paling banyak; jika mereka tidak memiliki kekhawatiran keamanan, tanyakan pertanyaan lanjutan untuk mempelajari alasan mengapa tidak.

1. Identifikasi bagian desain yang paling penting untuk keamanan untuk ditinjau lebih dekat.

* Tinjau antarmuka, penyimpanan, dan komunikasi - ini akan menjadi pusat perhatian utama.
* Kerjakan dari permukaan serangan yang paling terbuka menuju aset yang paling berharga, seperti yang dilakukan oleh penyerang yang ditentukan.
* Evaluasi sejauh mana desain menangani keamanan secara eksplisit.
* Jika diperlukan, tunjukkan perlindungan utama dan minta untuk disebutkan di desain sebagai fitur penting.

1. Kolaborasi dengan desainer untuk mengidentifikasi risiko dan membahas mitigasi.

* Sebagai reviewer, berikan perspektif keamanan tentang risiko dan mitigasi yang dibutuhkan.
* Pertimbangkan menggambar skenario yang menggambarkan bagaimana perubahan keamanan dapat menguntungkan di masa depan untuk membujuk desainer tentang kebutuhan mitigasi.
* Tawarkan lebih dari satu solusi untuk masalah ketika Anda dapat melakukannya, dan bantu desainer melihat kelebihan dan kekurangan alternatif ini.
* Terima bahwa desainer mendapatkan kata terakhir, karena mereka pada akhirnya bertanggung jawab atas desain.
* Dokumentasikan pertukaran ide, termasuk apa yang akan atau tidak akan masuk ke dalam desain.

1. Tulis laporan ringkasan tentang temuan dan rekomendasi.

* Atur laporan seputar perubahan desain yang spesifik yang menangani risiko keamanan.
* Habiskan sebagian besar upaya dan tinta Anda pada masalah prioritas tertinggi, dan proporsional lebih sedikit pada prioritas yang lebih rendah.
* Sarankan alternatif dan strategi, tanpa mencoba melakukan pekerjaan desainer untuk mereka.
* Prioritaskan temuan dan rekomendasi, menggunakan penilaian prioritas. (Klasifikasikan poin sebagai Harus/Sebaiknya/Rekemondasi
* Berfokuslah pada keamanan, tetapi jangan ragu untuk memberikan komentar terpisah untuk pertimbangan desainer juga.

1. Tindak lanjuti dengan perubahan desain selanjutnya untuk mengonfirmasi resolusi sebelum keluar.

* Untuk perubahan desain keamanan utama, Anda mungkin ingin berkolaborasi dengan desainer untuk memastikan bahwa perubahan dilakukan dengan benar.
* Jika ada perbedaan pendapat, peninjau harus menyertakan pernyataan tentang posisi dan rekomendasi khusus yang tidak diikuti untuk menandainya sebagai masalah terbuka.

1. **Static Application Security Testing (SAST)**

SAST adalah aktivitas untuk melakukan pemindaian kerentanan pada *source code* aplikasi menggunakan SAST tools.

Tools yang bisa digunakan jika pengembangan aplikasi tidak menggunakan CI/CD pipelines:

* Semgrep
* Sonarqube

Tools yang digunakan jika pengembangan aplikasi menggunakan CI/CD pipelines:

* Gitlab-SAST.yml default
* Semgrep docker image
* Sonarqube docker image

1. **Secure Code Review**

Secure Code Review adalah aktivitas untuk memeriksa celah keamanan pada *source code* aplikasi menggunakan Teknik manual analysis dan semi-manual/menggunakan tools.

Manual:

* [OWASP Code Review Guide | OWASP Foundation](https://owasp.org/www-project-code-review-guide/)

Semi-manual/Tools:

* [Code Checker | Free Code Security Tool Powered by AI | Snyk](https://snyk.io/code-checker/)
* <https://chat.openai.com/>

Scan per-file pada VS Code:

* [HCL AppScan CodeSweep - Visual Studio Marketplace](https://marketplace.visualstudio.com/items?itemName=HCLTechnologies.hclappscancodesweep)

- scan seluruh code ketika sebuah file source code di save

- setelah dicoba banyak "positif rate" nya dan lebih banyak nemuin celah dari tool sejenis yg lain

- tidak bisa scan lebih dari satu file sekaligus, atau scan keseluruhan project/repo.

* [SonarLint - Visual Studio Marketplace](https://marketplace.visualstudio.com/items?itemName=SonarSource.sonarlint-vscode)

- Suggestion/feedback code security muncul mulai programmer melakukan coding di VS Code (realtime), dan muncul pada tab "PROBLEMS"

- menurut percobaan, sedikit nemu security code problem dibanding HCL AppScabn CodeSweep

- tidak bisa scan lebih dari satu file sekaligus, atau scan keseluruhan project/repo.

Scan Seluruh Project/Repo pada VS Code:



1. **Software Composition Analysis (SCA)**

SCA adalah aktivitas untuk melakukan pemindaian kerentanan pada komponen-komponen pihak ketiga *(library)* yang digunakan sebagai penyusun aplikasi.

Tools yang bisa digunakan jika pengembangan aplikasi tidak menggunakan CI/CD pipelines bisa menggunakan extension berikut pada IDE Visual Studio Code (VS Code):

* [Code Project Scanner - Visual Studio Marketplace](https://marketplace.visualstudio.com/items?itemName=mk2000.code-project-scanner)

Tool ini dapat melakukan pemindaian pada keseluruhan *source code* yang sedang dibuka di VS Code dan langsung dapat terlihat hasil pemindaiannya

* [Snyk Security - Code, Open Source Dependencies, IaC Configurations - Visual Studio Marketplace](https://marketplace.visualstudio.com/items?itemName=snyk-security.snyk-vulnerability-scanner)

Tool ini membutuhkan *pre-requisites* dari dua tools yang harus sebelumnya terinstal di komputer, yaitu “Snyk Open Source” dan “Snyk CLI”.

Pemindaian akan otomatis dilakukan terhadap keseluruhan *source codes* yang ada pada project yang sedang dibuka pada VS Code. Pemindaian juga dapat dilkukan secara manual dengan menekan tombol pada *plugin tag*-nya.

Fitur SCA berada pada tab “OPEN SOURCE SECURITY” pada bagian Snyk plugin tab yang ada pada VS Code.

Tools yang digunakan jika pengembangan aplikasi menggunakan CI/CD pipelines:

* [OWASP Dependency-Check | OWASP Foundation](https://owasp.org/www-project-dependency-check/)

Tool ini memiliki versi docker image yang dapat diintegrasikan pada CI/CI pipelines

1. **Dynamic Application Security Testing (DAST)**

DAST adalah aktivitas yang dilakukan untuk menguji keamanan pada saat aplikasi berjalan di lingkungan “Testing” server.

Tools yang bisa digunakan jika pengembangan aplikasi tidak menggunakan CI/CD pipelines:

* Burp Suite Professional
* OWASP Zed Attack Proxy (ZAP)
* InsightVM
* Nikto
* Nuclei

Tools yang digunakan jika pengembangan aplikasi menggunakan CI/CD pipelines:

* OWASP ZAP docker image
* [Dastardly, from Burp Suite – PortSwigger](https://portswigger.net/burp/dastardly)